

Lecture Notes in Artificial Intelligence 2431

Subseries of Lecture Notes in Computer Science
Edited by J. G. Carbonell and J. Siekmann

Lecture Notes in Computer Science
Edited by G. Goos, J. Hartmanis, and J. van Leeuwen

Springer

Berlin

Heidelberg

New York

Barcelona

Hong Kong

London

Milan

Paris

Tokyo

Tapio Elomaa Heikki Mannila
Hannu Toivonen (Eds.)

Principles of Data Mining and Knowledge Discovery

6th European Conference, PKDD 2002
Helsinki, Finland, August 19-23, 2002
Proceedings



Springer

Series Editors

Jaime G. Carbonell, Carnegie Mellon University, Pittsburgh, PA, USA
Jörg Siekmann, University of Saarland, Saarbrücken, Germany

Volume Editors

Tapio Elomaa
Heikki Mannila
Hannu Toivonen
University of Helsinki, Department of Computer Science
P.O. Box 26, 00014 Helsinki, Finland
E-mail: {elomaa, heikki.mannila, hannu.toivonen}@cs.helsinki.fi

Cataloging-in-Publication Data applied for

Die Deutsche Bibliothek - CIP-Einheitsaufnahme

Principles of data mining and knowledge discovery : 6th European conference ;
proceedings / PKDD 2002, Helsinki, Finland, August 19 - 23, 2002. Tapio
Elomaa ... (ed.). - Berlin ; Heidelberg ; New York ; Barcelona ; Hong Kong ;
London ; Milan ; Paris ; Tokyo : Springer, 2002
(Lecture notes in computer science ; Vol. 2431 : Lecture notes in
artificial intelligence)
ISBN 3-540-44037-2

CR Subject Classification (1998): I.2, H.2, J.1, H.3, G.3, I.7, F.4.1

ISSN 0302-9743

ISBN 3-540-44037-2 Springer-Verlag Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer-Verlag. Violations are liable for prosecution under the German Copyright Law.

Springer-Verlag Berlin Heidelberg New York,
a member of BertelsmannSpringer Science+Business Media GmbH

<http://www.springer.de>

© Springer-Verlag Berlin Heidelberg 2002
Printed in Germany

Typesetting: Camera-ready by author, data conversion by DA-TeX Gerd Blumenstein
Printed on acid-free paper SPIN: 10870106 06/3142 5 4 3 2 1 0

Preface

We are pleased to present the proceedings of the *13th European Conference on Machine Learning* (LNAI 2430) and the *6th European Conference on Principles and Practice of Knowledge Discovery in Databases* (LNAI 2431). These two conferences were colocated in Helsinki, Finland during August 19–23, 2002. ECML and PKDD were held together for the second year in a row, following the success of the colocation in Freiburg in 2001. Machine learning and knowledge discovery are two highly related fields and ECML/PKDD is a unique forum to foster their collaboration.

The benefit of colocation to both the machine learning and data mining communities is most clearly displayed in the common workshop, tutorial, and invited speaker program. Altogether six workshops and six tutorials were organized on Monday and Tuesday. As invited speakers we had the pleasure to have Erkki Oja (Helsinki Univ. of Technology), Dan Roth (Univ. of Illinois, Urbana-Champaign), Bernhard Schölkopf (Max Planck Inst. for Biological Cybernetics, Tübingen), and Padhraic Smyth (Univ. of California, Irvine).

The main events ran from Tuesday until Friday, comprising 41 ECML technical papers and 39 PKDD papers. In total, 218 manuscripts were submitted to these two conferences: 95 to ECML, 70 to PKDD, and 53 as joint submissions. All papers were assigned at least three reviewers from our international program committees. Out of the 80 accepted papers 31 were first accepted conditionally; the revised manuscripts were accepted only after the conditions set by the reviewers had been met.

Our special thanks go to the tutorial chairs Johannes Fürnkranz and Myra Spiliopoulou and the workshop chairs Hendrik Blockeel and Jean-François Boulicaut for putting together an exiting combined tutorial and workshop program. Also the challenge chair Petr Berka deserves our sincerest gratitude. All the members of both program committees are thanked for devoting their expertise to the continued success of ECML and PKDD. The organizing committee chaired by Helena Ahonen-Myka worked hard to make the conferences possible. A special mention has to be given to Oskari Heinonen for designing and maintaining the web pages and Ilkka Koskenniemi for maintaining CyberChair, which was developed by Richard van de Stadt. We thank Alfred Hofmann of Springer-Verlag for cooperation in publishing these proceedings. We gratefully acknowledge the financial support of the Academy of Finland and KDNet.

We thank all the authors for contributing to what in our mind is a most interesting technical program for ECML and PKDD. We trust that the week in late August was most enjoyable for all members of both research communities.

June 2002

Tapio Elomaa
Heikki Mannila
Hannu Toivonen

ECML/PKDD-2002 Organization

Executive Committee

Program Chairs:	Tapio Elomaa (Univ. of Helsinki) Heikki Mannila (Helsinki Inst. for Information Technology and Helsinki Univ. of Technology) Hannu Toivonen (Nokia Research Center and Univ. of Helsinki)
Tutorial Chairs:	Johannes Fürnkranz (Austrian Research Inst. for Artificial Intelligence) Myra Spiliopoulou (Leipzig Graduate School of Management)
Workshop Chairs:	Hendrik Blockeel (Katholieke Universiteit Leuven) Jean-François Boulicaut (INSA Lyon)
Challenge Chair:	Petr Berka (University of Economics, Prague)
Organizing Chair:	Helena Ahonen-Myka (Univ. of Helsinki)
Organizing Committee:	Oskari Heinonen, Ilkka Koskenniemi, Greger Lindén, Pirjo Moen, Matti Nykänen, Anna Pienimäki, Ari Rantanen, Juho Rousu, Marko Salmenkivi (Univ. of Helsinki)

ECML Program Committee

H. Blockeel, Belgium	A. Hyvärinen, Finland
I. Bratko, Slovenia	T. Joachims, USA
P. Brazdil, Portugal	Y. Kodratoff, France
H. Boström, Sweden	I. Kononenko, Slovenia
W. Burgard, Germany	S. Kramer, Germany
N. Cristianini, USA	M. Kubat, USA
J. Cussens, UK	N. Lavrač, Slovenia
L. De Raedt, Germany	C.X. Ling, Canada
M. Dorigo, Belgium	R. López de Màntaras, Spain
S. Džeroski, Slovenia	D. Malerba, Italy
F. Esposito, Italy	S. Matwin, Canada
P. Flach, UK	R. Meir, Israel
J. Fürnkranz, Austria	J. del R. Millán, Switzerland
J. Gama, Portugal	K. Morik, Germany
J.-G. Ganascia, France	H. Motoda, Japan
T. Hofmann, USA	R. Nock, France
L. Holmström, Finland	E. Plaza, Spain

G. Paliouras, Greece	H. Tirri, Finland
J. Rousu, Finland	P. Turney, Canada
L. Saitta, Italy	R. Vilalta, USA
T. Scheffer, Germany	P. Vitányi, The Netherlands
M. Sebag, France	S. Weiss, USA
J. Shawe-Taylor, UK	G. Widmer, Austria
A. Siebes, The Netherlands	R. Wirth, Germany
D. Sleeman, UK	S. Wrobel, Germany
M. van Someren, The Netherlands	Y. Yang, USA
P. Stone, USA	

PKDD Program Committee

H. Ahonen-Myka, Finland	S. Morishita, Japan
E. Baralis, Italy	H. Motoda, Japan
J.-F. Boulicaut, France	G. Nakhaeizadeh, Germany
N. Cercone, Canada	Z.W. Raś, USA
B. Crémilleux, France	J. Rauch, Czech Republic
L. De Raedt, Germany	G. Ritschard, Switzerland
L. Dehaspe, Belgium	M. Sebag, France
S. Džeroski, Slovenia	F. Sebastiani, Italy
M. Ester, Canada	M. Sebban, France
R. Feldman, Israel	B. Seeger, Germany
P. Flach, UK	A. Siebes, The Netherlands
E. Frank, New Zealand	A. Skowron, Poland
A. Freitas, Brazil	M. van Someren, The Netherlands
J. Fürnkranz, Austria	M. Spiliopoulou, Germany
H.J. Hamilton, Canada	N. Spyrtatos, France
J. Han, Canada	E. Suzuki, Japan
R. Hilderman, Canada	A.-H. Tan, Singapore
S.J. Hong, USA	S. Tsumoto, Japan
S. Kaski, Finland	A. Unwin, Germany
D. Keim, USA	J. Wang, USA
J.-U. Kietz, Switzerland	K. Wang, Canada
R. King, UK	L. Wehenkel, Belgium
M. Klemettinen, Finland	D. Wettschereck, Germany
W. Klösgen, Germany	G. Widmer, Austria
Y. Kodratoff, France	R. Wirth, Germany
J.N. Kok, The Netherlands	S. Wrobel, Germany
S. Kramer, Germany	M. Zaki, USA
S. Matwin, Canada	

Additional Reviewers

N. Abe	S. Haustein	L. Peña
F. Aiolfi	J. He	Y. Peng
Y. Altun	K.G. Herbert	J. Petrak
S. de Amo	J. Himberg	V. Phan Luong
A. Appice	J. Hipp	K. Rajaraman
E. Armengol	S. Hoche	T. Reinartz
T.G. Ault	J. Hosking	I. Renz
J. Azé	E. Hüllermeier	C. Rigotti
M.T. Basile	P. Juvan	F. Rioult
A. Bonarini	M. Kääriäinen	M. Robnik-Šikonja
R. Bouckaert	D. Kalles	M. Roche
P. Brockhausen	V. Karkaletsis	B. Rosenfeld
M. Brodie	A. Karwath	S. Rüping
W. Buntine	K. Kersting	M. Salmenkivi
J. Carbonell	J. Kindermann	A.K. Seewald
M. Ceci	R. Klinkenberg	H. Shan
S. Chikkanna-Naik	P. Koistinen	J. Sinkkonen
S. Chiusano	C. Köpf	J. Struyf
R. Cicchetti	R. Kosala	R. Taouil
A. Clare	W. Kusters	J. Taylor
M. Degemmis	M.-A. Krogel	L. Todorovski
J. Demsar	M. Kukar	T. Urbancic
F. De Rosis	L. Lakhal	K. Vasko
N. Di Mauro	G. Lebanon	H. Wang
G. Dorffner	S.D. Lee	Y. Wang
G. Dounias	F. Li	M. Wiering
N. Durand	J.T. Lindgren	S. Wu
P. Erästö	J. Liu	M.M. Yin
T. Erjavec	Y. Liu	F. Zambetta
J. Farrand	M.-C. Ludl	B. Ženko
S. Ferilli	S. Mannor	J. Zhang
P. Floréen	R. Meo	S. Zhang
J. Franke	N. Meuleau	T. Zhang
T. Gaertner	H. Mogg-Schneider	M. Zlochin
P. Gallinari	R. Natarajan	B. Zupan
P. Garza	S. Nijssen	
A. Giacometti	G. Paaß	

Tutorials

Text Mining and Internet Content Filtering

José María Gómez Hidalgo

Formal Concept Analysis

Gerd Stumme

Web Usage Mining for E-business Applications

Myra Spiliopoulou, Bamshad Mobasher, and Bettina Berendt

Inductive Databases and Constraint-Based Mining

Jean-François Boulicaut and Luc De Raedt

An Introduction to Quality Assessment in Data Mining

Michalis Vazirgiannis and M. Halkidi

Privacy, Security, and Data Mining

Chris Clifton

Workshops

Integration and Collaboration Aspects of Data Mining, Decision Support and Meta-Learning

Marko Bohanec, Dunja Mladenić, and Nada Lavrač

Visual Data Mining

Simeon J. Simoff, Monique Noirhomme-Fraiture, and Michael H. Böhlen

Semantic Web Mining

Bettina Berendt, Andreas Hotho, and Gerd Stumme

Mining Official Data

Paula Brito and Donato Malerba

Knowledge Discovery in Inductive Databases

Mika Klemettinen, Rosa Meo, Fosca Giannotti, and Luc De Raedt

Discovery Challenge Workshop

Petr Berka, Jan Rauch, and Shusaku Tsumoto

Table of Contents

Contributed Papers

Optimized Substructure Discovery for Semi-structured Data	1
<i>Kenji Abe, Shinji Kawasoe, Tatsuya Asai, Hiroki Arimura, and Setsuo Arikawa</i>	
Fast Outlier Detection in High Dimensional Spaces	15
<i>Fabrizio Angiulli and Clara Pizzuti</i>	
Data Mining in Schizophrenia Research – Preliminary Analysis	27
<i>Stefan Arnborg, Ingrid Agartz, Håkan Hall, Erik Jönsson, Anna Sillén, and Göran Sedvall</i>	
Fast Algorithms for Mining Emerging Patterns	39
<i>James Bailey, Thomas Manoukian, and Kotagiri Ramamohanarao</i>	
On the Discovery of Weak Periodicities in Large Time Series	51
<i>Christos Berberidis, Ioannis Vlahavas, Walid G. Aref, Mikhail Atallah, and Ahmed K. Elmagarmid</i>	
The Need for Low Bias Algorithms in Classification Learning from Large Data Sets	62
<i>Damien Brain and Geoffrey I. Webb</i>	
Mining All Non-derivable Frequent Itemsets	74
<i>Toon Calders and Bart Goethals</i>	
Iterative Data Squashing for Boosting Based on a Distribution-Sensitive Distance	86
<i>Yuta Choki and Einoshin Suzuki</i>	
Finding Association Rules with Some Very Frequent Attributes	99
<i>Frans Coenen and Paul Leng</i>	
Unsupervised Learning: Self-aggregation in Scaled Principal Component Space	112
<i>Chris Ding, Xiaofeng He, Hongyuan Zha, and Horst Simon</i>	
A Classification Approach for Prediction of Target Events in Temporal Sequences	125
<i>Carlotta Domeniconi, Chang-shing Perng, Ricardo Vilalta, and Sheng Ma</i>	
Privacy-Oriented Data Mining by Proof Checking	138
<i>Amy Felty and Stan Matwin</i>	

XII Table of Contents

Choose Your Words Carefully: An Empirical Study of Feature Selection Metrics for Text Classification	150
<i>George Forman</i>	
Generating Actionable Knowledge by Expert-Guided Subgroup Discovery	163
<i>Dragan Gamberger and Nada Lavrač</i>	
Clustering Transactional Data	175
<i>Fosca Giannotti, Cristian Gozzi, and Giuseppe Manco</i>	
Multiscale Comparison of Temporal Patterns in Time-Series Medical Databases	188
<i>Shoji Hirano and Shusaku Tsumoto</i>	
Association Rules for Expressing Gradual Dependencies	200
<i>Eyke Hüllermeier</i>	
Support Approximations Using Bonferroni-Type Inequalities	212
<i>Szymon Jaroszewicz and Dan A. Simovici</i>	
Using Condensed Representations for Interactive Association Rule Mining	225
<i>Baptiste Jeudy and Jean-François Boulicaut</i>	
Predicting Rare Classes: Comparing Two-Phase Rule Induction to Cost-Sensitive Boosting	237
<i>Mahesh V. Joshi, Ramesh C. Agarwal, and Vipin Kumar</i>	
Dependency Detection in MobiMine and Random Matrices	250
<i>Hillol Kargupta, Krishnamoorthy Sivakumar, and Samiran Ghosh</i>	
Long-Term Learning for Web Search Engines	263
<i>Charles Kemp and Kotagiri Ramamohanarao</i>	
Spatial Subgroup Mining Integrated in an Object-Relational Spatial Database	275
<i>Willi Klösgen and Michael May</i>	
Involving Aggregate Functions in Multi-relational Search	287
<i>Arno J. Knobbe, Arno Siebes, and Bart Marseille</i>	
Information Extraction in Structured Documents Using Tree Automata Induction	299
<i>Raymond Kosala, Jan Van den Bussche, Maurice Bruynooghe, and Hendrik Blockeel</i>	
Algebraic Techniques for Analysis of Large Discrete-Valued Datasets	311
<i>Mehmet Koyutürk, Ananth Grama, and Naren Ramakrishnan</i>	
Geography of Differences between Two Classes of Data	325
<i>Jinyan Li and Limsoon Wong</i>	

Rule Induction for Classification of Gene Expression Array Data	338
<i>Per Lidén, Lars Asker, and Henrik Boström</i>	
Clustering Ontology-Based Metadata in the Semantic Web	348
<i>Alexander Maedche and Valentin Zacharias</i>	
Iteratively Selecting Feature Subsets for Mining from High-Dimensional Databases	361
<i>Hiroshi Mamitsuka</i>	
SVM Classification Using Sequences of Phonemes and Syllables	373
<i>Gerhard Paaß, Edda Leopold, Martha Larson, Jörg Kindermann, and Stefan Eickeler</i>	
A Novel Web Text Mining Method Using the Discrete Cosine Transform	385
<i>Laurence A.F. Park, Marimuthu Palaniswami, and Kotagiri Ramamohanarao</i>	
A Scalable Constant-Memory Sampling Algorithm for Pattern Discovery in Large Databases	397
<i>Tobias Scheffer and Stefan Wrobel</i>	
Answering the Most Correlated N Association Rules Efficiently	410
<i>Jun Sese and Shinichi Morishita</i>	
Mining Hierarchical Decision Rules from Clinical Databases Using Rough Sets and Medical Diagnostic Model	423
<i>Shusaku Tsumoto</i>	
Efficiently Mining Approximate Models of Associations in Evolving Databases	435
<i>Adriano Veloso, Bruno Gusmão, Wagner Meira Jr., Marcio Carvalho, Sriini Parthasarathy, and Mohammed Zaki</i>	
Explaining Predictions from a Neural Network Ensemble One at a Time	449
<i>Robert Wall, Pádraig Cunningham, and Paul Walsh</i>	
Structuring Domain-Specific Text Archives by Deriving a Probabilistic XML DTD	461
<i>Karsten Winkler and Myra Spiliopoulou</i>	
Separability Index in Supervised Learning	475
<i>Djamel A. Zighed, Stéphane Lallich, and Fabrice Muhlenbach</i>	

Invited Papers

Finding Hidden Factors Using Independent Component Analysis	488
<i>Erkki Oja</i>	

XIV Table of Contents

Reasoning with Classifiers	489
<i>Dan Roth</i>	
A Kernel Approach for Learning from Almost Orthogonal Patterns	494
<i>Bernhard Schölkopf, Jason Weston, Eleazar Eskin, Christina Leslie,</i> <i>and William Stafford Noble</i>	
Learning with Mixture Models: Concepts and Applications	512
<i>Padhraic Smyth</i>	
Author Index	513

Optimized Substructure Discovery for Semi-structured Data

Kenji Abe¹, Shinji Kawasoe¹, Tatsuya Asai¹,
Hiroki Arimura^{1,2}, and Setsuo Arikawa¹

¹ Department of Informatics, Kyushu University
6-10-1 Hakozaki Higashi-ku, Fukuoka 812-8581, Japan
{k-abe,s-kawa,t-asai,arim,arikawa}@i.kyushu-u.ac.jp

² PRESTO, JST, Japan

Abstract. In this paper, we consider the problem of discovering interesting substructures from a large collection of semi-structured data in the framework of optimized pattern discovery. We model semi-structured data and patterns with labeled ordered trees, and present an efficient algorithm that discovers the best labeled ordered trees that optimize a given statistical measure, such as the information entropy and the classification accuracy, in a collection of semi-structured data. We give theoretical analyses of the computational complexity of the algorithm for patterns with bounded and unbounded size. Experiments show that the algorithm performs well and discovered interesting patterns on real datasets.

1 Introduction

Recent progress of network and storage technologies have increased the species and the amount of electronic data, called *semi-structured data* [2], such as Web pages and XML data [26]. Since such semi-structured data are heterogeneous and huge collections of weakly structured data that have no rigid structures, it is difficult to directly apply traditional data mining techniques to these semi-structured data. Thus, there are increasing demands for efficient methods for extracting information from semi-structured data [10,18,19,27].

In this paper, we consider a data mining problem of discovering characteristic substructure from a large collection of semi-structured data. We model semi-structured data and patterns with *labeled ordered trees*, where each node has a constant label and has arbitrary many children ordered from left to right. For example, Fig. 1 shows a semi-structured data encoded as XML data, where the data is nested by pairs `<tag>` and `</tag>` of balanced parentheses.

Our framework of data mining is *optimized pattern discovery* [20], which has its origin in the statistical decision theory in 1970's [11] and extensively studied in the fields of machine learning, computational learning theory, and data mining for the last decade [5,13,14,16,17,20,23]. In optimized pattern discovery, the input data is a collection of semi-structured data with binary labels indicating if a user is interested in the data. Then, the goal of a mining algorithm is to discover

```

<people>
  <person age="40">
    <name> Alan</name> <tel> 7786</tel> <tel> 2133</tel> </person>
  <person height="155">
    <name> <first> Sara</first> <last> Green</last> </name> </person>
  <person age="33" height="187"> <name> Fred</name> </person>
</people>

```

Fig. 1. An Example of semi-structured data

such patterns that optimize a given statistical measure, such as the classification error [11] and the information entropy [22] over all possible patterns in the input collection. In other words, the goal is not to find *frequent* patterns but to find *optimal* patterns.

Intuitively speaking, the purpose of optimized pattern discovery is to find the patterns that characterize a given subset of data and separate them from the rest of the database [6]. For instance, suppose that we are given a collection of movie information entries from an online movie database¹. To find a characteristic patterns to its subcollection consisting only of *action movies*, a simplest approach is to find those patterns frequently appearing in action movies. However, if a characteristic pattern has small frequency, then its occurrences may be hidden by many trivial but frequent patterns.

Another approach is to find those patterns that appear more frequently in action movies but less in the other movies. By this, we can expect to find slight but interesting patterns that characterize the specified sub-collection. The precise description of optimized pattern discovery will be given in Section 2.1.

1.1 Main Results

We present an efficient algorithm OPTT for discovering optimized labeled ordered trees from a large collection of labeled ordered trees based on an efficient frequent tree miner FREQT devised in our previous paper [7]. Unlike previous tree miners equipped with a straightforward generate-and-test strategy [19] or Apriori-like subset-lattice search [15,27], FREQT is an efficient incremental tree miner that simultaneously constructs the set of frequent patterns and their occurrences level by level.

In particular, since we cannot use the standard frequency thresholding as in Apriori-like algorithms [3] in optimized pattern discovery, the potential search space will be quite large. To overcome this difficulty, we employ the following techniques to implement an efficient tree minor:

- Based on the rightmost expansion technique of [7,28], which is a generalization of the item set-enumeration tree technique of Bayardo [8], we can efficiently generate all labeled ordered trees without duplicates.

¹ E.g., *Internet Movie database*, <http://www.imdb.com/>

- Using the rightmost leaf occurrence representation [7], we can store and update the occurrences of patterns compactly.
- Using the convexity of the impurity function ψ , we can efficiently prune unpromising branch in a search process by the method of [21].

Then, we present theoretical results on the performance and the limitation of our tree miner OPTT. For patterns of bounded size k , we show a non-trivial $O(k^{k+1}b^k N)$ time upperbound of the running time of the algorithm OPTT, where N and b is the total size the maximum branching of an input database $\mathcal{D}$. This says that if k and b are small constants as in many applications, then the algorithm runs linear time in N , while a generate-and-test algorithm may have super-linear time complexity when the number of unique labels grows.

In contrast, for patterns of unbounded size, we also show that the optimal pattern discovery problem for labeled ordered trees is hard to approximate. Precisely, the maximum agreement problem, which is a dual problem of the classification error minimization, is not polynomial time approximable with approximation ratio strictly less than $770/767$ if $P \neq NP$.

Finally, we run some experiments on real datasets and show that the algorithm is scalable and useful in Web and XML mining. In particular, we observe that the pruning with convexity is effective in a tree miner and that the depth-first search strategy is an attractive choice from the view of space complexity.

1.2 Organization

The rest of this paper is organized as follows. In Section 2, we prepare basic notions and definitions. In Section 3, we present our algorithm OPTT for solving the optimized pattern discovery problem for labeled ordered trees. In Section 4, we give theoretical analysis on the computational complexity of the algorithm and the problem. In Section 5, we run experiments on real datasets to evaluate the proposed mining algorithm. In Section 6, we conclude.

1.3 Related Works

There are many studies on semi-structured databases [2,26]. In contrast, there have not been many studies on *semi-structured data mining* [7,10,15,18,19,27,28]. Among them, most of the previous studies [7,10,19,27,28] consider frequent pattern discovery but not optimized pattern discovery. We also note that most of these works other than [7,28] are based on a straightforward generate-and-test search or Apriori-like levelwise search and does not use the notion of the rightmost expansion.

On the other hand, the algorithm by Matsuda and Motoda *et al.* [18] finds near optimal tree-like patterns using a greedy search method called the graph-based induction. Inokuchi *et al.* [15] presented an Apriori-style algorithm for finding frequent subgraphs and generalized it for optimized pattern discovery.

Most related work would be a tree miner for labeled ordered tree with gaps by Zaki [28], recently proposed independently to our previous work [7]. The

algorithm uses the essentially same enumeration technique to ours, and equipped with a number of interesting ideas that speed-up the search.

2 Preliminaries

2.1 Optimized Pattern Discovery

We give a problem description of optimized pattern discovery according to [11,20]. A *sample* is a pair $(\mathcal{D}, \xi)$ of a collection $\mathcal{D} = \{D_1, \dots, D_m\}$, called the *database*, of *document trees* and an *objective attribute* $\xi : \mathcal{D} \rightarrow \{0, 1\}$ that indicates if a user is interested in a document tree. A tree $D \in \mathcal{D}$ is *positive* if $\xi(D) = 1$ and *negative* otherwise. We are also given a class $\mathcal{P}$ of patterns, i.e., the class of labeled ordered trees. For a fixed database $\mathcal{D}$, each pattern $T \in \mathcal{P}$ can be identified as a binary attribute $T : \mathcal{D} \rightarrow \{0, 1\}$ through tree matching, and splits the database $\mathcal{D}$ into disjoint sets D_1 and D_0 of *matched* and *unmatched* documents, where $\mathcal{D}_\alpha = \{D \in \mathcal{D} \mid T(D) = \alpha\}$ for every $\alpha = 0, 1$.

A natural question here is what patterns are better to characterize the subset D_1 relative to D_0 . We measure the goodness of a pattern $T : \mathcal{D} \rightarrow \{0, 1\}$ by using an *impurity function* $\psi : [0, 1] \rightarrow \mathbf{R}$ that is a convex function having the maximum value at $1/2$ and the minimum value at 0 and 1 , and represents the ambiguity of the split [11]. For example, the classification error $\psi_1(x) = \min(x, 1 - x)$ [16], the information entropy $\psi_2(x) = -x \log x - (1 - x) \log(1 - x)$ [22], and the Gini index functions $\psi_3(x) = 2x(1 - x)$ [11] are instances of impurity functions ψ . Now, we state the *Optimized Pattern Discovery Problem* for a class $\mathcal{P}$ of patterns and with impurity function ψ as follows:

Optimized Pattern Discovery Problem. The goal of the optimized pattern discovery is to discover a pattern $T \in \mathcal{P}$ that minimizes the following cost function induced from ψ :

$$\Psi_{S, \xi}(T) = (N_1^T + N_1^F) \cdot \psi\left(\frac{N_1^T}{N_1^T + N_1^F}\right) + (N_0^T + N_0^F) \cdot \psi\left(\frac{N_0^T}{N_0^T + N_0^F}\right) \quad (1)$$

where N_α^β is the number of the trees $D \in \mathcal{D}$ that has $T(D) = \alpha$ and $\xi(D) = \beta$ for every $\alpha \in \{1, 0\}$ and $\beta \in \{T, F\}$.

Then, such a pattern T is called *optimal* w.r.t. ψ . We note that the function $\Psi_{S, \xi}(T)$ above is directly optimized in our framework, while $\Psi_{S, \xi}(T)$ is used only as a guide of greedy search in many empirical learning algorithms such as *C4.5* [22].

Furthermore, it is shown that any algorithm that efficiently solves the optimized pattern discovery problem can approximate an arbitrary unknown distribution of labeled data well within a given class of patterns [16]. Thus, the optimized pattern discovery has been extensively studied and applied to the discovery of geometric patterns or numeric association rules [13,14,17], association rule [20,23], and string patterns [5,24].

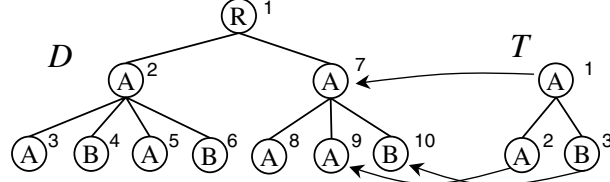


Fig. 2. A data tree D and a pattern tree T on the set $\mathcal{L} = \{A, B\}$ of labels

2.2 Labeled Ordered Trees

We define the class of labeled ordered trees as a formal model of semi-structured data and patterns [2] according to [7]. For the definitions of basic terminologies on sets, trees, and graphs, we refer to a textbook by, e.g. [4]. For a binary relation B , the *transitive closure* of B is denoted by B^+ .

First, we fix a possibly infinite alphabet $\mathcal{L} = \{\ell, \ell_0, \ell_1, \dots\}$ of *labels*. Then, a *labeled ordered tree on $\mathcal{L}$* is a rooted, connected directed acyclic graph T such that each node is labeled by an element of $\mathcal{L}$ and all node but the root have the unique parent and their children are ordered from left to right [4]. Note that the term *ordered* means the order *not* on labels but on children. More precisely, a *labeled ordered tree* of size $k \geq 0$ is represented to be a 6-tuple $T = (V, E, B, \mathcal{L}, L, v_0)$, where V is a set of nodes, $E \subseteq V^2$ is the set of edges (or the direct child relation), $B \subseteq V^2$ is the direct sibling relation, $L : V \rightarrow \mathcal{L}$ is the *labeling function*, and $v_0 \in V$ is the root of the tree. We denote the rightmost leaf of T by $rml(T)$. Whenever $T = (V, E, B, \mathcal{L}, L, v_0)$ is understood, we refer to V, E, B, L , respectively, as V_T, E_T, B_T and L_T throughout this paper.

A *pattern tree* on $\mathcal{L}$ (a *pattern*, for short) is a labeled ordered tree T on $\mathcal{L}$ whose node set is $V_T = \{1, \dots, k\}$ ($k \geq 0$) and all nodes are numbered consecutively by the preorder traversal [4] on T . Obviously, the root and the rightmost leaf of T are 1 and k , respectively. A k -*pattern* is a pattern of size exactly k . We assume the *empty tree* $\perp$ of size zero. For every $k \geq 0$, we denote by $\mathcal{T}, \mathcal{T}_k$, and $\mathcal{T}^k = \cup_{i \leq k} \mathcal{T}_i$ the classes of all patterns, all patterns of size exactly k , and all pattern of size at most k on $\mathcal{L}$, respectively.

Let $(\mathcal{D}, \xi)$ be a sample consisting of a database $\mathcal{D} = \{D_1, \dots, D_m\}$ of ordered trees on $\mathcal{L}$ and an objective attribute $\xi : \mathcal{D} \rightarrow \{0, 1\}$. Without loss of generality, we assume that V_{D_i} and V_{D_j} are disjoint if $i \neq j$. Then, a pattern tree $T \in \mathcal{P}$ *matches* a data tree $D \in \mathcal{D}$ if there exists some *order-preserving embedding* or a *matching function of T into D* , that is, any function $\varphi : V_T \rightarrow V_D$ that satisfies the following conditions (i)–(iv) for any $v, v_1, v_2 \in V_T$:

- (i) φ is a one-to-one mapping .
- (ii) φ preserves the parent relation, i.e., $(v_1, v_2) \in E_T$ iff $(\varphi(v_1), \varphi(v_2)) \in E_D$.
- (iii) φ preserves the (transitive closure of) the sibling relation, i.e., $(v_1, v_2) \in (B_T)^+$ iff $(\varphi(v_1), \varphi(v_2)) \in (B_D)^+$.
- (iv) φ preserves the labels, i.e., $L_T(v) = L_D(\varphi(v))$.

Algorithm OPTT

Input: An integer $k \geq 0$, a sample $(\mathcal{D}, \xi)$, and an impurity function ψ .

Output: All ψ -optimal patterns T of size at most k on $(\mathcal{D}, \xi)$.

Variable: A collection $BD \subseteq (\mathcal{T} \times (V_{\mathcal{D}})^*)$ of pairs of a pattern and its rightmost occurrences in $\mathcal{D}$, called *boundary set*, and a priority queue $R \subseteq \mathcal{T} \times \mathbf{R}$ of patterns with real weight.

1. $BD := \{ \langle \perp, RMO(\perp) \rangle \}$, where $RMO(\perp)$ is the preorder traversal of D .
2. While $BD \neq \emptyset$, do:
 - (a) $\langle T, RMO(T) \rangle := Pop(BD)$;
 - (b) Compute $eval := \Psi_{\mathcal{D}, \xi}(T)$ using $RMO(T)$ and ξ ; $R := R \cup \{ \langle T, eval \rangle \}$;
 - (c) Let (x, y) be the stamp point of T and $eval_{opt}$ be the smallest $eval$ value in R . Then, if $\min(\Phi(x, 0), \Phi(0, y)) > eval_{opt}$ then the next step and go to the beginning of the while-loop.
 - (d) For each $\langle S, RMO(S) \rangle \in \text{Expand-A-Tree}(T, RMO(T))$, do:
 - $Push(\langle S, RMO(S) \rangle, BD)$;
3. Return all optimal patterns $\langle T, eval \rangle$ in the priority queue R .

Fig. 3. An efficient algorithm for discovering the optimal pattern of bounded size, where search strategy is either breadth-first or depth-first depending on the choice of the boundary set BD

Then, we also say that T occurs in D . We assume that the empty tree $\perp$ matches to any tree at any node. Suppose that there exists some matching function φ of k -pattern T into a data tree $D \in \mathcal{D}$. Then, we define the *root occurrence* and the *rightmost leaf occurrence* of T in D w.r.t. φ by the node $Root(\varphi) = \varphi(1)$ and the node $Rmo(\varphi) = \varphi(k)$, respectively. We denote by $RMO_{\mathcal{D}}(T)$ the set of all rightmost leaf occurrences of T in trees of $\mathcal{D}$.

Example 1. In Fig. 2, we show examples of labeled ordered trees D and T on $\mathcal{L} = \{A, B\}$, where the node name is attached to the right corner of and a label is contained in a circle. A set of three arrows from T to D illustrates a matching function φ_1 of T to D . Then, there are two root-occurrences of T in D , namely 2 and 7, while there are three rightmost leaf occurrences 4, 6 and 10.

In a labeled ordered tree T , the *depth* of node v , denoted by $depth(v)$, is the length of the path, the number of nodes in it, from the root to v . For every $p \geq 0$, the p -th *parent* of node v , denoted by $\pi_T^p(v)$, is the unique ancestor u of v such that the length of the path from u to v is $p + 1$. Clearly, $\pi_T^0(v) = v$ itself.

3 Mining Algorithms

In this section, we present an efficient algorithm for solving the optimal pattern discovery problem for labeled ordered trees.

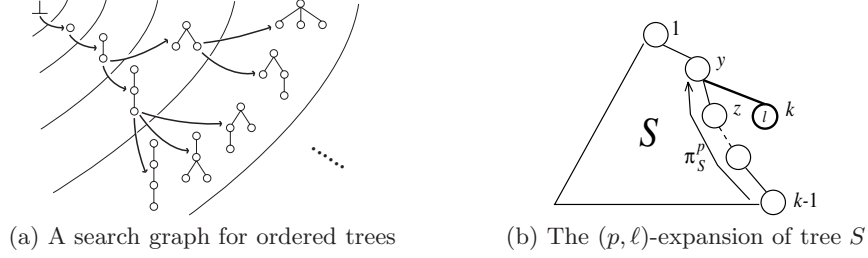


Fig. 4. The rightmost expansion for ordered trees

3.1 Overview of the Algorithm

Let us fix an impurity function ψ , and $k \geq 0$ be the maximum size of patterns. In Fig. 3, we present a mining algorithms **OPTT** for discovering all optimal patterns T of size at most k that minimize the cost function $\Psi_{\mathcal{D}, \xi}(T)$ in the sample $(\mathcal{D}, \xi)$ for the class of labeled ordered trees of bounded size k .

In Fig. 3, a *boundary set* is a collection BD of labeled ordered trees with the push operation $Push(BD, x)$ and the pop operation $Pop(BD)$. The algorithm **OPTT** maintains candidate patterns in the boundary set BD to search those labeled ordered trees appearing in database $\mathcal{D}$. The algorithm and its sub-procedures **Expand-A-Tree** (Fig. 5), **Update-RMO** (Fig. 6) also maintain for each candidate tree T , the list $RMO(T)$ of its rightmost occurrences in $\mathcal{D}$.

Starting with the boundary set BD containing only the empty pattern $\perp$, the algorithm **OPTT** searches the hypothesis space $\mathcal{T}^k = \cup_{0 \leq i \leq k} \mathcal{T}_i$ with growing candidate patterns in BD by attaching a new node one by one (Sec. 3.2). Whenever a successor $S \in \mathcal{T}$ is generated from a pattern T using the rightmost expansion, the algorithm incrementally computes the new occurrence list $RMO(S)$ of S from the old rightmost occurrence list $RMO(T)$ of T (Sec. 3.3). Repeating this process, the algorithm finally exits from the while loop and reports all optimal patterns with the smallest *eval* values in R .

3.2 Efficient Enumeration of Ordered Trees

In this subsection, we present an enumeration technique for generating all ordered trees of normal form without duplicates by incrementally expanding them from smaller to larger. This is a generalization of the itemset enumeration technique of [8], called the *set-enumeration tree*.

A *rightmost expansion* of a $(k-1)$ -pattern T is any k -pattern S obtained from T by attaching a new leaf x , namely $x = k$, with a label $\ell \in \mathcal{L}$ to a node y on the rightmost branch so that k is the rightmost child of y . Then, we say S is a *successor* of T and write $T \rightarrow S$. In the case that the attached node y is the p -th parent $\pi_T^p(x)$ of the rightmost leaf x of T and the label of y is $\ell \in \mathcal{L}$, then S is called the (p, ℓ) -*expansion* of T ((a) of Fig. 4). An *enumeration graph* on $\mathcal{T}$ is the graph $G = (\mathcal{T}, \rightarrow)$ with the node set $\mathcal{T}$ and the node set $\rightarrow$, the corresponding successor relation over $\mathcal{T}$.

Algorithm Expand-A-Tree($T, RMO(T)$)

```

 $\Gamma := \emptyset;$ 
For each pairs  $(p, \ell) \in \{0, \dots, \text{depth}(rml(T)) - 1\} \times \mathcal{L}$ , do:
  -  $S := \text{the } (p, \ell)\text{-expansion of } T;$   $RMO(S) := \text{Update-RMO}(RMO(T), p, \ell);$ 
  -  $\Gamma := \Gamma \cup \{S, RMO(S)\};$ 
Return  $\Gamma;$ 

```

Fig. 5. The algorithm for computing all successors of a pattern

Theorem 1 ([7]). *The enumeration graph $(\mathcal{T}, \rightarrow)$ is a tree with the unique root $\perp$, that is, a connected acyclic graph such that all nodes but the unique root $\perp$ have exactly one parent. This is true even if we restrict nodes to $\mathcal{T}^{(k)}$.*

Using the rightmost expansion technique, Expand-A-Tree of Fig. 5 enumerates all members of $\mathcal{T}$ without duplicates using an appropriate tree traversal method.

3.3 Updating Occurrence Lists

A key of our algorithm is how to efficiently store and update the information of a matching φ of each pattern T in $\mathcal{D}$. Instead of recording the full information $\varphi = \langle \varphi(1), \dots, \varphi(k) \rangle$, we record only the rightmost occurrences $Rmo(\varphi) = \varphi(k)$ as the partial information on φ . Based on this idea, our algorithm maintains the rightmost occurrence list $RMO(T)$ for each candidate pattern $T \in BD$.

Fig. 6 shows the algorithm Update-RMO that, given the (p, ℓ) -expansion T of a pattern S and the corresponding occurrence list $RMO(S)$, computes the occurrence list $RMO(T)$ without duplicates. This algorithm is based on the following observation: For every node y , y is in $RMO(T)$ iff there is a node x in $RMO(S)$ such that y is the strict younger sibling of the $(p-1)$ -th parent of x . Although a straightforward implementation of this idea still results in duplicates, the *Duplicate-Detection* technique [7] at Step 2(b) ensures the uniqueness of the elements of $RMO(T)$ (See [7], for detail).

Lemma 1 (Asai et al. [7]). *For a pattern S , the algorithm Update-RMO exactly computes all the elements in $RMO(T)$ from $RMO(S)$ without duplicates, where T is a rightmost expansion of S .*

3.4 Pruning by Convexity

Let N^T and N^F be the total numbers of positive and negative data trees in $\mathcal{D}$ and $N = N^T + N^F$. For a pattern $T \in \mathcal{T}$, a *stamp point* corresponding to T is a pair $(x, y) \in [0, N^T] \times [0, N^F]$ of integers, where $x = N_1^T$ and $y = N_1^F$ are the numbers of matched positive and negative data trees in $\mathcal{D}$. Recall that the goal is to minimize the cost function $\Psi_{S, \xi}(T)$ of Eq. 1 in Section 2.1. Since N^T and

Algorithm Update-RMO(RMO, p, ℓ)

-
1. Set RMO_{new} to be the empty list ε and $check := null$.
 2. For each element $x \in RMO$, do:
 - (a) If $p = 0$, let y be the leftmost child of x .
 - (b) Otherwise, $p \geq 1$. Then, do:
 - If $check = \pi_D^p(x)$ then skip x and go to the beginning of Step 2 (Duplicate-Detection).
 - Else, let y be the next sibling of $\pi_D^{p-1}(x)$ (the $(p-1)$ st parent of x in D) and set $check := \pi_D^p(x)$.
 - (c) While $y \neq null$, do the following:
 - If $L_D(y) = \ell$, then $RMO_{\text{new}} := RMO_{\text{new}} \cdot (y)$; /* Append */
 - $y := next(y)$; /* the next sibling */
 3. Return RMO_{new} .
-

Fig. 6. The incremental algorithm for updating the rightmost occurrence list of the (p, ℓ) -expansion of a given pattern T from that of T

N^F are constants for a fixed sample $(\mathcal{D}, \xi)$, we can regard $\Psi_{S, \xi}(T)$ as a function of a stamp point (x, y) and written as follows:

$$\Psi_{S, \xi}(T) = (x + y) \cdot \psi\left(\frac{x}{x + y}\right) + (N - (x + y)) \cdot \psi\left(\frac{N^T - x}{N - (x + y)}\right). \quad (2)$$

To emphasize this fact, we write $\Phi(x, y) \stackrel{\text{def}}{=} \Psi_{S, \xi}(T)$ as a function of (x, y) . Then, Morishita [21] showed that if $\psi(\theta)$ is an impurity function, then $\Phi(x, y)$ is *convex*, i.e., for every stamp points $\mathbf{x}_1, \mathbf{x}_2 \in [0, N^T] \times [0, N^F]$, $\Phi(\alpha \mathbf{x}_1 + (1 - \alpha) \mathbf{x}_2) \geq \alpha \Phi(\mathbf{x}_1) + (1 - \alpha) \Phi(\mathbf{x}_2)$ for any $0 \leq \alpha \leq 1$. This means that the stamp points with optimal values locates the edges of the 2-dimensional plain $[0, N^T] \times [0, N^F]$. Thus, we have the following theorem:

Theorem 2 (Morishita and Sese [21]). *Let T be any pattern and S be any pattern obtained from T by finite application of the rightmost expansion. Let (x, y) and (x', y') be the stamp points corresponding to T and S , respectively, w.r.t. $(\mathcal{D}, \xi)$. Then,*

$$\Phi(x', y') \geq \min(\Phi(x, 0), \Phi(0, y)) \quad (3)$$

From the above theorem, we incorporate the following pruning rule in the algorithm OPTT of Fig. 3 at Step 2(c).

Convexity Pruning Rule. During the computation of OPTT, for any pattern $T \in \mathcal{T}$ with the stamp point (x, y) , if $\min(\Phi(x, 0), \Phi(0, y))$ is strictly larger than the present optimal value of the patterns examined so far, then prune T and all of its successors.

4 Theoretical Analysis

4.1 The Case of Bounded Pattern Size

For a sample database $(\mathcal{D}, \xi)$, we introduce the parameters N , l , and b as the total number of nodes, the number of distinct labels, and the maximum branching factor of data trees in $\mathcal{D}$, respectively. In real databases such as collections of Web pages or XML data, we can often observe that l is not a constant but a slowly growing function $l(N)$ in N , while b is a constant.

In this setting, we can analyze the running time $T(N)$ of a straightforward generate-and-test algorithm for the optimized pattern discovery. Let $\mathcal{L}(\mathcal{D})$ be the set of labels in $\mathcal{D}$. Since there exists $\Theta(2^{ck}l(N)^k)$ distinct labeled ordered trees on $\mathcal{L}(\mathcal{D})$ for some c , if we assume $l(N) = O(N^\alpha)$ is a polynomial with degree $0 < \alpha < 1$ then the estimation of the running time is $T(N) = \Theta(2^{ck}N^{1+k\alpha})$, and thus not linear in N even if k and b are constants. In contrast, we show the following theorem on the time complexity of our algorithm OPTT, which is linear for constants k and b .

Theorem 3. *Under the above assumptions, the running time of OPTT on a sample $(\mathcal{D}, \xi)$ is bounded by $O(k^{k+1}b^kN)$.*

Proof. For the maximum pattern size K and every $0 \leq k \leq K$, let $\mathcal{C}_k$ be the set of all k -patterns and $R(k)$ be the total length of the rightmost occurrences (rmo) of the patterns in $\mathcal{C}_k$. We will estimate the upper bound of $R(k)$. First, we partition the patterns in $\mathcal{C}_k = \cup_p \mathcal{C}_{k,p}$ by the value of $0 \leq p < k$ when the pattern is generated by (p, ℓ) -expansion. Let $R(k, p)$ be the total length of the rmo of the patterns in $\mathcal{C}_{k,p}$. Then, we can show that $R(0, p) \leq N$ and $R(k, p) \leq bR(k-1)$ for any p . Since $R(k) \leq \sum_{p=0}^{k-1} bR(k-1, p)$, we have the recurrence $R(k) \leq kbR(k-1)$ for every $k \geq 0$. Solving this, we have $R(k) = O(k!b^k) = O(k^{k-1}b^kN)$. Since the running time of OPTT is bounded by $R = \sum_{k=1}^K kR(k) = O(K^{K+1}b^KN)$, the result immediately follows. $\square$

4.2 The Case of Unbounded Pattern Size

The maximum agreement problem is a dual problem of the classification error minimization problem and defined as follows: Given a pair (D, ξ) , find a pattern T that maximizes the *agreement* of T , i.e., the ratio of documents in S that is correctly classified by T .

Recently, Ben-David *et al.* [9] showed that for any $\varepsilon > 0$, there is no polynomial time $(770/767 - \varepsilon)$ -approximation algorithm for the maximum agreement problem for Boolean conjunctions if $P \neq NP$. When we can use arbitrary many labels, we can show the following theorem by using the approximation factor preserving reduction [25]. For the proof of Theorem 4, please consult the full paper [1]. The proof is not difficult, but we present the theorem here for it indicates the necessity of the bound on the maximum pattern size for efficient mining.

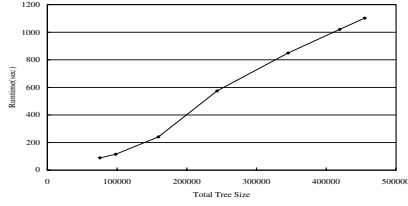


Fig. 7. The scalability: The running time with varying the input data size

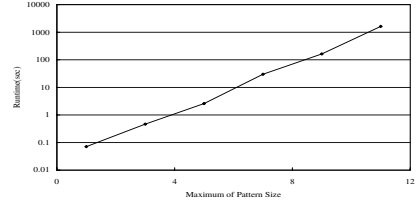


Fig. 8. The running time with varying the maximum pattern size

Table 1. Comparison of tree mining algorithms in running time and space

Algorithm	OPTT+DF	OPTT+DF+C	OPTT+BF	OPTT+BF+C	FREQT(0.1%)+BF
Time	29.7 (sec)	21.5 (sec)	20.2 (sec)	20.0 (sec)	10.4 (sec)
Space	8.0 (MB)	8.0 (MB)	96.4 (MB)	96.4 (MB)	20.7 (MB)

Theorem 4. For any $\varepsilon > 0$, there exists no polynomial time $(770/767 - \varepsilon)$ -approximation algorithm for the maximum agreement problem for labeled ordered trees of unbounded size on an unbounded label alphabet if $P \neq NP$. This is true even when either the maximum depth of trees is at most three or the maximum branching of trees is at most two.

5 Experimental Results

We run experiments on the following two data sets *Citeseers* and *Imdb*. *Citeseers* consists of CGI generated HTML pages from an Web site², and *Imdb* is a collection of movie entries in XML obtained and hand-transformed from an online movie database³. Both data contains several hundred thousands of nodes and several thousands of unique tags. We implemented several versions of the optimized tree miner OPTT in Java (SUN JDK1.3.1 JIT) using a DOM library (OpenXML). In the experiments, the suffix BF, DF, and C following OPTT designate the versions with the breadth-first, the depth-first, and the convex pruning. All experiments were run on PC (Pentium III 600MHz, 512 MB, Linux 2.2.14).

Scalability and Running Time

Fig. 7 shows the running time with a constant maximum pattern size $k = 5$ with varying the size of the data tree from 316 KB (22,847 nodes) to 5.61 MB (402,740 nodes) on *Citeseers*. The running time seems to linearly scale on this data set for fixed k and this fits to the theoretical bound of Theorem 3.

² Research Index, <http://citeseer.nj.nec.com/>

³ Internet Movie Database, <http://www.imdb.com/>

```

% Optimal: All action movies and some family movie have genre "action"
No. 2, Size 3, Gini 0.125, X 15/15 (Action), Y 1/15 (Family):
<MOVIE> <GENRE> ACTION </GENRE> </MOVIE>

% Optimal: Most action movie has been rated as no-one-under-15 at a country
No. 4, Size 4, Gini 0.333, X 12/15 (Action), Y 0/15 (Family):
<MOVIE> <CERTIFICATION> <CERTIF> 15 </CERTIF> </CERTIFICATION> </MOVIE>

% Frequent: Any movie is directed by someone
No. 4, Size 3, Freq 1.00, X 15/15 (Action), Y 15/15 (Family):
<MOVIE> <DIRECTED_BY> <PERSON> </PERSON> </DIRECTED_BY> </MOVIE>

```

Fig. 9. Examples of discovered optimal patterns

Fig. 8 shows the running time on a fixed dataset a subset of *Imdb* of size 40 KB (5835 nodes) with varying the maximum pattern tree size k from 1 to 11. Since the y-axis is log-scaled, this plot indicates that when the data size is fixed, the running time is exponential in the maximum pattern size k .

Search Strategies and Pruning Techniques

Table. 1 shows the running time of optimized tree miners OPTT+DF, OPTT+DF+C, OPTT+BF, OPTT+BF+C, and a frequent tree miner FREQT on *Imdb* data of size 40 KB. This experiment shows that on this data set, OPTT+DF saves the main memory size more than ten times than OPTT+BF, while the difference in the running time between them is not significant. Also, the use of pruning with convexity (denoted by C) in Section 3.4 is effective in the depth-first search; OPTT+DF+C is 1.5 times faster than OPTT+DF.

Examples of Discovered Patterns. In Fig. 9, we show examples of optimal patterns in XML format discovered by the OPTT algorithm by optimizing the Gini index ψ on a collection of XML entries for 15 action movies and 15 family movies from the *Imdb* dataset. Total size is 1 MB and it contains over two hundred thousands nodes. At the header, the line “No. 4, Size 4, Gini 0.333, X 12/15 (Action), Y 0/15 (Family)” means that the pattern is the 4th best pattern with Gini index 0.333 and that appears in 12/15 of action movies and 0/15 of family movies. The first optimal pattern is rather trivial and says that “*All action movies and some family movie have genre action*” and the second optimal pattern says that “*Most action movie has been rated as no-one-under-15 in at least one country.*” For comparison, we also show a frequent but trivial pattern saying that “*Any movie is directed by someone.*”

6 Conclusion

In the context of semi-structured data mining, we presented an efficient mining algorithm that discovers all labeled ordered trees that optimize a given statistical objective function on a large collection of labeled ordered trees. Theoretical analyses show that the algorithm works efficiently for patterns of bounded size. Experimental results also confirmed the scalability of the algorithm and the effectiveness of the search strategy and the pruning technique with convexity.

Acknowledgments

The authors would like to thank Akihiro Yamamoto, Masayuki Takeda, Ayumi Shinohara, Daisuke Ikeda and Akira Ishino for fruitful discussion on Web and text mining. We are also grateful to Shinichi Morishita, Masaru Kitsuregawa, Takeshi Tokuyama, and Mohammed Zaki for their valuable comments.

References

1. K. Abe, S. Kawasoe, T. Asai, H. Arimura, S. Arikawa, Optimized substructure discovery for semi-structured data, DOI, Kyushu Univ., DOI-TR-206, Mar. 2002. <ftp://ftp.i.kyushu-u.ac.jp/pub/tr/trcs206.ps.gz> 10
2. S. Abiteboul, P. Buneman, and D. Suciu. *Data on the Web*. Morgan Kaufmann, 2000. 1, 3, 5
3. R. Agrawal, R. Srikant, Fast algorithms for mining association rules, In *Proc. VLDB'94*, 487–499, 1994. 2
4. A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *Data Structures and Algorithms*. Addison-Wesley, 1983. 5
5. H. Arimura, A. Wataki, R. Fujino, S. Arikawa, An efficient algorithm for text data mining with optimal string patterns, In *Proc. ALT'98*, LNAI 1501, 247–261, 1998. 1, 4
6. H. Arimura, J. Abe, R. Fujino, H. Sakamoto, S. Shimozone, S. Arikawa, Text Data Mining: Discovery of Important Keywords in the Cyberspace, In *Proc. IEEE Kyoto Int'l Conf. on Digital Libraries*, 2000. 2
7. T. Asai, K. Abe, S. Kawasoe, H. Arimura, H. Sakamoto, and S. Arikawa. Efficient substructure discovery from large semi-structured data. In *Proc. the 2nd SIAM Int'l Conf. on Data Mining (SDM2002)*, 158–174, 2002. 2, 3, 5, 8
8. R. J. Bayardo Jr. Efficiently mining long patterns from databases. In *Proc. SIGMOD98*, 85–93, 1998. 2, 7
9. S. Ben-David, N. Eiron, and P. M. Long, On the difficulty of Approximately Maximizing Agreements, In *Proc. COLT 2000*, 266–274, 2000. 10
10. L. Dehaspe, H. Toivonen, and R. D. King. Finding frequent substructures in chemical compounds. In *Proc. KDD-98*, 30–36, 1998. 1, 3
11. L. Devroye, L. Györfi, G. Lugosi, *A Probabilistic Theory of Pattern Recognition*, Springer-Verlag, 1996. 1, 2, 4
12. R. Fujino, H. Arimura, S. Arikawa, Discovering unordered and ordered phrase association patterns for text mining. In *Proc. PAKDD2000*, LNAI 1805, 2000.

13. T. Fukuda, Y. Morimoto, S. Morishita, and T. Tokuyama. Data mining using two-dimensional optimized association rules. In *Proc. SIGMOD'96*, 13–23, 1996. [1](#), [4](#)
14. R. C. Holte, Very simple classification rules perform well on most commonly used datasets, *Machine Learning*, 11, 63–91, 1993. [1](#), [4](#)
15. A. Inokuchi, T. Washio and H. Motoda. An Apriori-Based Algorithm for Mining Frequent Substructures from Graph Data, In *Proc. PKDD 2000*, 13–23, 2000. [2](#), [3](#)
16. M. J. Kearns, R. E. Shapire, L. M. Sellie, Toward efficient agnostic learning. *Machine Learning*, 17(2–3), 115–141, 1994. [1](#), [4](#)
17. W. Maass, Efficient agnostic PAC-learning with simple hypothesis, In *Proc. COLT94*, 67–75, 1994. [1](#), [4](#)
18. T. Matsuda, T. Horiuchi, H. Motoda, T. Washio, *et al.*, Graph-based induction for general graph structured data. In *Proc. DS'99*, 340–342, 1999. [1](#), [3](#)
19. T. Miyahara, T. Shoudai, T. Uchida, K. Takahashi, and H. Ueda. Discovery of frequent tree structured patterns in semistructured web documents. In *Proc. PAKDD-2001*, 47–52, 2001. [1](#), [2](#), [3](#)
20. S. Morishita, On classification and regression, In *Proc. Discovery Science '98*, LNAI 1532, 49–59, 1998. [1](#), [4](#)
21. S. Morishita and J. Sese, Traversing Itemset Lattices with Statistical Metric Pruning, In *Proc. PODS'00*, 226–236, 2000. [3](#), [9](#)
22. J. R. Quinlan, *C4.5: Program for Machine Learning*, Morgan Kaufmann Publishers, San Mateo, CA, 1993. [2](#), [4](#)
23. R. Rastogi, K. Shim, Mining Optimized Association Rules with Categorical and Numeric Attributes, In *Proc. ICDE'98*, 503–512, 1998. [1](#), [4](#)
24. H. Arimura, S. Arikawa, S. Shimozone, Efficient discovery of optimal word-association patterns in large text databases *New Gener. Comput.*, 18, 49–60, 2000. [4](#)
25. V. V. Vazirani, *Approximation Algorithms*, Springer, Berlin, 1998. [10](#)
26. W3C Recommendation. *Extensible Markup Language (XML) 1.0*, second edition, 06 October 2000. <http://www.w3.org/TR/REC-xml>. [1](#), [3](#)
27. K. Wang and H. Q. Liu. Discovering structural association of semistructured data. *IEEE Trans. Knowledge and Data Engineering (TKDE2000)*, 12(3):353–371, 2000. [1](#), [2](#), [3](#)
28. M. J. Zaki. Efficiently mining frequent trees in a forest. Computer Science Department, Rensselaer Polytechnic Institute, *PRI-TR01-7-2001*, 2001. <http://www.cs.rpi.edu/~zaki/PS/TR01-7.ps.gz> [2](#), [3](#)

Fast Outlier Detection in High Dimensional Spaces

Fabrizio Angiulli and Clara Pizzuti

ISI-CNR, c/o DEIS, Università della Calabria
87036 Rende (CS), Italy
{angiulli,pizzuti}@isi.cs.cnr.it

Abstract. In this paper we propose a new definition of distance-based outlier that considers for each point the sum of the distances from its k nearest neighbors, called weight. Outliers are those points having the largest values of weight. In order to compute these weights, we find the k nearest neighbors of each point in a fast and efficient way by linearizing the search space through the Hilbert space filling curve. The algorithm consists of two phases, the first provides an approximated solution, within a small factor, after executing at most $d + 1$ scans of the data set with a low time complexity cost, where d is the number of dimensions of the data set. During each scan the number of points candidate to belong to the solution set is sensibly reduced. The second phase returns the exact solution by doing a single scan which examines further a little fraction of the data set. Experimental results show that the algorithm always finds the exact solution during the first phase after $\bar{d} \ll d + 1$ steps and it scales linearly both in the dimensionality and the size of the data set.

1 Introduction

Outlier detection is an outstanding data mining task referred to as *outlier mining* that has a lot of practical applications such as telecom or credit card frauds, medical analysis, pharmaceutical research, financial applications. Outlier mining can be defined as follows: "Given a set of N data points or objects, and n , the expected number of outliers, find the top n objects that are considerably dissimilar with respect to the remaining data" [9]. Many data mining algorithms consider outliers as noise that must be eliminated because it degrades their predictive accuracy. For example, in classification algorithms mislabelled instances are considered outliers and thus they are removed from the training set to improve the accuracy of the resulting classifier [6]. However, as pointed out in [9], "one person's noise could be another person's signal", thus outliers themselves can be of great interest. The approaches to outlier mining can be classified in supervised-learning based methods, where each example must be labelled as exceptional or not, and the unsupervised-learning based ones, where the label is not required. The latter approach is more general because in real situations we do not have such information. Unsupervised-learning based methods for outlier detection can be categorized in several approaches. The first is *statistical-based*

and assumes that the given data set has a distribution model. Outliers are those points that satisfies a discordancy test, that is that are significantly larger (or smaller) in relation to the hypothesized distribution [4]. In [20] a Gaussian mixture model to represent the normal behaviors is used and each datum is given a score on the basis of changes in the model. High score indicates high possibility of being an outlier. This approach has been combined in [19] with a supervised-learning based approach to obtain general patterns for outliers. *Deviation-based* techniques identify outliers by inspecting the characteristics of objects and consider an object that deviates from these features an outlier [3,16]. A completely different approach that finds outliers by observing *low dimensional projections* of the search space is presented in [1]. Yu et al. [7] introduced *FindOut*, a method based on wavelet transform, that identifies outliers by removing clusters from the original data set. Wavelet transform has also been used in [18] to detect outliers in stochastic processes. Another category is the *density-based*, presented in [5] where a new notion of local outlier is introduced that measures the degree of an object to be an outlier with respect to the density of the local neighborhood. This degree is called *Local Outlier Factor LOF* and is assigned to each object. The computation of LOFs, however, is expensive and it must be done for each object. To reduce the computational load, Jin et al. in [10] proposed a new method to determine only the top- n local outliers that avoids the computation of LOFs for most objects if $n \ll N$, where N is the data set size. *Distance-based* outlier detection has been introduced by Knorr and Ng [12] to overcome the limitations of statistical methods. A *distance-based* outlier is defined as follows: *A point p in a data set is an outlier with respect to parameters k and δ if no more than k points in the data set are at a distance of δ or less from p .* This definition of outlier has a number of benefits but, as observed in [14], it depends on the two parameters k and δ and it does not provide a ranking of the outliers. Furthermore the two algorithms proposed are either quadratic in the data set size or exponential in the number of dimensions, thus their experiments cannot go beyond five dimensions. In the work [14] the definition of outlier is modified to address these drawbacks and it is based on the distance of the k -th nearest neighbor of a point p , denoted with $D^k(p)$. The new definition of outlier is the following: *Given a k and n , a point p is an outlier if no more than $n-1$ other points in the data set have a higher value for D^k than p .* This means that the top n points having the maximum D^k values are considered outliers. The experiments presented, up to 10 dimensions, show that their method scales well. This definition is interesting but does not take into account the local density of points. The authors note that "points with large values for $D^k(p)$ have more sparse neighborhoods and are thus typically stronger outliers than points belonging to dense clusters which will tend to have lower values of $D^k(p)$." However, consider Figure 1. If we set $k = 10$, $D^k(p_1) = D^k(p_2)$, but we can not state that p_1 and p_2 can be considered being outliers at the same way.

In this paper we propose a new definition of outlier that is distance-based but that considers for each point p the sum of the distances from its k nearest neighbors. This sum is called the *weight* of p , $\omega_k(p)$, and it is used to rank

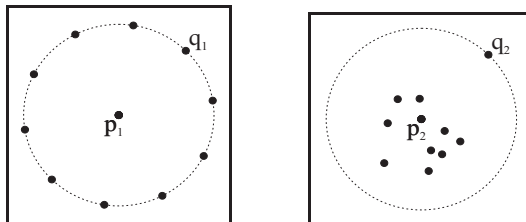


Fig. 1. Two points with same D^k values ($k=10$)

the points of the data set. Outliers are those points having the larger values of ω_k . In order to compute these weights, we find the k nearest neighbors of each point in a fast and efficient way by linearizing the search space. We fit the d -dimensional data set $\mathbf{DB}$ in the hypercube $D = [0, 1]^d$, then we map D into the interval $I = [0, 1]$ by using the *Hilbert space filling curve* and obtain the k nearest neighbors of each point by examining its predecessors and successors on I . The mapping assures that if two points are close in I , they are close in D too, although the reverse is not always true. To limit the loss of nearness, the data set is shifted $d + 1$ times along the main diagonal of the hypercube $[0, 2]^d$. The algorithm consists of two phases, the first provides an approximated solution, within a small factor, after executing at most $d + 1$ scans of the data set with a low time complexity cost. During each scan a better lower bound for the weight of the k -th outlier of $\mathbf{DB}$ is obtained and the number of points candidate to belong to the solution set is sensibly reduced. The second returns the exact solution by doing a single scan which examines further a little fraction of the data set. However, as experimental results show, we always find the exact solution during the first phase after $\bar{d} \ll d + 1$ steps.

It is worth to note that approaches based on wavelet transform apply this multi-resolution signal processing technique to transform the original space in a new one of the same dimension and find outliers in the transformed space at different levels of approximation. In our approach, however, space filling curves are used to map a multidimensional space in a one dimensional space to obtain the nearest neighbors of each point in a fast way, but the distance computation is done in the original space.

The paper is organized as follows. Section 2 gives definitions and properties necessary to introduce the algorithm and an overview of space filling curves. Section 3 presents the method. In Section 4, finally, experimental results on several data sets are reported.

2 Definitions and Notations

In this section we present the new definition of outlier and we introduce the notions that are necessary to describe our algorithm. The L_t distance between two points $p = (p_1, \dots, p_d)$ and $q = (q_1, \dots, q_d)$ is defined as $d_t(p, q) = (\sum_{i=1}^d |p_i - q_i|^t)^{1/t}$ for $1 \leq t < \infty$, and $\max_{1 \leq i \leq d} |p_i - q_i|$ for $t = \infty$.

Let $\mathbf{DB}$ be a d -dimensional data set, k a parameter and let p be a point of $\mathbf{DB}$. Then the *weight* of p in $\mathbf{DB}$ is defined as $\omega_k(p) = \sum_{i=1}^k d_t(p, nn_i(p))$, where $nn_i(p)$ denotes the i -th nearest neighborhood of p in $\mathbf{DB}$.

Given a data set $\mathbf{DB}$, parameters k and n , a point $p \in \mathbf{DB}$ is the n -th outlier with respect to k , denoted as $outlier_k^n$, if there are exactly $n - 1$ points q in $\mathbf{DB}$ such that $\omega_k(q) > \omega_k(p)$.

Given a data set $\mathbf{DB}$, parameters k and n , we denote with Out_k^n the set of the top n outliers of $\mathbf{DB}$ with respect to k . Let Out^* be a set of n points of $\mathbf{DB}$ and ϵ a positive real number, we say that Out^* is an ϵ -approximation of Out_k^n if $\omega^* \epsilon \geq \omega^n$, where ω^* is $\min\{\omega_k(p) \mid p \in Out^*\}$ and ω^n is the weight of $outlier_k^n$.

Points in $\mathbf{DB}$ are thus ordered according to their weights $\omega_k(p)$, computed by using any L_t metrics. The n points Out_k^n having the maximum ω_k values are considered outliers. To compute the weights, the k nearest neighbors are obtained by using *space-filling curves*. The concept of *space-filling curve* came out in the 19-th century and it is accredited to Peano [15] who, in 1890, proved the existence of a continuous mapping from the interval $I = [0, 1]$ onto the square $Q = [0, 1]^2$. Hilbert in 1891 defined a general procedure to generate an entire class of space-filling curves. He observed that if the interval I can be mapped continuously onto the square Q then, after partitioning I into four congruent subintervals and Q into four congruent sub-squares, each subinterval can be mapped onto one of the sub-squares. Sub-squares are ordered such that each pair of consecutive sub-squares share a common edge. If this process is continued ad infinitum, I and Q are partitioned into 2^{2h} replicas for $h = 1, 2, 3 \dots$. In practical applications the partitioning process is terminated after h steps to give an approximation of a space-filling curve of order h . For $h \geq 1$ and $d \geq 2$, let $\mathcal{H}_h^d$ denote the h -th order approximation of a d -dimensional Hilbert space-filling curve that maps 2^{hd} subintervals of length $1/2^{hd}$ into 2^{hd} sub-hypercubes whose centre-points are considered as points in a space of finite granularity. The Hilbert curve, thus, passes through every point in a d -dimensional space once and once only in a particular order. This establishes a mapping between values in the interval I and the coordinates of d -dimensional points. Let D be the set $\{p \in \mathbb{R}^d : 0 \leq p_i \leq 1, 1 \leq i \leq d\}$ and p a d -dimensional point in D . The inverse image of p under this mapping is called its *Hilbert value* and is denoted by $\mathcal{H}(p)$. Let $\mathbf{DB}$ be a set of points in D . These points can be sorted according to the order in which the curve passes through them. We denote by $\mathcal{H}(\mathbf{DB})$ the set $\{\mathcal{H}(p) \mid p \in \mathbf{DB}\}$ sorted with respect to the order relation induced by the Hilbert curve. Given a point p the predecessor and the successor of p , denoted $\mathcal{H}_{pred}(p)$ and $\mathcal{H}_{succ}(p)$, in $\mathcal{H}(\mathbf{DB})$ are thus the two closest points with respect to the ordering induced by the Hilbert curve. The m -th predecessor and successor of p are denoted by $\mathcal{H}_{pred}(p, m)$ and $\mathcal{H}_{succ}(p, m)$. Space filling curves have been studied and used in several fields [8, 11, 17]. A useful property of such a mapping is that if two points from the unit interval I are close then the corresponding images are close too in the hypercube D . The reverse statement, however, is not true because two close points in D can have non-close inverse images in I . This implies that the reduction of dimensionality from d to one can provoke the loss of the property

of nearness. In order to preserve the closeness property, approaches based on the translation and/or rotation of the hypercube D have been proposed [13,17]. Such approaches assure the maintenance of the closeness of two d -dimensional points, within some factor, when they are transformed into one dimensional points. In particular, in [13], the number of shifts depends on the dimension d . Given a data set $\mathbf{DB}$ and the vector $v^{(j)} = (j/(d+1), \dots, j/(d+1)) \in \mathbb{R}^d$, each point $p \in \mathbf{DB}$ can be translated $d+1$ times along the main diagonal in the following way: $p^j = p + v^{(j)}$, for $j = 0, \dots, d$. The shifted copies of points thus belong to $[0, 2]^d$ and, for each p , $d+1$ Hilbert values in the interval $[0, 2]$ can be computed. In this paper we make use of this family of shifts to overcome the loss of the nearness property.

An r -region is an open ended hypercube in $[0, 2)^d$ with side length $r = 2^{1-l}$ having the form $\prod_{i=0}^{d-1} [a_i r, (a_i + 1)r)$, where each a_i , $0 \leq i < d$, and l are in $\mathbb{N}$. The *order* of an r -region of side r is the quantity $-\log_2 r$.

Let p and q be two points. We denote by $MinReg(p, q)$ the side of smallest r -region containing both p and q . We denote by $MaxReg(p, q)$ the side of the greatest r -region containing p but not q .

Let p be a point, and let r be the side of an r -region. Then

$$MinDist(p, r) = \min_{i=1}^d \{\min\{p_i \bmod r, r - p_i \bmod r\}\}$$

$$MaxDist(p, r) = \begin{cases} (\sum_{i=1}^d (\max\{p_i \bmod r, r - p_i \bmod r\})^t)^{1/t} & \text{for } 1 \leq t < \infty \\ \max_{i=1}^d \{\max\{p_i \bmod r, r - p_i \bmod r\}\} & \text{for } t = \infty \end{cases}$$

where $x \bmod r = x - \lfloor x/r \rfloor r$, and p_i denotes the value of p along the i -th coordinate, are respectively the perpendicular distance from p to the nearest face of the r -region of side r containing p , i.e. a lower bound for the distance between p and a point lying out of the above r -region, and the distance from p to the furthest vertex of the r -region of side r containing p , i.e. an upper bound for the distance between p and a point lying into the above r -region.

Let p be a point in $\mathbb{R}^d$, and let r be a non negative real. Then the d -dimensional *neighborhood* of p (under the L_t metric) of radius r , written $\mathcal{B}(p, r)$, is the set $\{q \in \mathbb{R}^d \mid d_t(p, q) \leq r\}$.

Let p , q_1 , and q_2 be three points. Then

$$BoxRadius(p, q_1, q_2) = MinDist(p, \min\{MaxReg(p, q_1), MaxReg(p, q_2)\})$$

is the radius of the greatest neighborhood of p entirely contained in the greatest r -region containing p but neither q_1 nor q_2 .

Lemma 1. *Given a data set $\mathbf{DB}$, a point p of $\mathbf{DB}$, two positive integers a and b , and the set of points*

$$I = \{\mathcal{H}_{pred}(p, a), \dots, \mathcal{H}_{pred}(p, 1), \mathcal{H}_{succ}(p, 1), \dots, \mathcal{H}_{succ}(p, b)\}$$

let r be $BoxRadius(p, \mathcal{H}_{pred}(p, a-1), \mathcal{H}_{succ}(p, b+1))$ and $S = I \cap \mathcal{B}(p, r)$. Then

1. The points in S are the true first $|S|$ nearest-neighbors of p in **DB**;
2. $d_t(p, nn_{|S|+1}(p)) > r$.

The above Lemma allows us to determine, among the $a + b$ points, nearest neighbors of p with respect to the Hilbert order (thus they constitute an approximation of the true closest neighbors), the exact $|S| \leq a + b$ nearest neighbors of p and to establish a lower bound to the distance from p to the $(|S| + 1)$ -th nearest neighbor. This result is used in the algorithm to estimate a lower bound to the weight of any point p .

3 Algorithm

In this section we give the description of the *HilOut* algorithm. The method consists of two phases, the first does at most $d + 1$ scans of the input data set and guarantees a solution that is an $k\epsilon_d$ -approximation of Out_k^n , where $\epsilon_d = 2d^{\frac{1}{d}}(2d + 1)$ (for a proof of this statement we refer to [2]), with a low time complexity cost. The second phase does a single scan of the data set and computes the set Out_k^n . At each scan *HilOut* computes a lower bound and an upper bound to the weight ω_k of each point and it maintains the n greatest lower bound values in the heap *WLB*. The n -th value ω^* in *WLB* is a lower bound to the weight of the n -th outlier and it is used to detect those points that can be considered candidate outliers. The upper and lower bound of each point are computed by exploring a neighborhood of the point on the interval I . The neighborhood of each point is initially set to $2k$, then it is widened, proportionally to the number of remaining candidate outliers, to obtain a better estimate of the true k nearest neighbors. At each iteration, as experimental results show, the number of candidate outliers sensibly diminishes. This allows the algorithm to find the exact solution in few steps, in practice after $\bar{d}$ steps with $\bar{d} \ll d + 1$. Before starting with the description, we introduce the concept of *point feature*. A *point feature* f is a 7-tuple $\langle point, hilbert, level, weight, weight_0, radius, count \rangle$ where *point* is a point in $[0, 2)^d$, *hilbert* is the Hilbert value associated to *point* in the h -th order approximation of the d -dimensional Hilbert space-filling curve mapping the hypercube $[0, 2)^d$ into the integer set $[0, 2^{hd})$, *level* is the order of the smallest r -region containing both *point* and its successor in **DB** (with respect to the Hilbert order), *weight* is an upper bound to the weight of *point* in **DB**, *radius* is the radius of a d -dimensional neighborhood of *point*, *weight₀* is the sum of the distances between *point* and each point of **DB** lying in the d -dimensional neighborhood of radius *radius* of *point*, while *count* is the number of these points.

In the following with the notation $f.point$, $f.hilbert$, $f.level$, $f.weight$, $f.weight_0$, $f.radius$ and $f.count$ we refer to the *point*, *hilbert*, *level*, *type*, *weight*, *weight₀*, *radius*, and *count* value of the point feature f respectively. Let f be a point feature, we denote by $wlb(f)$ the value $f.weight_0 + (k - f.count) \times f.radius$. $wlb(f)$ is a lower bound to the weight of $f.point$ in **DB**. The algorithm, reported in Figure 2, receives as input a data set **DB** of N points in the hypercube $[0, 1]^d$,

the number n of top outliers to find and the number k of neighbors to consider. The data structures employed are the two heaps of n point features OUT and WLB , the set TOP , and the list of point features PF . At the end of each iteration, the features stored in OUT are those with the n greatest values of the field $weight$, while the features f stored in WLB are those with the n greatest values of $wlb(f)$. TOP is a set of at most $2n$ point features which is set to the union of the features stored in OUT and WLB at the end of the previous iteration. PF is a list of point features. In the following, with the notation PF_i we mean the i -th element of the list PF .

First, the algorithm builds the list PF associated to the input data set, i.e. for each point p of **DB** a point feature f with $f.point = p$, $f.weight = \infty$, and the other fields set to 0, is inserted in PF , and initializes the set TOP and the global variables ω^* , N^* , and n^* . ω^* is a lower bound to the weight of the $outlier_k^n$ in **DB**. This value, initially set to 0, is then updated in the procedure *Scan*. N^* is the number of point features f of PF such that $f.weight \geq \omega^*$. The points whose point feature satisfies the above relation are called *candidate outliers* because the upper bound to their weight is greater than the current lower bound ω^* . This value is updated in the procedure *Hilbert*. n^* is the number of true outliers in the heap OUT . It is updated in the procedure *TrueOutliers* and it is equal to $|\{f \in OUT \mid wlb(f) = f.weight \wedge f.weight \geq \omega^*\}|$. The main cycle, consists of at most $d+1$ steps. We explain the single operations performed during each step of this cycle.

Hilbert. The *Hilbert* procedure calculates the value $\mathcal{H}(PF_i.point + v^{(j)})$ of each point feature PF_i of PF , places this value in $PF_i.hilbert$, and sorts the point features in the list PF using as order key the values $PF_i.hilbert$. After sorting, the procedure *Hilbert* updates the value of the field $level$ of each point feature. In particular, the value $PF_i.level$ is set to the order of the smallest r -region containing both $PF_i.point$ and $PF_{i+1}.point$, i.e. to $MinReg(PF_i.point, PF_{i+1}.point)$, for each $i = 1, \dots, N-1$. For example, consider figure 3 where seven points in the square $[0, 1]^2$ are consecutively labelled with respect to the Hilbert order. Figure 3 (b) highlights the smallest r -region containing the two points 5 and 6 while Figure 3 (c) that containing the two points 2 and 3. The values of the levels associated with the points 5 and 2 are thus three and one because the order of corresponding r -regions are $-\log_2 2^{1-4} = 3$ and $-\log_2 2^{1-2} = 1$ respectively. On the contrary, the smallest r -region containing points 1 and 2 is all the square.

Scan. The procedure *Scan* is reported in Figure 2. This procedure performs a sequential scan of the list PF by considering only those features that have a weight upper bound not less than ω^* , the lower bound to the weight of $outlier_k^n$ of **DB**. These features are those candidate to be outliers, the others are simply skipped. If the value $PF_i.count$ is equal to k then $F_i.weight$ is the true weight of $PF_i.point$ in **DB**. Otherwise $PF_i.weight$ is an upper bound for the value $\omega_k(PF_i.point)$ and it could be improved. For this purpose the function *FastUpperBound* calculates a novel upper bound ω to the weight of $PF_i.point$, given by $k \times MaxDist(PF_i.point, 2^{-level})$, by examining k points among its successors and predecessors to find $level$, the order of the smallest r -region con-

<pre> <i>HilOut</i> (DB, n, k) { Initialize(PF, DB); /* First Phase */ $TOP = \emptyset$; $N^* = N$; $n^* = 0$; $\omega^* = 0$; $j = 0$; while ($j \leq d$ && $n^* < n$) { Initialize(OUT); Initialize(WLB); Hilbert($v^{(j)}$); Scan($v^{(j)}$, $\frac{kN}{N^*}$); TrueOutliers(OUT); $TOP = OUT \cup WLB$; $j = j + 1$; } /* Second Phase */ if ($n^* < n$) Scan($v^{(d)}$, N); return OUT; } </pre>	<pre> Scan(v, k_0) { for ($i = 1$; $i \leq N$; $i++$) if ($PF_i.weight \geq \omega^*$) { if ($PF_i.count < k$) { $\omega = FastUpperBound(i)$; if ($\omega < \omega^*$) $PF_i.weight = \omega$ else { $maxc = \min(2k_0, N)$; if ($PF_i \in TOP$) $maxc = N$; InnerScan(i, $maxc$, v, NN); if ($NN.radius > PF_i.radius$) { $PF_i.radius = NN.radius$; $PF_i.weight_0 = NN.weight_0$; $PF_i.count = NN.count$; } if ($NN.weight < PF_i.weight$) $PF_i.weight = NN.weight$; } } } Update(OUT, PF_i); Update(WLB, $wlb(PF_i)$); $\omega^* = Max(\omega^*, Min(WLB))$; } </pre>

Fig. 2. The algorithm *HilOut* and the procedure *Scan*

taining both $PF_i.point$ and other k neighbors. If ω is less than ω^* , no further elaboration is required. Otherwise the procedure *InnerScan* returns the data structure NN which has the fields $NN.weight$, $NN.weight_0$, $NN.radius$ and $NN.count$. If $NN.radius$ is greater than $PF_i.radius$ then a better lower bound for the weight of $PF_i.point$ is available, and the fields $radius$, $weight_0$, and $count$ of PF_i are updated. Same considerations hold for the value $PF_i.weight$. Finally, the heaps WLB and OUT process $wlb(PF_i)$ and PF_i respectively, and the value ω^* is updated.

InnerScan. This procedure takes into account the points whose Hilbert value lies in a one dimensional neighborhood of the integer value $PF_i.hilbert$. In particular, if PF_i belongs to TOP , then the size of the above neighborhood, stored in $maxc$ is at most N , otherwise this size is at most $2kN/N^*$, i.e. it is inversely proportional to the number N^* of candidate outliers. This procedure manages a data structure NN constituted by a heap of k real numbers and the fields $NN.weight$, $NN.weight_0$, $NN.count$, and $NN.radius$. At the end of *InnerScan*, NN contains the k smallest distances between the point $PF_i.point$ and the points of the above defined one dimensional neighborhood, $NN.radius$ is the radius of the d -dimensional neighborhood of $PF_i.point$ explored when considering these points, calculated as in Lemma 1, $NN.weight$ is the sum of the elements stored in the heap of NN , $NN.weight_0$ is the sum of the elements stored in the heap of NN which are less than or equal to $NN.radius$, while $NN.count$ is their number. Thus *InnerScan* returns a new upper bound and a new lower bound

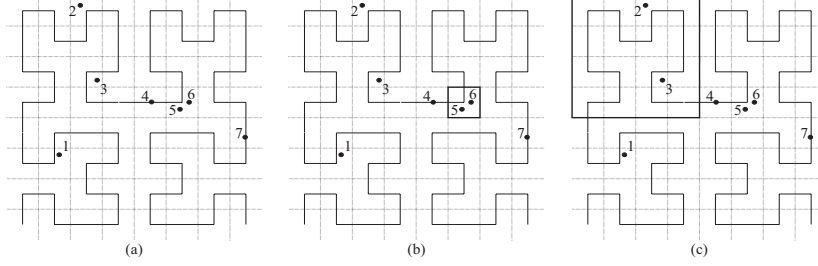


Fig. 3. The *level* field semantics

for the weight of $PF_i.point$. We note that the field *level* of each point feature is exploited by *InnerScan* to determine in a fast way if the exact k nearest neighbors of such point have already been encountered (see [2] for a detailed description). The main cycle of the algorithm stops when $n^* = n$, i.e. when the heap *OUT* is equal to the set of top n outliers, or after $d + 1$ iterations. At the end of the first phase, the heap *OUT* contains a $k\epsilon_d$ -approximation of Out_k^n . Finally, if $n^* < n$, that is if the number of true outliers found by the algorithm is not n , then a final scan computes the exact solution. This terminates the description of the algorithm.

As for the time complexity analysis, the time complexity of the first phase of the algorithm is $dN(d \log N + (n + k)(d + \log k))$. Let N^* be the number of candidate outliers at the end of the first phase. Then the time complexity of the second phase is $N^*(\log n + N^*(d + \log k))$.

4 Experimental Results and Conclusions

We implemented the algorithm using the C programming language on a Pentium III 850MHz based machine having 512Mb of main memory. We used a 64 bit floating-point type to represent the coordinates of the points and the distances, and the 32th order approximation of the d -dimensional Hilbert curve to map the hypercube $[0, 2)^d$ onto the set of integers $[0, 2^{32d})$. We studied the behavior of the algorithm when the dimensionality d and the size N of the data set, the number n of top outliers we are searching for, the number k of nearest neighbors to consider and the metric L_t are varied. In particular, we considered $d \in \{2, 10, 20, 30\}$, $N \in \{10^3, 10^4, 10^5, 10^6\}$, $n, k \in \{1, 10, 100, 1000\}$, and the metrics L_1 , L_2 and L_∞ . We also studied how the number of candidate outliers decreases during the execution of the algorithm. To test our algorithm, we used three families of data sets called GAUSSIAN, CLUSTERS and DENSITIES. A data set of the GAUSSIAN family is composed by points generated from a normal distribution and scaled to fit into the unit hypercube. A data set of the CLUSTERS family is composed by 10 hyper-spherical clusters, formed by the same number of points generated from a normal distribution, having diameter 0.05 and equally spaced along the main diagonal of the unit hypercube. Each cluster is surrounded by 10 equally spaced outliers lying on a circumference of

radius 0.1 and center in the cluster center. A data set of the DENSITIES family is composed by 2 gaussian clusters composed by the same number of points but having different standard deviations (0.25 and 0.75 respectively). The data sets of the same family differs only for their size N and for their dimensionality d . Figure 4 (a) shows the two dimensional GAUSSIAN data set together with its top 100 outliers (for $k = 100$) with $N = 10000$ points. In all the experiments considered, the algorithm terminates with the exact solution after executing a number of iterations much less than $d + 1$. Thus, we experimentally found that in practice the algorithm behaves as an exact algorithm without the need of the second phase. The algorithm exhibited the same behavior on all the considered data set families. For the lack of space, we report only the experiments relative to the GAUSSIAN data set (see [2] for a detailed description). Figures 4 (b) and (c) show the execution times obtained respectively varying the dimensionality d and the size N of the data set. The curves show that the algorithm scales linearly both with respect to the dimensionality and the size of the data set. Figures 4 (d) and (e) report the execution times obtained varying the number n of top outliers and the type k of outliers respectively. In the range of values considered the algorithm appears to be slightly superlinear. Figure 4 (f) illustrates the execution times corresponding to different values t of the metric L_t . Also in this case the algorithm scales linearly in almost all the experiments. The algorithm scales superlinearly only for L_∞ with the GAUSSIAN data set. This happens since, under the L_∞ metric, the points of the GAUSSIAN data set tend to have the same weight as the dimensionality increases. Finally, we studied how the number of candidate outliers decreases during the algorithm. Figure 4 (g) reports, in logarithmic scale, the number of candidate outliers at the beginning of each iteration for the thirty dimensional GAUSSIAN data set and for various values of the data set size N . These curves show that, at each iteration, the algorithm is able to discharge from the set of the candidate outliers a considerable fraction of the whole data set. Moreover, the same curves show that the algorithm terminates, in all the considered cases, performing less than 31 iterations (5, 7, 10 and 13 iterations for N equal to 10^3 , 10^4 , 10^5 and 10^6 respectively). Figure 4 (h) reports, in logarithmic scale, the number of candidate outliers at the beginning of each iteration for various values of the dimensionality d of the GAUSSIAN data set with $N = 100000$. We note that, in the considered cases, if we fix the size of the data set and increase its dimensionality, then the ratio $\bar{d}/(d + 1)$, where $\bar{d}$ is the number of iterations needed by the algorithm to find the solution, sensibly decreases, thus showing the very good behavior of the method for high dimensional data sets.

To conclude, we presented a distance-based outlier detection algorithm to deal with high dimensional data sets that scales linearly with respect to both the dimensionality and the size of the data set. We presented experiments up to 1000000 of points in the 30-dimensional space. We are implementing a disk-based version of the algorithm to deal with data sets that cannot fit into main memory.

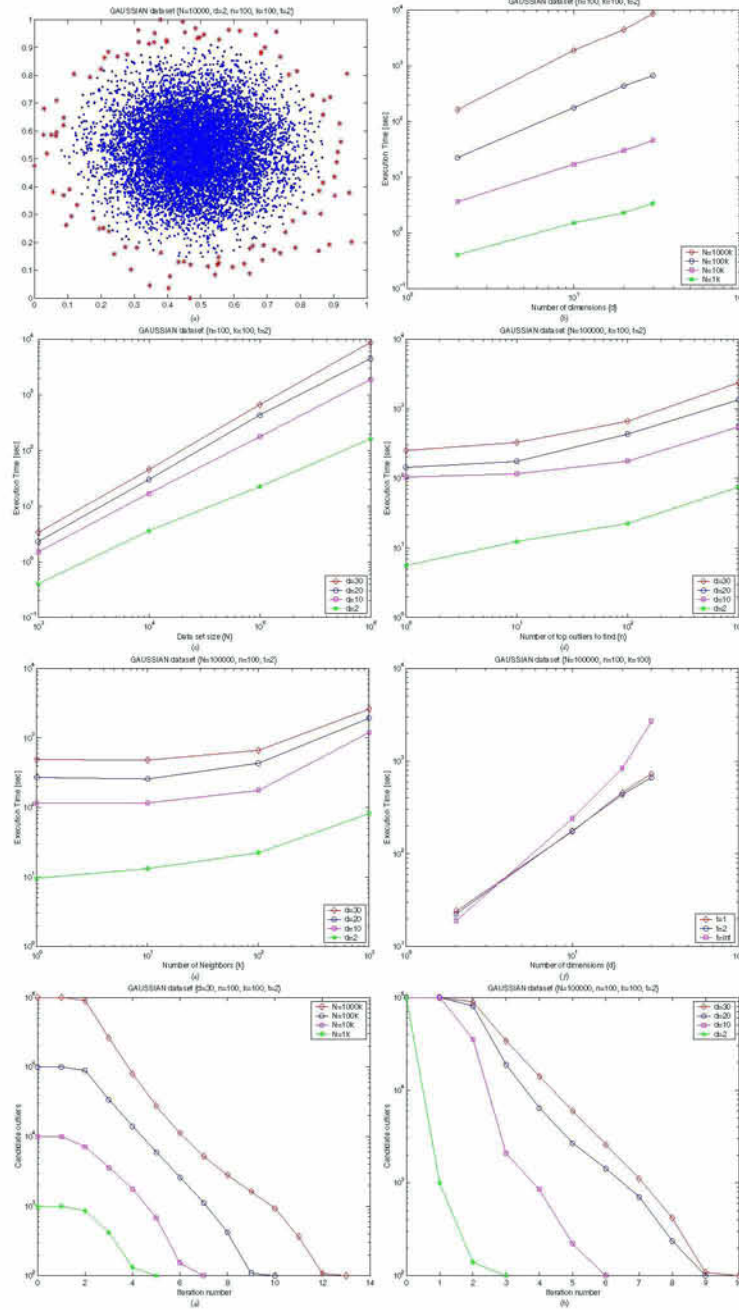


Fig. 4. Experimental results

References

1. C. C. Aggarwal and P. S. Yu. Outlier detection for high dimensional data. In *Proc. ACM Int. Conference on Management of Data (SIGMOD'01)*, 2001. 16
2. F. Angiulli and C. Pizzuti. Fast outlier detection in high dimensional spaces. In *Tech. Report, n. 25, ISI-CNR*, 2002. 20, 23, 24
3. A. Arning, C. Aggarwal, and P. Raghavan. A linear method for deviation detection in large databases. In *Proc. Int. Conf. on Knowledge Discovery and Data Mining*, pages 164–169, 1996. 16
4. V. Barnett and T. Lewis. *Outliers in Statistical Data*. John Wiley & Sons, 1994. 16
5. M. M. Breunig, H. Kriegel, R. T. Ng, and J. Sander. Lof: Identifying density-based local outliers. In *Proc. ACM Int. Conf. on Management of Data (SIGMOD'00)*, 2000. 16
6. C. E. Brodley and M. Friedl. Identifying and eliminating mislabeled training instances. In *Proc. National American Conf. on Artificial Intelligence (AAAI/IAAI 96)*, pages 799–805, 1996. 15
7. Yu D., Sheikholeslami S., and A. Zhang. Findout: Finding outliers in very large datasets. In *Tech. Report, 99-03, Univ. of New York, Buffalo*, pages 1–19, 1999. 16
8. C. Faloutsos and S. Roseman. Fractals for secondary key retrieval. In *Proc. ACM Int. Conf. on Principles of Database Systems (PODS'89)*, pages 247–252, 1989. 18
9. J. Han and M. Kamber. *Data Mining, Concepts and Technique*. Morgan Kaufmann, San Francisco, 2001. 15
10. H. V. Jagadish. Linear clustering of objects with multiple attributes. In *Proc. ACM Int. Conf. on Management of Data (SIGMOD'90)*, pages 332–342, 1990. 16
11. H. V. Jagadish. Linear clustering of objects with multiple attributes. In *Proc. ACM Int. Conf. on Management of Data (SIGMOD'90)*, pages 332–342, 1990. 18
12. E. Knorr and R. Ng. Algorithms for mining distance-based outliers in large datasets. In *Proc. Int. conf. on Very Large Databases (VLDB98)*, pages 392–403, 1998. 16
13. M. Lopez and S. Liao. Finding k -closest-pairs efficiently for high dimensional data. In *Proc. 12th Canadian Conf. on Computational Geometry (CCCG)*, pages 197–204, 2000. 19
14. S. Ramaswamy, R. Rastogi, and K. Shim. Efficient algorithms for mining outliers from large data sets. In *Proc. ACM Int. Conf. on Management of Data (SIGMOD'00)*, pages 427–438, 2000. 16
15. Hans Sagan. *Space Filling Curves*. Springer-Verlag, 1994. 18
16. S. Sarawagi, R. Agrawal, and N. Megiddo. Discovery-driven exploration of olap data cubes. In *Proc. Sixth Int. Conf on Extending Database Thecnology (EDBT)*, Valencia, Spain, March 1998. 16
17. J. Shepherd, X. Zhu, and N. Megiddo. A fast indexing method for multidimensional nearest neighbor search. In *Proc. SPIE Conf. on Storage and Retrieval for image and video databases VII*, pages 350–355, 1999. 18, 19
18. Z. R. Struzik and A. Siebes. Outliers detection and localisation with wavelet based multifractal formalism. In *Tech. Report, CWI, Amsterdam, INS-R0008*, 2000. 16
19. K. Yamanishi and J. Takeuchi. Discovering outlier filtering rules from unlabeled data. In *Proc. ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, pages 389–394, 2001. 16

20. K. Yamanishi, J. Takeuchi, G. Williams, and P. Milne. On-line unsupervised learning outlier detection using finite mixtures with discounting learning algorithms. In *Proc. ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, pages 250–254, 2000. [16](#)

Data Mining in Schizophrenia Research - Preliminary Analysis

Stefan Arnborg^{1,2}, Ingrid Agartz³, Håkan Hall³, Erik Jönsson³,
Anna Sillén⁴, and Göran Sedvall³

¹ Royal Institute of Technology
SE-100 44, Stockholm, Sweden
`stefan@nada.kth.se`

² Swedish Institute of Computer Science

³ Department of Clinical Neuroscience, Section of Psychiatry, Karolinska Institutet
SE-171 76 Solna, Sweden

⁴ Department of Clinical Neuroscience, Karolinska Institutet
SE-171 76 Solna, Sweden

Abstract. We describe methods used and some results in a study of schizophrenia in a population of affected and unaffected participants, called patients and controls. The subjects are characterized by diagnosis, genotype, brain anatomy (MRI), laboratory tests on blood samples, and basic demographic data. The long term goal is to identify the causal chains of processes leading to disease. We describe a number of preliminary findings, which confirm earlier results on deviations of brain tissue volumes in schizophrenia patients, and also indicate new effects that are presently under further investigation. More importantly, we discuss a number of issues in selection of methods from the very large set of tools in data mining and statistics.

1 Introduction

Mental disorders account for a very significant part of total disability in all societies. In particular, every large human population in all parts of the world shows an incidence of schizophrenia between 0.5% and 1.3%. As for other mental disorders, the cause of the disease is not known, but it has been statistically confirmed that genetic factors and environmental factors before, during and immediately after birth affect its incidence. There is no treatment that cures the disease. Schizophrenia usually leads to life-long disability at great cost for the affected individuals and their families as well as for society.

The HUBIN[13] multi-project is a set of projects aimed at understanding the mechanisms behind mental disorders and in particular schizophrenia. Despite the statement above that the cause of schizophrenia is not known, there are several current and serious hypotheses[3,18]. These center around the development of the neuronal circuitry before birth and during childhood. This development is assumed to be influenced by factors such as genotype, infections, stress and

social stimulus. The signs of this process can be seen in clinical journals, neuropsychological and psychiatric assessments, and physiological measurements of brain structure and blood contents.

We will describe preliminary findings, and also how the research questions and data available influenced the selection of data analysis methods. These methods are typically adaptations of known tools in statistics and data mining. In section 2 we outline data acquisition, in section 3 the data analysis strategy. Sections 4 and 5 deal with frequentist association assessment and Bayesian multivariate characterization of collected data, respectively. In section 6 we show how the false discovery rate method was used to focus future collection of genetics data, and in section 7 we describe how supervised and unsupervised classification methods are applied to approach our research questions.

2 Data Acquisition

The participants included in the study are affected patients with schizophrenia and controls. Each individual has given written consent to participate as regulated by Karolinska Institutet and the 1964 Helsinki Declaration. Exclusion criteria are several conditions that are known to cause unwanted effects on measured variables, among others organic brain disease and brain trauma. Affected individuals were schizophrenia patients recruited from the northern Stockholm region. The control group was recruited from the same region and matched to the affected group with respect to age and gender. All participants underwent interview by an experienced psychiatrist to confirm schizophrenia in the affected group and absence of mental disorders in the control group.

For a set of genes believed to be important for systems disturbed in persons developing schizophrenia, most of the participants were genotyped using single nucleotide polymorphisms (SNP:s). This characterization was obtained using the pyrosequencing method[2].

Participants were investigated in an MR scanner using a standard protocol giving a resolution of 1.5 mm. This protocol admits reliable discrimination of the main brain tissues grey and white matter and cerebro-spinal fluid (CSF), as well as other tissue or fluid like venous blood. Volumes of specific tissues/fluids in regions of interest were obtained by weighted voxel counting. A more detailed description of MR data acquisition can be found in [1].

Blood samples were obtained in which the concentration of certain substances and metabolites are measured with standard laboratory tests. Standard demographic data were obtained, like gender, month of birth, age, and age at first admittance for psychiatric care for the patients.

The choice of variables is determined by current medical hypotheses held by researchers and the possibility of obtaining high quality measurements with reasonable expenditure. Ongoing work aims at collection of detailed psychiatric characterizations, neuropsychological variables and additional genetics information.

3 Data Analysis

The long term goal is to understand the causal chains leading to the disease. It is believed that associations in the data can give important clues. Such clues are then used to determine new data acquisition strategies to confirm preliminary findings and hypotheses.

The most voluminous part of the data set used in this investigation is structural MRI information. The MR scans were converted to 3D images and processed by the BRAINS software developed at University of Iowa[4,21], to make possible comparisons of corresponding anatomical brain regions in different subjects. This is necessary because of the large individual variations in brain size and shape. The volume of each tissue or fluid type is obtained in a number of regions, like the frontal, temporal, parietal, occipital, subcortical, brainstem and ventricular regions. A number of anatomically distinguishable regions in the vermis of cerebellum (posterior inferior, posterior superior, and anterior vermis) and the cerebellar hemisphere were manually traced (because we have no means to identify them automatically) and measured (total tissue volume only). The reason for including the vermis region is that it is involved in control of eye movements, which are atypical for persons with schizophrenia. The vermis data have been previously analyzed for a limited number of male participants [17]. In the data used here, there are 144 participants, 63 affected and 81 controls, with 30 brain region variables given both in absolute value (ml) and relative to intracranial volume, six summary brain size measures (total volume of discriminated tissue/fluid types), 5 manually measured cerebellar volumes (with absolute and relative values), 58 blood test variables, 20 genetic (SNP) variables (all except 8 were unfortunately uninformative in the sense that almost all participants had the same type), 8 demographic variables, making altogether 144 variables. For some of these there are missing values, which can be regarded as missing completely at random.

The ultimate goal of schizophrenia research is to explain the disease with its large psychiatric and physiological diversity, and to find a good treatment. A more immediate goal is to find tentative answers to the following questions:

- How can causal chains leading to disease be found from our observational data?
- Is it possible to predict the diagnosis and a persons psychiatric conditions from physiological data?
- Do the categorizations used by psychiatrists correspond to recognizable classes for physiological variables?

We have experimented with many available data mining approaches applicable for our type of data, and some useful experiences can be reported. The standard methods described in textbooks are often not immediately applicable, but a method selection and adaptation is called for depending both on the questions to which the answers are sought and on the detailed characteristics of the data. There are many effects of great interest that are very weak or possibly only

noise in our data, so statistical significance concepts are very important. On the other hand, once an effect has been considered significant and interesting, the best way to communicate the nature of the effect is typically as a graph set - graphical models or scatter plots.

There are recently developed methods to find causal links in observational data sets, usually based on the identifiability of arc directions in directed graphical models. Testing our data against a number of different such methods[12], it turned out that in several instances the participants age, month of birth and genotype came out as caused by the phenotype, *e.g.*, the size of a part of the brain would have been a cause of the persons DNA variant. This is highly unlikely to be correct, since the genotype is determined at time of conception, before the development of the individual starts. This finding confirms our belief that the variables presently measured do not include all important information needed to ultimately explain the disease. Our study is thus at this stage oriented somewhat humbly to finding fragments of the processes leading to disease.

4 Association Tests Based on Randomizations

The general test method is as follows: We investigate the null hypothesis that the diagnosis was determined at random after the variables were measured. A test statistic was chosen giving a 'difference' between affected and controls, and its value for the data was compared to the cumulative distribution of the test statistics from many random assignments of the diagnosis (in the same proportion as in the original data[10]). The p -value obtained is the proportion of more extreme test statistics occurring in the randomized data. As test statistics for single variables we chose the difference between affected and controls in mean and variance. For pairs of variables we normalize the variables and find the angle between the directions, for patients and for controls, of largest variation. The absolute value of the cosine of this angle is used as test statistic.

Multiple Comparison Considerations

The p -values obtained show what the significance would be if the corresponding test was the only one performed, it being customary to declare an effect significant if the p -value is below 1% or 5% depending on circumstances. However, since many variables and variable pairs were tested, one would expect our tables of significant effects to contain hundreds of spurious entries – even if there were no real effects.

In family-wise error control (FWE[15]), one controls the probability of finding at least one erroneous rejection. A Bonferroni correction divides the desired significance, say 5%, with the number of tests made, and the p -values below this value are stated as significant. More sophisticated approaches are possible.

A recent proposal is the control of false discovery rate[6]. Here we are only concerned that the rate (fraction) of false rejections is below a given level. If this rate is set to 5%, it means that of the rejected null hypotheses, on the average no

more than 5% are falsely rejected. It was shown that if the tests are independent or positively correlated in a certain sense, one should truncate the rejection list at element k where $k = \max\{i : p_i \leq qi/m\}$, m is the number of tests and p_i is the ordered list of p -values. This cut-off rule will be denoted FDRi.

If we do not know how the tests are correlated, it was also shown in [7] that the cut-off value is safe if it is changed from qi/m to $qi/(mH_m)$, where $H_m = \sum_{i=1}^m 1/i$. This rule is denoted FDRd. The most obvious correlations induced by the testing in our application satisfy the criterion of positive (monotone) correlation of [7].

Table 1. Significant associations at 5% with different corrections for multiple testing

	m	Bonf	FDRi	FDRd	no correction
mean	95	28	52	34	56
variance	95	25	28	26	37
angle	4371	53	412	126	723

The result of applying various FWE and FDR corrections at 5% are shown in table 1. The conclusion is that there is most likely a large number of dependencies among the variables – many more than those found significant above – and the pattern is apparently not explainable by simple models. In application terms, one could say that the disease interacts globally with the development of the brain and permeates into every corner of it. In order to obtain a reasonable amount of clues, we must obviously consider how to find the most important effects. This is a common concern analyzing large and disparate statistical data sets obtainable with modern technology. It has been proposed that maybe Bayes factors are better indicators than p -values[16] of effects. The question is not settled, but let us try the Bayesian method and see what we get.

5 Bayesian Association Determination

The Bayesian paradigm does not work by rejecting a null hypothesis, but compares two or more specific hypotheses. In our case, hypotheses are compared for each possible association, and the relative support the data give them are summarized as Bayes factors for one against the rest. We have not given detailed prior probabilities to the hypotheses. We can check for multiple testing effects by introducing costs for the two types of error possible. This will have exactly the same effect as a prior probability promoting the null hypothesis. We have penalized for mass testing by giving low prior odds for the dependency hypothesis, so that on the whole our prior information is that on the average only one of the variables should be dependent on the diagnosis.

The hypotheses in this case are that the same distribution generated the variables for affected and controls, and that two different distributions generated them, respectively. As distribution family we take piece-wise constant functions, which translates to discretization of the variables. The prior distribution over the family is taken to be a Dirichlet distribution. Then the standard association tests of discrete distributions used *e. g.* in graphical model learning[5,14] are applied. An empirical Bayes approach is used, where the granularity is chosen to give a sufficient number of points in each discretization level.

Bayesian Association Models

For a chosen discretization, a variable will be described as an occurrence vector $(n_i)_{i=1}^d$, where d is the number of levels and n_i is the number of values falling in bin i . Let $\bar{x} = (x_i)_{i=1}^d$ be the probability vector, x_i being the probability of a value falling in bin i . A Bayesian association test for two variables is a comparison of two hypotheses, one H_d in which the variables are jointly generated and one H_i in which they are independently generated.

Table 2. Bayesian association (log Bayes factor), variable to diagnosis. Strongly associated variables are brain regions, but also serum triglycerides

Variable	$\log(BF)$	Variable	$\log(BF)$
rel post sup vermis	8.08	serum triglycerides	2.91
abs post sup vermis	7.77	rel post inf vermis	2.78
rel temporal CSF	6.37	abs post inf vermis	2.71
abs total vermis	5.68	abs ventricular white	2.55
rel total vermis	5.18	rel total CSF	2.35
abs temporal CSF	4.29	rel ventricular white	2.34
ratio CSF/grey	4.25	abs anterior vermis	2.32
rel brainstem CSF	3.41	rel ventricular CSF	2.24
rel total CSF	3.27	abs subcortical white	2.23
abs brainstem CSF	3.08	abs ventricular CSF	2.1
abs total CSF	3.06	rel anterior vermis	1.89

Table 2 gives the log Bayes factors, $\log(p(\bar{n}|H_d)/p(\bar{n}|H_i))$, of H_d against H_i for variables discretized into 5 levels. Assuming the previously mentioned prior, entries with log Bayes factor above 2 would be deemed significant.

For the co-variation investigation we chose to compare the eight undirected graphical models on triples of variables, one of which is the diagnosis. If the graph described by the complete graph, a triangle, on the three variables has high posterior probability, then this means that the variation of the data cannot be described as resulting from influence of the diagnosis on one of the two variables or as independent influence on both – the association between the variables is different for affected and controls. In figure 1, the left graph represents

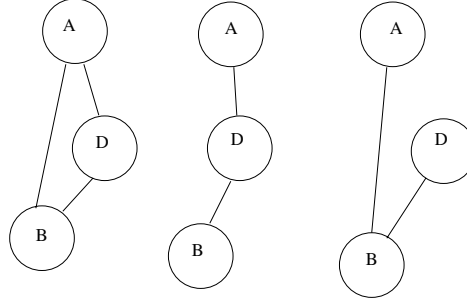


Fig. 1. Graphical models detecting co-variation

the type of co-variation we are looking for. The next graph explains data as the diagnosis D affecting variables A and B separately, whereas the rightmost graph describes a situation where the association between A and B is similar for affected and controls. This method can be generalized to higher order interactions, but we need substantially more data before this becomes meaningful. In both frequentist and Bayesian pairwise variable association studies, the posterior superior vermis was highly linked via the diagnosis to several other variables. Particularly interesting is the age variable, which is known to be independent (figure 2(b)). For patients the posterior superior vermis is smaller and not dependent on age, whereas for controls it decreases with age. The hypothesis that the change in vermis size develops before outbreak is natural and made even more likely by not being visibly dependent of medication and length of the disease period.

Bayesian association estimates are the basis for graphical models giving an overview of co-variation of variable sets. Based on the strength of pairwise variable associations, decomposable graphical models were obtained from the data matrix. For the matrix containing demographic, physiology and automatically measured white, gray and CSF volumes, and genotype, the central part of the

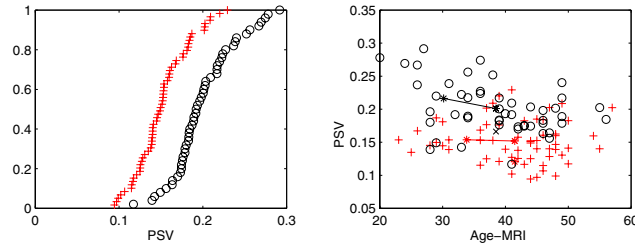


Fig. 2. (a) Empirical cumulative distributions for posterior superior vermis, + : affected, o : controls. (b) Scatter plot of association in angle of principal directions of variation, $\log p \approx -3$

model was found to be as shown in figure 3(a). In words, the diagnosis is most distinctly, in statistical terms, associated with the CSF volumes in the brainstem and temporal regions.

Even more closely associated are the vermis regions that were measured manually (volumes only). When the vermis variables are also included, the model looks like figure 3(b). The position of the temporal CSF variable in the diagram suggests that a brain region affected similarly to the vermis can be located in the temporal boxes.

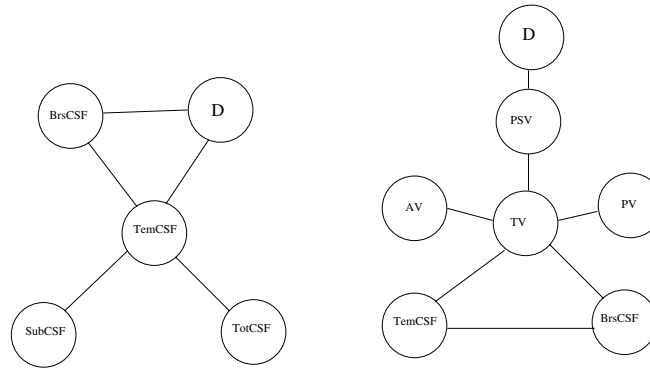


Fig. 3. Graphical models, neighborhoods of Diagnosis (D) - (a)grey/white/CSF volumes; (b) Vermis volumes added

6 Genetics Data

The genetic variables were not strongly associated to phenotype variables measured. This does not mean that they are uninteresting. The basic machinery of biological systems is run by proteins, and the genes are blueprints of these proteins. The different variations, alleles, of a gene result in small variations in the proteins they produce. For single gene diseases there is often a single variation of the protein that immediately leads to the disease, but for multiple gene diseases, to which schizophrenia apparently belongs, there is a whole family of genes with variants that have small effects on the disposition for disease, in other words they push the whole organisms development slightly in the direction where the etiology of disease can begin to unfold. The complexities of these processes are overwhelming, and although a large amount of knowledge has been accumulated over a few decades, it is fair to say that even more is presently unknown.

The SNP genotyping separates the alleles of a gene into two classes, and these two classes can be slightly different in function of the protein and its effect on the development of an individual. By finding associations between genotype

and other variables, information about the role of the corresponding protein and its metabolites in development can be extracted. Because of the weakness of the statistical signals, genetics data must be examined with the most powerful – but sound – statistical methods available.

The informative genes and their polymorphisms measured for the population are shown in table 3.

Table 3. Genes with informative SNP:s

Gene	SNP	type	polym	informative	function
DBH	Ala55Ser	G/T	98 24 0		dopamine beta-hydroxylase
DRD2	Ser311Cys	G/C	118 4 0		dopamine receptor D2
DRD3	Ser9Gly	A/G	49 59 14		dopamine receptor D3
HTR5A	Pro15Ser	C/T	109 11 2		serotonin receptor 5A
NPY	Leu7Pro	T/C	1 7 114		neuropeptide Y
SLC6A4	ins/del	S/L	20 60 42		serotonin transporter
BDNF	Val66Met	A/G	5 37 80		brain derived neurotrophic factor

A Bayesian comparison with the diagnosis variable speaks weakly in favor of independence between genotype and diagnosis for all polymorphisms available. The same is true when using Fishers exact test. But we have also many other variables related to brain development. Testing the genes against all variables, 63 p -values below 5% were found. However, applying the FDRd or FDRi correction on 5% false rejection rate, none of these survive. It is somewhat remarkable, however, that 30 small p -values are related to the polymorphism in the BDNF gene. There is thus ground for the suspicion that this polymorphism has a an influence on brain development that will probably identifiable in MR images with slightly more cases. It is possible to state this influence in statistical terms: in the FDR sense, on the average 80% of 30 variables are affected by this polymorphism. Among these variables are the manually traced posterior inferior vermis, gray matter in the frontal, parietal, subcortical, temporal and ventricular regions. Interestingly, a Bayesian modeling with a linear model and using SSVS variable selection of [11] does not show this effect unless the noise level is forced down unreasonably. This can be explained by the imperfect fit to the linear model assumptions. Thus, in this case the randomization and FDR methods are instrumental because they give easily a justifiable inference that points to a subset of the included genes as promising, and allows further data collection to concentrate on these.

Some of the strongest associations found are shown in figure 4. In summary, the current genetics data show that there are likely genetic dependencies of variables, but the statistical power is not yet adequate to identify specific associations with high significance.

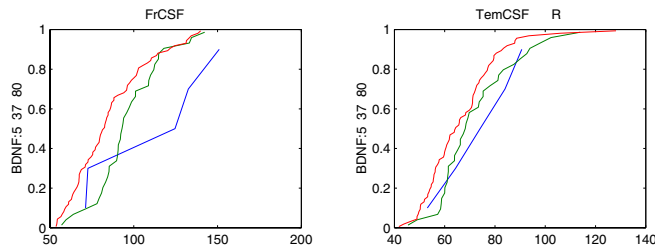


Fig. 4. Possible genetic effects picked out by p -values and Bayesian model comparisons. Empirical cumulative distributions of the variable (frontal CSF and temporal CSF) for the three types (A/A, A/G and G/G) of SNP in gene BDNF.

7 Prediction and Classification Study

It is a known difficult problem to determine the diagnosis of schizophrenia from physiological variables. The problem is important for the potential of early risk detection in research and for treatment. We checked that this is also the case for our data sets. Supervised classifiers were built using both support vector[9] and decision tree[5] techniques. When trained on random samples of 75% of the participants and tested on the remaining individuals, a classification accuracy of approximately 78% was obtained. If only the optimal discriminating single variable, post sup vermis, is used with the same training process, 71% accuracy is obtained, and it thus has a dominating explanatory power (see figure 2(a)). Another technique that has been useful is unsupervised classification. The interest in this problem is motivated by the possibility of there being several different processes leading to disease with different physiological traces, and for checking against the wide span of symptom sets that develop in different patients. Classifications of schizophrenia patients have usually been performed with cluster analysis paradigms[19]. The AUTOCLASS paradigm tries to find classifications with high probability, under assumptions of particular distributions for the variables within each class. We ran the AUTOCLASS software[8] on a subset of participants and variables without missing values. The variables were assumed to be independent, categorical or normally distributed within each class. AUTOCLASS searches for the classification in the form of a probability mixture with maximal probability of generating the data. In our case, a four class mixture was identified for the population consisting of 42% affected. The most important variable for the classification was total (absolute) volume of gray matter. One of the classes has mainly (75%) controls and a high value for gray matter. The next class has a high proportion of patients (83%), somewhat less but dispersed (high variance) gray matter. A third class has the same amount of gray matter, but with low variance and not very discriminated *wrt* diagnosis (33% affected). The final class has low volume of gray matter and 34% affected. Other classification approaches are presently under investigation.

8 Summary and Conclusions

The application findings reported above are interesting although the ultimate explanation of schizophrenia seems still far away.

On the methodological side, it is clear that the data mining philosophy implies making a large number of tests and similar investigations involving both p -values, Bayesian model comparisons and visual inspections. We found Bayesian analysis of graphical model sets useful for characterization of multivariate data with obvious interdependencies, whereas randomization and the FDR method was more sensitive for detecting associations between genotype and phenotype. It is also obvious that the methods complement each other - testing is appropriate when the null hypothesis can be precisely formulated and then for any test statistic randomization tests can give a p -value, the classical risk that an erroneous conclusion is drawn. But Bayesian model comparison is a very natural and easily implemented method that gives answers also when there is no obvious null hypothesis. Lastly, graphical visualizations are necessary as confirmations of statistical effects.

The strict control of multiple comparison effects can not easily be formalized in typical data mining since many investigations are summarily discarded for lack of obvious effects or lack of interest from application experts. Our method must make an intricate balance between creation of significance and creation of sense - without inventing an implausible story. This is a problem that seems not yet fully addressed, neither in the statistics nor in the data mining communities. Some further development of the q -values proposed by Storey[20] might be useful for this purpose

Acknowledgments

The cerebellum variables included in this study were obtained by Gaku Okugawa. The AUTOCLASS run results were obtained by Can Mert. We acknowledge valuable discussions with Stig Larsson, Tom McNeil, Lars Terenius and Manuela Zamfir. The HUBIN project is funded by the Wallenberg foundation.

References

1. I. Agartz, V. Magnotta, M. Nordström, G. Okugawa, and G. Sedvall. Reliability and reproducibility of brain tissue volumetry from segmented MR scans. *European Archives of Psychiatry and Clinical Neuroscienc*, pages 255–261, 2001. 28
2. A. Ahmadian, B. Gharizadeh, A. C. Gustafsson, F. Sterky, P. Nyren, M. Uhlen, and J. Lundeberg. Single-nucleotide polymorphism analysis by pyrosequencing. *Analytical Biochemistry*, 2000. 28
3. N. C. Andreasen. Linking mind and brain in the study of mental illnesses: a project for a scientific psychopathology. *Science*, 275:1586–1593, 1997. 27
4. N. C. Andreasen, R. Rajarethinam, T. Cizaldo, S. Arndt, V. W. II Swayze, L. A. Flashman, D. S. O’Leary, J. C. Ehrherdt, and W. T. C. Yuh. Automatic atlas-based volume estimation of human brain regions from MR images. *J. Comput. Assist. Tomogr.*, 20:98–106, 1996. 29

5. S. Arnborg. A survey of Bayesian data mining. Technical report, 1999. SICS TR T99:08. [32](#), [36](#)
6. Y. Benjamini and Y. Hochberg. Controlling the false discovery rate: A practical and powerful approach to multiple testing. *J. of the Royal statistical Society B*, 57:289–300, 1995. [30](#)
7. Y. Benjamini and D. Yekutieli. The control of the false discovery rate in multiple testing under dependency. Technical report, Stanford University, Dept of Statistics, 2001. [31](#)
8. P. Cheeseman and J. Stutz. Bayesian classification (AUTOCLASS): Theory and results. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*. 1995. [36](#)
9. N. Cristianini and J. Shawe-Taylor, editors. *Support Vector Machines and other kernel based methods*. Cambridge University Press, 2000. [36](#)
10. E. Edgington. *Randomization Tests*. New York, M. Dekker, 1987. [30](#)
11. E. I. George and R. E. McCulloch. Approaches for bayesian variable selection. Technical report, The Univeristy of Texas, Austin, 1996. [35](#)
12. C. Glymour and G. Cooper, editors. *Computation, Causation and Discovery*. MIT Press, 1999. [30](#)
13. Håkan Hall, Stig Larsson, and Göran Sedvall. Human brain informatics - HUBIN web site. 1999. <http://hubin.org>. [27](#)
14. David Heckerman. Bayesian networks for data mining. *Data Mining and Knowledge Discovery*, 1:79–119, 1997. [32](#)
15. Y. Hochberg. A sharper Bonferroni procedure for multiple tests of significance. *Biometrika*, 75:800–803, 1988. [30](#)
16. J. I. Marden. Hypothesis testing: From p -values to Bayes factors. *J. American Statistical Ass.*, 95:1316–1320, 2000. [31](#)
17. G. Okugawa, G. Sedvall, M. Nordström, N. C. Andreasen, R. Pierson, V. Magnotta, and I. Agartz. Selective reduction of the posterior superior vermis in men with chronic schizophrenia. *Schizophrenia Research*, (April), 2001. In press. [29](#)
18. G. Sedvall and L. Terenius. In *Schizophrenia: Pathophysiological mechanisms. Proceedings of the Nobel Symposium 111(1998) on Schizophrenia*. Elsevier, 2000. [27](#)
19. S. R. Sponheim, W. G. Iacono, P. D. Thuras, and M. Beiser. Using biological indices to classify schizophrenia and other psychotic patients. *Schizophrenia Research*, pages 139 – 150, 2001. [36](#)
20. J. Storey. The false discovery rate: A Bayesian interpretation and the q -value. Technical report, Stanford University, Dept of Statistics, 2001. [37](#)
21. R. P. Woods, S. T. Grafton, J. D. Watson, N. L. Sicotte, and J. C. Mazziotta. Automated image registration: II. intersubject validation of linear and non-linear models. *J. Comput. Assist. Tomogr.*, 22:155–165, 1998. [29](#)

Fast Algorithms for Mining Emerging Patterns

James Bailey, Thomas Manoukian, and Kotagiri Ramamohanarao

Department of Computer Science & Software Engineering
The University of Melbourne, Australia
{jbailey,tcm,rao}@cs.mu.oz.au

Abstract. Emerging Patterns are itemsets whose supports change significantly from one dataset to another. They are useful as a means of discovering distinctions inherently present amongst a collection of datasets and have been shown to be a powerful technique for constructing accurate classifiers. The task of finding such patterns is challenging though, and efficient techniques for their mining are needed.

In this paper, we present a new mining method for a particular type of emerging pattern known as a jumping emerging pattern. The basis of our algorithm is the construction of trees, whose structure specifically targets the likely distribution of emerging patterns. The mining performance is typically around 5 times faster than earlier approaches. We then examine the problem of computing a useful subset of the possible emerging patterns. We show that such patterns can be mined even more efficiently (typically around 10 times faster), with little loss of precision.

1 Introduction

Discovery of powerful distinguishable features between datasets is an important objective in data mining. Addressing this problem, work presented in [6] introduced the concept of *emerging patterns*. These are itemsets whose support changes significantly from one dataset to another. Because of sharp changes in support, emerging patterns have strong discriminating power and are very useful for describing the contrasts that exist between two classes of data. Work in [11] has shown how to use them as the basis for constructing highly accurate data classifiers. In this paper, we focus on mining of a particular type of emerging pattern called a *jumping emerging pattern* (JEP). A JEP is a special type of emerging pattern, an itemset whose support increases abruptly from zero in one dataset, to non-zero in another dataset. Due to this infinite increase in support, JEPs represent knowledge that discriminates between different classes of data more strongly than any other type of emerging pattern. They have been successfully applied for discovering patterns in gene expression data [12].

Efficient computation of JEPs remains a challenge. The task is difficult for high dimensional datasets, since in the worst case, the number of patterns present in the data may be exponential. Work in [6] introduced the notion of a border

for concisely representing JEPs. Yet even using borders, the task still has exponential complexity and methods for improving efficiency are an open issue. With the volume and dimensionality of datasets becoming increasingly larger, development of such techniques is consequently crucial. Indeed for large datasets, approximation methods are also necessary, to ensure tractability.

In this paper, we describe algorithms for computing JEPs that are 2-10 times faster than previous methods. Our approach has two novel features: The first is the use of a new tree-based data structure for storing the raw data. This tree is similar to the so-called frequent pattern tree, used in [9] for calculating frequent itemsets. However, there are significant differences in the kinds of tree shapes that promote efficient mining and interesting new issues and tradeoffs are seen to arise. The second feature is the development of a mining algorithm operating directly on the data contained in the trees. The mining of emerging patterns is unlike (and indeed harder than) that of frequent itemsets. Monotonicity properties relied on by algorithms such as a-priori do not exist for JEPs and thus our algorithm requires greater complexity than the techniques in [9].

We then look at the problem of mining only a subset of the JEPs using approximate thresholding techniques. We outline methods which can achieve further speedups from 2-20 times faster and demonstrate that a small number of patterns can still provide sufficient information for effective classification.

Related Work: Emerging patterns first appeared in [6], which also introduced the notion of the border for concisely representing emerging patterns. Unlike this paper, no special data structure was used for mining the JEPs. Techniques for building classifiers using JEPs, whose accuracy is generally better than state-of-the-art classifiers such as C4.5 [16] appeared in [11]. Emerging patterns are similar to version spaces [14]. Given a set of positive and a set of negative training instances, a version space is the set of all generalisations that each match (or are contained in) every positive instance and no negative instance in the training set. In contrast, a JEP space is the set of all item patterns that each match (or are contained in) one or more (not necessarily every) positive instance and no negative instance in the set. Therefore, the consistency restrictions with the training data are quite different for JEP spaces.

Work in [9] presented a technique for discovering frequent itemsets (which are useful in tasks such as mining association rules [1]). The primary data structure utilised was the Frequent Pattern Tree (**FP-tree**), for storing the data to be mined. The trees we use in this paper are similar but important new issues arise and there are also some significant differences. Given that we are mining emerging patterns and there are multiple classes of data, tree shape is a crucial factor. Unlike in [9], building trees to allow maximum compression of data is not necessarily desirable for mining emerging patterns and we show that better results are obtained by sacrificing some space during tree construction.

Recent work in [8] also uses trees for calculation of emerging patterns. The focus is different, however, since the algorithm is neither complete nor sound (i.e. it does not discover all JEPs and indeed may output itemsets which aren't

actually JEPs). In contrast, work in this paper focuses on both i) Sound and complete mining of JEPs and ii) Sound but not complete JEP mining.

The emerging pattern (EP) mining problem can also be formulated as discovering a theory that requires the solution to a conjunction of constraints. Work in [5,10] defined three constraint types; i) $f \leq p$, $p \leq f$, $\neg(f \leq p)$ and $\neg(p \leq f)$ ii) $freq(f, D)$ iii) $freq(f, D_1) \leq t$, $freq(f, D_2) \geq t$. Using the first and third, JEP mining for some class D_i with reference D_j can be expressed as; $solution(c_1 \wedge c_3)$ where f is a JEP, $p \in D_i$ and $t = 0$.

Other methods for mining EPs have relied upon the notion of borders, both as inputs to the mining procedure in the form of large borders and as a means of representing the output. [7] employed the Max-Miner [2] algorithm whilst work in [15] is also applicable in generating large borders. JEP mining procedures do not require such sophisticated techniques. Rather than generate large borders, horizontal borders [6] are sufficient. Work in [13] restricts border use to subset closed collections and allows minimal elements that do not appear in the base collections. The borders used in this paper reflect interval closed collections and contain only minimal elements derived from the base collections.

An outline of the remainder of this paper is as follows. In section 2 we give some necessary background and terminology. Section 3 presents the tree data structure we use for mining JEPs and describes several variations. Section 4 gives the algorithm for complete mining of JEPs using this tree and Section 5 gives an experimental comparison with previous techniques. Section 6 then discusses approximate methods for mining a subset of the patterns present. Finally, in section 7 we provide a summary and outline directions for future research.

2 Background and Terminology

Assume two data sets D_1 and D_2 , the growth rate of an itemset i in favour of D_1 is defined as $\frac{support_{D_1}(i)}{support_{D_2}(i)}$. An Emerging Pattern [6] is an itemset whose support in one set of data differs from its support in another. Thus a ρ Emerging Pattern favouring a class of data C , is one in which the growth rate of an itemset (in favour of C) is $\geq \rho$. This growth rate could be finite or infinite. Therefore, we define another type of pattern, known as a *jumping emerging pattern* (JEP), whose growth rate must be infinite (i.e. it is present in one and absent in the another). JEPs can be more efficiently mined than general emerging patterns and have been shown to be useful in building powerful classifiers [11]. We will illustrate our algorithms for mining JEPs assuming the existence of two datasets D_p (the positive dataset) and D_n (the negative dataset). The mining process extracts all patterns (i.e. itemsets) which occur in D_p and not in D_n .

A *border* [6] is a succinct representation of some collection of sets. [6] also showed that the patterns comprising the left bound of the border representing the JEP collection are the most expressive. Therefore, our procedures will be referring to mining the left bound border of the JEP collection.

In previous work, mining of JEPs used a cross-product based algorithm known as *border-diff* [6]. It takes as input some transaction in D_p from which

one wishes to extract JEPs (the positive transaction) and a set of transactions from D_n (negative transactions). Its output is then all JEPs from this positive instance (i.e. all subsets of the positive transaction which do not appear within any negative transaction). We will make use of the *border-diff* algorithm, but use the structure of the tree in order to determine when it should be called and what input should be passed to it. This results in significant performance gains.

Classification using JEPs is described in [11]. Initially all JEPs for each of the classes are computed - *observe this needs to be done once only for the datasets* (the training time). Then, given some test instance, a score is calculated for each class. This score is proportional to the number of JEPs (from the class being examined) contained within the test. Typically the contribution of an individual JEP to the overall score is some function of its support (hence JEPs with high support have a greater influence on classification). The test instance is deemed to match the class with the highest overall score.

3 Trees

The tree based data structure we use for mining JEPs is based on the frequent pattern tree [9]. Since we are dealing with several classes of data, each node in the tree must record the frequency of the item for each class. Use of a tree structure provides two possible advantages:

- when multiple transactions share an itemset, they can be merged into individual nodes with increased counts. This results in compression proportional to the number of itemsets which share some prefix of items and the length of the prefix. Such compression can allow the data structure to be kept in memory and thus accessed faster, rather than being stored on disk.
- Different groupings of positive transactions (those from D_p) and negative transactions (those from D_n) become possible. The efficiency of mining is highly dependent on how this grouping is done. We now examine how transactions are ordered to achieve different groupings.

In choosing an appropriate ordering for the items contained within the itemsets being inserted into the tree, we have the following 2 aims i) To minimise the number of nodes in the tree and ii) To minimise the effort required in traversing the tree to mine JEPs. [9] addressed the first of these in the context of computing frequent itemsets. However, for mining JEPs, we will see that the second is the more important. We have investigated six types of orderings.

Frequent tree ordering. Same as in [9]. Take each item and find its probability in the set $(D_p \cup D_n)$. Items are ordered in descending probability. This ordering aims to minimise the number of nodes in the tree.

Ratio tree ordering and inverse ratio tree ordering. Let the probability of an item in D_p be p_1 and its probability in D_n be p_2 . For $p = p_1/p_2$, order items in descending value of p . The intuition here is that we expect JEPs to reside much higher up in the tree than they would under the frequent tree ordering and this

will help limit the depth of branch traversals needed to mine them. The inverse ratio ordering is just the reverse of this ordering.

Hybrid ordering. A combination of the ratio tree ordering and the frequent tree ordering. First, calculate both the ratio tree ordering and frequent tree ordering. For a given percentage α , the initial α items are extracted from and ordered according to the ratio ordering. All items not yet covered are then ordered according to the frequent ordering. The hybrid ordering thus produces trees which are ordered like a ratio tree in the top α segment and like a frequent tree in the bottom $1 - \alpha$ segment. The intuition behind it is that trees are created which possess both good compression characteristics as well as good mining properties.

Least probable in the negative class ordering (LPNC). Let the probability of an item in D_n be p . Items are ordered in ascending value of p . The intuition behind this ordering is similar to that for the ratio ordering, JEPs are likely to occur higher up in the tree, since the quantity of nodes higher up in the tree containing zero counts for the negative classes is greater.

Most probable in the positive class ordering (MPPC). Let p be the probability of an item in D_p . Items are ordered in descending value of p (first item has highest probability). The intuition here is that by placing nodes higher up in the tree (in accordance with their frequency in the positive class), then if the datasets are inherently dissimilar, we are more likely to find JEPs in the tree's upper regions.

4 Tree-Based JEP Mining

We now describe our tree mining procedure. It uses a core function called *border-diff* [6] with format *border-diff(positive_transaction, vector negative_transactions)*. This function returns the set of JEPs present within *positive_transaction* with reference to the list of *negative_transactions*. i.e. All subsets of *positive_transaction* which don't occur within a member of the negative transaction list. The efficiency of *border-diff* is dependent upon the number of negative transactions which are passed to it, their average dimensionality and the dimensionality of the positive transaction. Our tree mining procedure makes use of this function in a way aimed to reduce all of these parameters.

The initially constructed tree contains a null root node, with each of its children forming the root of a subtree referred to hereafter as a *component tree*. For each component tree, we perform a downwards traversal of every branch. Looking during traversal for nodes which contain a non-zero counter for the class for which we are mining JEPs, and zero counters for every other class (such nodes are called *base* nodes). The significance of these nodes is that the itemset spanning from the root of the branch to the base node is unique to the class being processed. This itemset is therefore a *potential JEP* and hence any subset of this itemset is also potentially a JEP. The importance of base nodes is not simply that they identify potential JEPs, but they also provide a means of partitioning our problem. By considering all branches which share some root node (i.e. all branches of some component tree) and some additional node, (a base node), we can isolate all other transactions containing these two items. Using this set of transactions as the basis for a sub mining problem provides great flexibility in

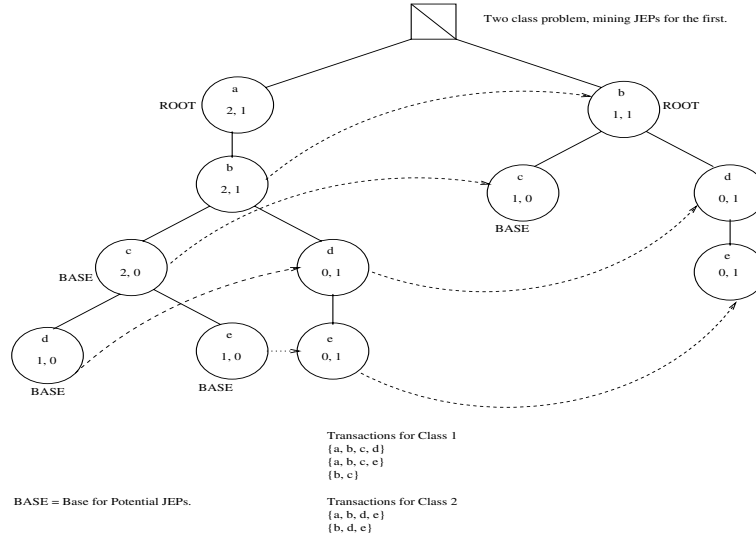
the inputs which are provided to the border-diff function. We have some control over the number of negative transactions, their cardinality and the cardinality of the positive transaction, all key determinants in performance. After identifying a potential JEP, we gather up all negative transactions that are related to it (i.e. share the root and base node). These negative transactions can be obtained using side links which join all nodes representing the same item. Border-diff is then invoked to identify all actual JEPs contained within the potential JEP. After examining all branches for a particular component tree, we re-insert them back into the remaining component trees, having removed the initial node of each. The following pseudo-code gives a high-level outline of the mining algorithm.

```

Component_Trees CTs = build_tree();
For each component tree, ct of CTs
  For each branch b of ct
    if(b is a potential JEP)
      border_diff(b, negative_transactions);
      relocate_branches(ct, CTs);

```

Example: Consider the following tree.



Beginning at the leftmost component tree (with root a) we traverse its children looking for base nodes for potential JEPs. In this case there are three, $\{c, d$ and $e\}$. On finding c we attempt to gather the associated negative transactions (with reference the root a and base c), in this case there exist no such transactions. $\{a, c\}$ is output as a JEP. On encountering base node d , we are able to collect the negative transaction $\{a, b, d\}$, border-diff is called with $\{\{a, b, c, d\}, \{a, b, d\}\}$. Finally on discovering e as a base node and collecting the associated negative transactions, we call border-diff with the input $\{\{a, b, c, e\}, \{a, b, d, e\}\}$. The component tree with a as the root has now been processed

and its transactions are re-inserted with the a element removed. Mining would then continue on this next component tree.

By examining potential JEPs only, we ensure that the only patterns we mine are JEPs. The fact that we examine every potential JEP with reference to every component tree (and thus every item in the problem), ensures completeness. In all cases the number of component trees is equal to the number of unique items in the database. Component tree traversal in the worst case requires visiting a number nodes equal to the number of attributes in the problem, per branch. Subsequent collation of all negative transactions, in the worst-case, requires gathering $|DB|-1$ transactions.

5 Performance of Tree Mining

The following table displays the performance of our JEP mining procedure using various types of tree orderings. The Original column refers to the implementation used to mine JEPs using previous published work. Times recorded are user times and are given in seconds. The column headed `hybrid`, represents a hybrid tree with the the first thirty percent of items ordered by ratio, the remainder by absolute frequency. The column labelled `speedup` compares the hybrid tree with the original approach. The data sets used were acquired from the UCI Machine Learning Repository [3]. All experiments were performed on a 500MHz Pentium III PC, with 512 MB of memory, running Linux (RedHat).

Dataset	MPPC-tree	FP-tree	LPNC-tree	iRatio-tree	Ratio-tree	Hybrid	Original	Speedup
mushroom	38.57	27.59	37.35	17.32	16.77	13.66	138.45	10.13
census	497.17	459.65	385.51	221.84	214.23	182.84	1028.00	5.62
ionosphere	83.63	69.75	59.04	21.91	21.15	19.07	86.74	4.55
vehicle	5.86	6.01	5.82	3.92	3.85	3.20	5.33	1.67
german	140.84	138.46	94.61	50.75	47.54	38.59	131.37	3.40
segment	68.16	66.86	65.65	45.52	41.42	35.60	71.99	2.02
hypothyroid	3.65	3.03	3.84	1.99	1.89	1.77	1.79	1.01
pendigits	687.32	719.09	631.62	563.05	560.92	507.55	2951.26	5.81
letter-rec	3632.92	3537.37	3199.32	1815.82	1700.91	1485.20	6896.07	4.64
soybean-l	321.28	490.70	457.45	135.46	127.03	74.15	611.50	8.25
waveform	4382.27	4391.85	2814.14	2794.07	2779.66	2560.16	26180.50	10.23
chess	811.51	871.91	865.30	245.96	238.95	90.62	358.62	3.96

We see that using our tree mining algorithm, significant savings are achieved over the original method. We now rank the various types of trees:

1. Hybrid tree - always the fastest performer. The parameter α was set to 30% (we conducted other experiments for alternative values, with this choice giving the best overall times). The performance gains are typically around 5 times faster than the original method.
2. Ratio and inverse ratio trees.

3. LPNC tree.
4. Frequent pattern tree and MPPC tree. Each with similar running times.
5. Original method of [6]. The slowest technique serving as a benchmark for the tree-based methods.

We can make a number of observations about these results:

1. The relative memory usage among the various trees will of course vary between the various datasets. However, from additional data not included here due to lack of space, the ranking (from least average tree size to largest average tree size) is i) Frequent pattern tree, ii) MPPC tree, iii) Hybrid tree (when using $\alpha = 30\%$), iv) Ratio and inverse ratio trees, v) LPNC tree. The frequent pattern tree uses the least memory, but takes the longest time to mine. This would indicate that tree size is not a dominant factor in determining the mining effort needed for JEPs. This is in contrast to the work in [9], where the main objective in using frequent pattern trees was to reduce tree size, so that frequent itemset calculation could be carried out entirely within main memory.
2. The LPNC and MPPC trees are consistently worse than the ratio tree variants. The orderings for these trees only consider one of the positive and negative datasets and thus there is less overlap between positive and negative transactions. Consequently more potential JEPs will need testing.
3. Ratio and inverse ratio trees are superior for mining than frequent pattern trees. We believe this is because ratio/inverse ratio tree structure results in fewer potential JEPs needing to be tested. As the component trees are processed according to the ratio order, singleton items which have high support in one class and low support in the other are pruned earlier. Such items are strong differentiators between the classes. Thus the tendency is for positive and negative transactions to have greater overlap as processing proceeds and hence fewer potential JEPs (especially duplicate ones) will be tested by border-diff.
4. The hybrid tree is faster to mine than the pure ratio tree. We conjecture that frequent trees allow items which have high support in both classes to be pruned earlier. This in turn means that there will be fewer and fewer transactions per component tree as processing proceeds, also decreasing the number of required border-diff calls. Combining this property of frequent pattern trees with the properties of ratio trees, results in a tree that is very fast to mine. It is the subject of further research to more deeply analyse the interplay of factors here.

Overall, these tree based methods are significant improvements on previous methods for mining emerging patterns. Nevertheless, for datasets of very high dimensionality, the running time of a complete mine may still be prohibitive. This motivates a supplementary approach which mines only a subset of the complete set of JEPs.

6 Mining the Highest Support JEPs Using Thresholds

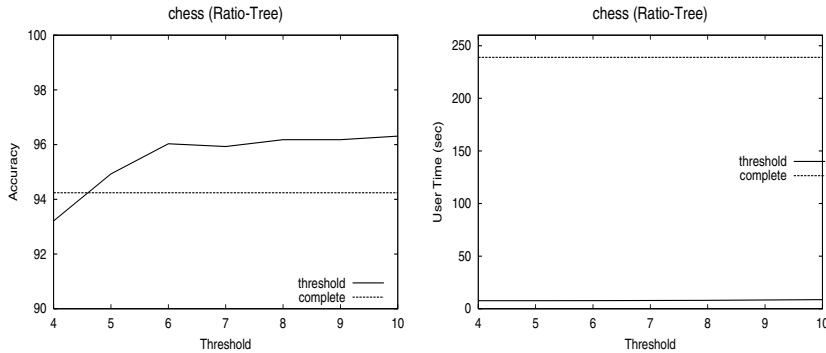
We now examine a method which sacrifices completeness of JEP mining in return for faster computation.

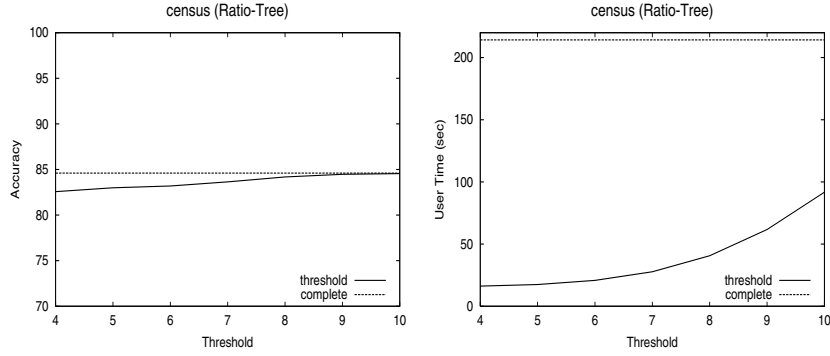
Since completeness will no longer hold, we wish the JEPs we mine to be “important ones”, i.e. they should have high support. Examining the characteristics of JEPs of high support, it should be clear that in general shorter JEPs will experience greater support levels than longer JEPs.

We therefore aim to mine as many of the short JEPs as possible. Our mining procedure is now modified to only identify potential JEPs whose length is below a specified threshold. Any potential JEPs above this threshold will not be examined by the *border-diff* function to see if actual JEPs are present. Whilst this method will not necessarily ensure mining of only the highest support JEPs, it presents an attractive alternative due to the relatively small computation time and its algorithmic simplicity.

Applying such thresholds means that the number of times the *border-diff* function is called is drastically reduced, as well as ensuring that when used it is not too expensive, since we have complete control over one of the factors, the cardinality of the itemsets passed to it.

The success of such a strategy is dependent upon how many short JEPs actually reside within the threshold one chooses to impose. Sometimes application of the threshold may mean that some short JEPs may be lost. e.g. A potential JEP $J = \{a, b, c, d, e, f, g, h, i, j\}$ in a ten attribute problem (where a is the root item and j is the **base** item) may actually contain the following JEPs; $\{a, b, j\}$, $\{a, c, j\}$ and $\{a, e, j\}$. However, choosing a threshold of four for this example would eliminate the possibility of discovering these JEPs. The following diagrams now illustrate the merit of various threshold values applied to a ratio tree. The four graphs illustrate the variance of accuracy and user time versus threshold for two datasets. The accuracy and time of JEP mining without thresholds (complete mining) is provided as a reference. For these examples we can see that as thresholds increase, accuracy converges relatively quickly and user time increases relatively slowly.





Dataset	pt=4	pt=5	pt=6	pt=7	pt=8	pt=9	pt=10	original
mushroom	6.03	6.11	6.28	6.48	6.82	7.38	8.19	138.45
census	16.23	17.46	20.78	27.75	40.61	61.71	91.75	1028.00
ionosphere	1.37	1.43	1.45	1.56	1.67	1.83	1.99	86.74
vehicle	0.96	0.99	1.00	1.07	1.19	1.33	1.50	5.33
german	1.53	1.64	1.86	2.28	2.93	3.86	5.19	131.37
segment	8.44	8.86	9.71	11.08	12.92	15.34	18.15	71.99
hypothyroid	1.16	1.21	1.22	1.25	1.27	1.30	1.35	1.79
pendigits	63.25	72.53	88.32	111.37	142.30	181.77	230.28	2951.26
letter-rec	249.14	256.49	275.44	319.41	396.38	510.47	659.98	6896.07
soybean-l	13.67	13.68	13.69	13.75	13.83	14.09	14.24	611.50
waveform	18.09	21.31	30.53	47.13	72.71	110.47	165.44	26180.50
chess	7.64	7.66	7.75	7.83	8.02	8.33	8.68	358.62

timing-pure thresholding (ratio-tree)

Dataset	pt=4	pt=5	pt=6	pt=7	pt=8	pt=9	pt=10	complete
mushroom	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
census	82.57	82.99	83.19	83.64	84.18	84.47	84.55	84.60
ionosphere	90.32	90.27	91.11	90.83	91.13	90.85	90.85	88.32
vehicle	46.23	48.35	49.66	52.27	53.10	55.93	58.52	65.11
german	71.50	72.10	72.90	73.70	73.90	74.40	74.20	74.70
segment	94.67	94.80	94.76	94.98	94.89	94.46	94.59	93.81
hypothyroid	97.60	97.72	97.76	97.76	97.76	97.76	97.88	98.48
pendigits	93.17	94.72	95.30	95.57	95.72	95.92	95.95	96.16
letter-rec	63.07	76.90	82.12	84.34	85.28	86.63	87.85	92.21
soybean-l	82.73	83.43	83.92	85.66	84.71	84.88	83.06	84.92
waveform	70.48	77.62	79.38	80.12	80.16	80.32	80.44	82.94
chess	93.21	94.93	96.03	95.93	96.18	96.18	96.31	94.24

accuracy-pure thresholding (ratio-tree)

The two tables above provide more complete information on mining behaviour using thresholds. We see that mining with a threshold value of 4 is

substantially faster than mining the complete set of JEPs using a ratio tree. Classification accuracy is degraded for three of the datasets (Vehicle, Waveform and Letter-recognition) though. Analysis of the vehicle and chess datasets aid in explaining this outcome (supporting figures have been excluded due to lack of space). It is clear that classification accuracy is dependent upon finding patterns that strongly discriminate and at the same time are strongly representative of the instances of a particular class. The number of patterns one finds can be viewed as an indicator of how well a class' instances can be differentiated from instances of another class. The importance of each pattern, as a representative of the class, can be measured as its support. The discrepancy in classification accuracy of the vehicle dataset, from a threshold of 4 to 10, may be accounted for by a large difference in the number of patterns found for two of its classes (*saab* and *opel*) between threshold 4 and threshold 10. The average support of patterns is roughly the same at each of these threshold values. In contrast, for the chess dataset, we don't experience such marked fluctuation in classification over the thresholds, since the balance between number of JEPs and their average support is kept constant as threshold value increases. A threshold of 4 for chess has fewer number of JEPs, but their average support is greater, while a threshold of 10, has lower average support JEPs, but possesses a greater number of them. Clearly both factors are important and further work is required to determine their precise relationship(s).

Isolating behaviour with a threshold value of 10, we see that the improvement in mining time is not as great, but still substantial (around 2-10 times faster than mining the complete set with a ratio tree, and around 2-158 times faster than the original method). Classification accuracy is also the same (only about 1% difference) as classification using the full set of JEPs.

Adopting a threshold of 10 then, is a useful means of speeding up mining without appreciable loss of precision. For further work, it would be interesting to see whether it is possible to automatically choose different thresholds according to the characteristics of the input datasets or to develop more complex thresholding criteria.

7 Summary and Future Work

In this paper we have developed efficient algorithms to mine emerging patterns. We presented a mining algorithm that used tree data structures to explicitly target the likely distribution of JEPs. This achieved considerable performance gains over previous approaches. We also looked at methods for computing a subset of the possible JEPs, corresponding to those with the highest support in the dataset. These approximate methods achieved additional performance gains, while still attaining competitive precision. For future work we intend to:

- Extend our techniques to handle finite growth rate emerging patterns.
- Investigate further ways of ordering trees and investigate whether methods that have been developed in other machine learning contexts (e.g. for ranking

attributes or splitting in decision trees) can help.

- Develop analytical justification for the hybrid tree's performance.

Acknowledgements

This work was supported in part by an Expertise Grant from the Victorian Partnership for Advanced Computing.

References

1. R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In Proceedings of the Twentieth International Conference on Very Large Data Bases, Santiago, Chile, 1994. p. 487-499. 40
2. Bayardo, R. J. Efficiently Mining Long Patterns from Databases. SIGMOD 1998. 41
3. C. L. Blake and P. M. Murphy. UCI Repository of machine learning [www.ics.uci.edu/~mlearn/MLRepository.html]. 45
4. C. V. Cormack, C. R. Palmer and C. L. A. Clarke. Efficient construction of large test collections. In Proceedings of the Twenty-first Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Melbourne, Australia, 1998. p. 282-289.
5. De Raedt, L., Kramer, S. The Level-Wise Version Space Algorithm and its Application to Molecular Fragment Finding. (IJCAI-01), 2001. 41
6. G. Dong and J. Li. Efficient mining of emerging patterns: Discovering trends and differences. In Proceedings of the fifth International Conference on Knowledge Discovery and Data Mining, San Diego, USA, (SIGKDD'99), 1999, p.43-52. 39, 40, 41, 43, 46
7. Dong, G., Li, J. and Zhang, X. Discovering Jumping Emerging Patterns and Experiments on Real Datasets. (IDC99), 1999. 41
8. Fan, H. and Ramamohanarao, K. An efficient Single-scan Algorithm for mining Essential Jumping Emerging Patterns for Classification Accepted at PAKDD-2002, Taipei, May6-8, Taiwan.C. 40
9. J. Han, J. Pei, Y. Yin. Mining frequent patterns without candidate generation. In Proceedings of the International Conference on Management of Data, Dallas, Texas, USA (ACM SIGMOD), 2000. p. 1-12. 40, 42, 46
10. Kramer, S., De Raedt, L., Helma, C. Molecular Feature Mining in HIV Data. ACM SIGKDD (KDD-01), 2001. 41
11. J. Li, G. Dong and K. Ramamohanarao. Making use of the most expressive jumping emerging patterns for classification. In Proceedings of the Fourth Pacific-Asia Conference on Knowledge Discovery and Data Mining, Kyoto, Japan, 2000. p. 220-232. 39, 40, 41, 42
12. J. Li and L. Wong. Emerging patterns and Gene Expression Data. In proceedings of 12th Workshop on Genome Informatics. Japan. December 2001, pages 3-13. 39
13. Mannila, H. and Toivonen, H. Levelwise Search and Borders of Theories in Knowledge Discovery. Data Mining and Knowledge Discovery 1(3), 1997. 41
14. T. M. Mitchell. Generalization as search. Artificial Intelligence, 18, 203-226, 1982. 40
15. Pasquier, N., Bastide, R., Taouil, R. and Lakhal, L. Efficient Mining of Association Rules using Closed Itemset Lattices. Information Systems 24(1), 1999. 41
16. J. R. Quinlan: C4.5 Programs for Machine Learning. Morgan Kaufmann, 1993. 40

On the Discovery of Weak Periodicities in Large Time Series

Christos Berberidis¹, Ioannis Vlahavas¹, Walid G. Aref²,
Mikhail Atallah^{2,*}, and Ahmed K. Elmagarmid²

¹Department of Informatics, Aristotle University of Thessaloniki
54006 Thessaloniki Greece
{berber, vlahavas}@csd.auth.gr

²Dept. of Computer Sciences, Purdue University
{aref, mja, ake}@cs.purdue.edu

Abstract. The search for weak periodic signals in time series data is an active topic of research. Given the fact that rarely a real world dataset is perfectly periodic, this paper approaches this problem in terms of data mining, trying to discover weak periodic signals in time series databases, when no period length is known in advance. In existing time series mining algorithms, the period length is user-specified. We propose an algorithm for finding approximate periodicities in large time series data, utilizing autocorrelation function and FFT. This algorithm is an extension to the partial periodicity detection algorithm presented in a previous paper of ours. We provide some mathematical background as well as experimental results.

1 Introduction

Periodicity is a particularly interesting feature that could be used for understanding time series data and predicting future trends. However, little attention has been paid on the study of the periodic behavior of a temporal attribute. In real world data, rarely a pattern is perfectly periodic (according to the strict mathematical definition of periodicity) and therefore an almost periodic pattern can be considered as periodic with some confidence measure. Partial periodic patterns are patterns that are periodic over some but not all the points in it. An interesting extension of the problem of capturing all kinds of periodicities that might occur in real world time series data is the discovery of approximate periodicities. That is, periodicities where a small number of occurrences are not 100% punctual.

Early work in time-series data mining addresses the pattern-matching problem. Agrawal et al. in the early 90's developed algorithms for pattern matching and simi-

* Portions of this work were supported by Grant EIA-9903545 from the National Science Foundation, Contract N00014-02-1-0364 from the Office of Naval Research, and by sponsors of the Center for Education and Research in Information Assurance and Security.

larity search in time series databases [1, 2, 3]. Mannila et al. [4] introduce an efficient solution to the discovery of frequent patterns in a sequence database. Chan et al. [5] study the use of wavelets in time series matching and Faloutsos et al. in [6] and Keogh et al. in [7] propose indexing methods for fast sequence matching using R^* trees, the Discrete Fourier Transform and the Discrete Wavelet Transform. Toroslu et al. [8] introduce the problem of mining cyclically repeated patterns. Han et al. [9] introduce the concept of partial periodic patterns and propose a data structure called the Max Subpattern Tree for finding partial periodic patterns in a time series. Aref et al. in [10] extend this work by introducing algorithms for incremental, on-line and merge mining of partial periodic patterns.

The algorithms proposed in the above articles, discover periodic patterns for a user-defined period length. If the period length is not known in advance, then these algorithms are not directly applicable. One would have to exhaustively apply them for each possible period length, which is impractical. In other words, it is assumed that the period is known in advance thus making the process essentially ad-hoc, since unsuspected periodicities will be missed. Berberidis et al. [13] propose an algorithm for detecting the period when searching for multiple and partial periodic patterns in large time series.

In this paper we attempt to detect weak periodic signals in large, real world time series. By “weak periodic signals” we mean partial and approximate periodicities. We introduce the notion of approximate periodicities, which is the case when some periodic instances of a symbol might be appearing a number of time points before or after their expected periodic occurrence. Our work extends the algorithm introduced in [13], for discovering multiple and partial periodicities, without any previous knowledge of the nature of the data. We use discretization to reduce the cardinality of our data. The time series is divided into a number of intervals and a letter is assigned to each interval. Thus, the original time series is transformed into a character sequence. The algorithm follows a filter-refine paradigm. In the filter step, the algorithm utilizes the *Fast Fourier Transform* to compute a *Circular Autocorrelation Function* that provides us with a conservative set of candidate period lengths for every letter in the alphabet of our time series. In the refine step, we apply Han’s algorithm [9] for each candidate period length. The complexity of our algorithm is $O(AN\log N)$, where A is the size of the alphabet and N the size of the time series. The algorithm speeds up linearly both to the number of time points and the size of the alphabet.

The rest of this paper proceeds as follows: the next section contains notation and definitions for the problem. In section 3 we outline the steps of the algorithm we propose for discovering partial periodicities and we explain how it works in detail. We provide some theoretical background and we discuss the computational complexity of the algorithm. We test our algorithm with various data sets, produce some experimental results and verify them using Han’s algorithm. In section 4 we discuss an extension to the partial periodicity algorithm of section 3, for finding approximate periodicities. In the last section we conclude this paper and suggest some directions for further research.

2 Notation

A **pattern** is a string $s = s_1 \dots s_p$ over an **alphabet** $L \cup \{*\}$, where the letter $*$ stands for *any single symbol from L* . A pattern $s' = s'_1 \dots s'_p$ is a **subpattern** of another pattern s if for each position i , $s'_i = s_i$ or $s'_i = *$. For example, $ab*d$ is a subpattern of $abcd$.

Assume that a pattern is periodic in a time series S of length N with a **period** of length p . Then, S can be divided into $\lfloor N/p \rfloor$ segments of size p . These segments are called

periodic segments. The **frequency count** of a pattern is the number of the periodic segments of the time series that match this pattern. The **confidence** of a pattern is defined as the division of its frequency count by the number of period segments in the time series ($\lfloor N/p \rfloor$). For example, in the series $abcdabddabfccba$, the pattern $ab**$ is periodic with a period length of 4, a frequency count of 3, and a confidence of $3/4$.

According to the **Apriori property on periodicity** discussed in [9] “each subpattern of a frequent pattern of period p is itself a frequent pattern of period p ”. For example, assume that $ab**$ is a periodic pattern with a period of 4, then $a***$ and $*b**$ are also periodic with the same period. Conversely, knowing that $a***$ and $*b**$ are periodic with period 4 does not necessarily imply that $ab**$ is periodic with period 4.

3 Discovering Partial Periodicities – The PPD Algorithm

Based on the Apriori property described in the previous section, we present the algorithm we proposed in [13], that generates a set of candidate periods for the symbols of a time series. We call this algorithm PPD, which stands for Partial Periodicity Detection.

The *filter/refine paradigm* is a technique that has been used in several contexts, e.g., in spatial query processing [11]. The filter phase reduces the search space by eliminating those objects that are unlikely to contribute to the final solution. The refine phase, which is CPU-intensive, involves testing the candidate set produced at the filter step in order to verify which objects fulfill the query condition.

The filter/refine paradigm can be applied in various search problems such as the search for periodicity in a time series. We use the *circular autocorrelation function* as a tool to filter out those periods that are definitely not valid.

We outline the major steps performed by our algorithm. The explanation of the steps is given further down in this section.

- Scan the time series once and create a binary vector of size N for every symbol in the alphabet of the time series.
- For each symbol of the alphabet, compute the circular autocorrelation function vector over the corresponding binary vector. This operation results in an output autocorrelation vector that contains frequency counts.
- Scan only half the autocorrelation vector (maximum possible period is $N/2$) and filter out those values that do not satisfy the minimum confidence threshold and keep the rest as candidate periods.

- Apply Han's algorithm to discover periodic patterns for the candidate periods produced in the previous step.

Steps 1—3 correspond to the filter phase while Step 4 corresponds to the refine phase, which uses Han's Max-subpattern Hit Set Algorithm that mines for partial periodic patterns in a time series database. It builds a tree, called the Max-Subpattern tree, whose nodes represent a candidate frequent pattern for the time series. Each node has a count value that reflects the number of occurrences of the pattern represented by this node in the entire time series. For brevity, we refer the reader to [9] for further details.

3.1 The Filter Step

The first step of our method is the creation of a number of binary vectors. Assume we have a time series of size N . We create a binary vector of size N for every letter in our alphabet. An ace will be present for every occurrence of the corresponding letter and a zero for every other letter.

The next step is to calculate the Circular Autocorrelation Function for every binary vector. The term autocorrelation means self-correlation, i.e., discovering correlations among the elements of the same vector. We use Autocorrelation as a tool to discover estimates for every possible period length.

The computation of autocorrelation function is the sum of N dot products between the original signal and itself shifted every time by a lag k . In *circular* autocorrelation, one point, at the end of the series, is shifted out of the product in every step and is moved to the beginning of the shifting vector. Hence in every step we compute the following dot product, for all N points:

$$r(k) = \frac{1}{N} \sum_{x=1}^N f(x)f(x+k) \quad (1)$$

This convolution-like formula calculates the discrete 1D circular autocorrelation function for a lag k . For our purposes we need to calculate the value of this function for every lag, that is for N lags. Therefore, (1) is computed for all $k=1 \dots N$. The complexity of this operation is $O(N^2)$, which is quite expensive, especially when dealing with very large time series. Utilizing the Fast Fourier Transform (FFT) effectively reduces the cost down to $O(N \log N)$, as follows:

$$f(x) \xrightarrow{FFT} F(x) \longrightarrow R(F(x)) = \frac{1}{N} F(x) * \bar{F}(x) \xrightarrow{IFFT} r(f(x)) \quad (2)$$

In the above formula $F(x) * \bar{F}(x)$ is the dot product of $F(x)$ with its complex conjugate. The mathematical proof can be found in the bibliography.

Example 1: Consider the series *abcdabebadfcacdcfcaa* of length 20, where *a* is periodic with a period of 4 and a confidence of 3/4. We create the binary vector 10001000100010000011. The autocorrelation of this vector is given in Figure 1.

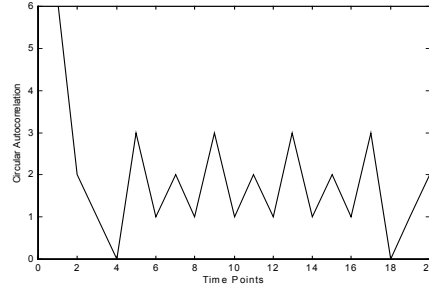


Fig. 1. Circular Autocorrelation Function when the length is a multiple of the period

The first value of the autocorrelation vector is the dot product of the binary vector with itself, since the shifting lag is 0 and therefore the two vectors align perfectly. Thus, the resulting value is the total number of aces, which is the total number of occurrences of the letter *a*. The peak identified in the above chart at position 5 implies that there is probably a period of length 4 and the value of 3 at this position is an estimate of the frequency count of this period. According to this observation, we can extract those peaks, hence acquiring a set of candidate periods. Notice that a period of length 4 also results in peaks at positions 5, 9, 13 etc.

The user can specify a minimum confidence threshold c and the algorithm will simply extract those autocorrelation values that are greater than or equal to cN/p , where p is the current position where a period could exist.

One of the most important issues one has to overcome when dealing with real world data is the inevitable presence of noise. The computation of the autocorrelation function over binary vectors eliminates a large number of non-periodic aces due to their multiplication with zeroes, and hence leaving the periodic aces basically to contribute to the resulting value. Otherwise, using autocorrelation over the original signal, would cause all the non-periodic instances to contribute into a totally unreliable score estimate. Consequently, this value could be an acceptable estimate of the frequency count of a period. Note that the value of the estimate can never be smaller than the real one. Therefore, all the valid periodicities will be included in the candidate set together with a number of false ones that are the effect of the accumulation of random, non-periodic occurrences with the periodic ones.

One major weakness of the circular autocorrelation is that when the length of the series is not an integer multiple of the period, the circularly shifting mechanism results in vectors with a higher occurrence of unexpected values. This is usually increased by the randomness of real world data and the presence of noise. In our example the length of the series is $N=20$, which is an integer multiple of the period $p=4$. When the length of the series is 21 (e.g., by adding a zero at the end of the binary vector), this results in the circular autocorrelation given in Figure 2.

Another problem could arise when a number of successive occurrences of a letter are repeated periodically. For example the periodic repetition of aa^* would result in an unusually high autocorrelation value. Consider the series $aabaacaadacdbdbdabc$, where aa^* is repeated in 3 out of 6 periodic segments, while a^{**} is repeated in 4 periodic segments. The circular autocorrelation chart for the symbol *a* is given in

Figure 2b. A clear peak at position 4 can be seen, implying the existence of a period of 3. The frequency estimate according to the autocorrelation function is 6, which happens to be two times the actual frequency count, which is 3.

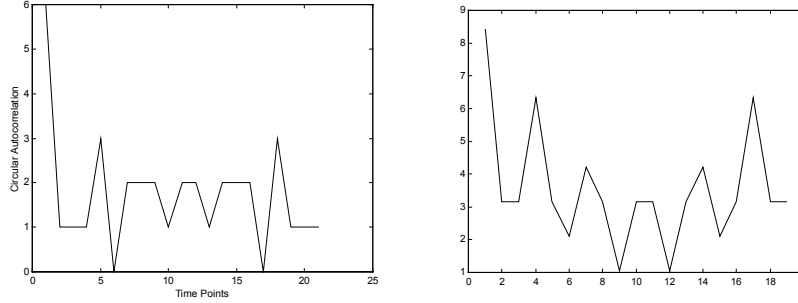


Fig. 2. (a) Circular Autocorrelation Function when the length is not a multiple of the period. (b) Circular Autocorrelation Function when successive occurrences of a letter are repeated periodically

Repeating the algorithm described so far, for every symbol in the alphabet of our time series will result in a set of possible periods for each one of them. Note that a letter can have more than one period. For every candidate period, there will be an estimate of its confidence, according to their autocorrelation value. Utilizing the Apriori property on periodicity discussed earlier in this article, we can create periodicity groups, that is, groups of letters that have the same period. Han's algorithm [9] can be applied to verify the valid periods and extract the periodic patterns.

Theorem: Consider a time series with N points. Also let a letter of that time series feature periodicity with a period p_1 with a confidence c_1 . We can prove that this letter is also periodic with a period of p_2 and confidence $c_2 \geq c_1$, when p_2 is a multiple of p_1 .

For example, if a is periodic with a period length of 4 and a confidence of 75% then it is also periodic with a period of 8, 12, 16 etc. and the corresponding confidence measures are equal to or greater than 0.75. Assume that b is periodic with a period of 8. Based on the previous theorem we know that a is also periodic with a period of 8 and therefore, we can create a periodicity group consisting of those two letters and apply Han's algorithm to check whether there is a periodic pattern with a period of 8 or any of its multiples.

3.2 Analysis

Our algorithm requires 1 scan over the database in order for the binary vectors to be created. Then it runs in $O(N \log N)$ time for every letter in the alphabet of the series. Consequently the total run time depends on the size of the alphabet. Generally speaking we can say that this number is usually relatively small since it is a number of *user specified classes* in order to divide a range of continuous values. Despite the fact that some non-periodic peaks might occur, the method we propose is complete since all valid periods are extracted.

3.3 Experimental Results

We tested our algorithm over a number of data sets. The most interesting data sets we used were supermarket and power consumption data. The former contain sanitized data of timed sales transactions for some Wal-Mart stores over a period of 15 months. The latter contain power consumption rates of some customers over a period of one year and were made available through a funded project. Synthetic control data taken from the Machine Learning Repository [12] were also used. Different runs over different portions of the data sets showed that the execution time is linearly proportional to the size of the time series as well as the size of the alphabet. Figure 3a shows the behavior of the algorithm against the number of the time points in the time series.

Figure 3b shows that the algorithm speeds up linearly to alphabets of different size. The size of the alphabet implies the number FFT computations of size N required. The times shown on the chart below correspond to a synthetic control data set of $N = 524288$ time points.

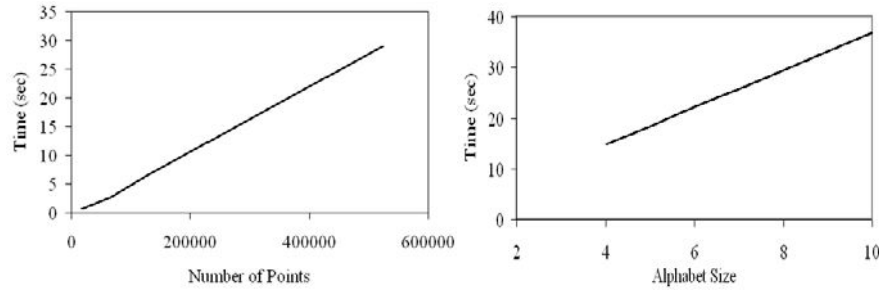


Fig. 3. Run time against data sets of different size

Experiments have confirmed our expectation regarding the completeness of PPD. In three datasets containing the number of customers per hour in three Wal-Mart stores, the algorithm returned the period that is most likely to be correct. Alternatively, instead of searching for a single candidate period, we could mine for a larger set of candidates. Table 1a summarizes the results. The “ACF” column is the Autocorrelation estimate produced for the periodic occurrences of a letter, while the “Freq.” column is the number of occurrences of each letter. Notice that for most letters in all three datasets the suggested period is 24 or a multiple of it (e.g. 168, 336). Table 1b contains the patterns produced by Han’s algorithm for a period length of 24.

4 Capturing Approximate Periodicities – The APPD Algorithm

We define **approximate periodicity** as a periodicity, some periodic instances of which, might be shifted a user-limited number of time points before or after their expected periodic occurrence. Normally, these instances would be considered missing and therefore this periodicity would be considered partial. Capturing those instances is a particularly interesting task that provides us with useful information regarding the strength of a periodicity. We try to capture those “shifted” occurrences in terms of

frequency estimate. In other words, we use the autocorrelation function over the binary vectors of the occurrences of a letter, as a means in order to acquire a reliable measure of the strength of a periodicity. We call our algorithm APPD, which stands for Approximate Periodicity Detection.

Table 1. (a) Results for the Wal-Mart stores. (b) Verification with Han’s algorithm

(a)					(b)	
Data	Symbols	Period	ACF	Freq.	Pattern	Conf.
Store 1	A	24	228	3532	AAAAAABBBB*****B*A	62.4
	B	168	1140	2272	AAAAAA**BB*****AA	72.6
	C	24	94	1774	AAAAAA***BC*****AA	60.9
	D	336	648	874	AAAAAA***B*****AA	75.7
	E	504	2782	2492	AAAAAA*BB*****BAA	63.3
	F	4105	81	48	AAAAAA*BBB*****AA	60.9
Store 2	A	24	252	3760	AAAAAABBB*****BAA	61.3
	B	168	1750	2872	AAAAAABBB*****B*A	69.6
	C	168	936	2199	AAAAAABBB*****AA	65.7
	D	168	851	2093		
	E	1176	90	140		
Store 3	A	168	2034	3920		
	B	168	1436	2331		
	C	168	950	2305		
	D	336	434	655		
	E	24	99	1830		
	F	-	-	23		

Our approach is an extension to PPD. At the preprocessing stage, we assume that all the occurrences of a letter could be part of a periodicity and that they might be shifted. Every such occurrence is represented in a binary vector by an ace. By replacing zeroes around every ace with values in the range between 0 and 1, we attempt to capture all these possible shiftings. Consider the following example:

Example 3. Given the following binary vector of the occurrences of a letter in a time series: $u = [1,0,0,0,1,0,0,0,1,0,0,0,1,0,0,0,1,0,0,0,1,0,0,0,1,0,0,0,1,0,0,0,1,0,0,0,1,0,0,0,1,0,0,0]$, consisting of 44 points and featuring a perfect periodicity with period length 4, we shift the 3 last ones by 1 position before or after (arbitrarily), thus taking the following vector: $v = [1,0,0,0,1,0,0,0,1,0,0,0,1,0,0,0,1,0,0,0,1,0,0,0,1,0,0,0,1,0,0,0,1,0,0,0,1,0,0,0,1,0,0,0]$. The autocorrelation function of vectors u and v are shown in the following figures.

The autocorrelation value of 11 at position 5 of the first vector implies a periodicity of length 4. Shifting 3 aces by 1 position, results in an autocorrelation value of 8 at position 5. Thus, those 3 aces were not considered at all. In real world data, where randomness and noise is always present, such effects are usually expected, while perfectly distributed periodic instances are quite unlikely to occur. Changing the two zeroes, before and after every ace, to 0.5 we make them contribute to the accuracy of the estimate of the periodicity, implying thus that there is a 50% probability that every ace's natural position might be the one before or the one after.

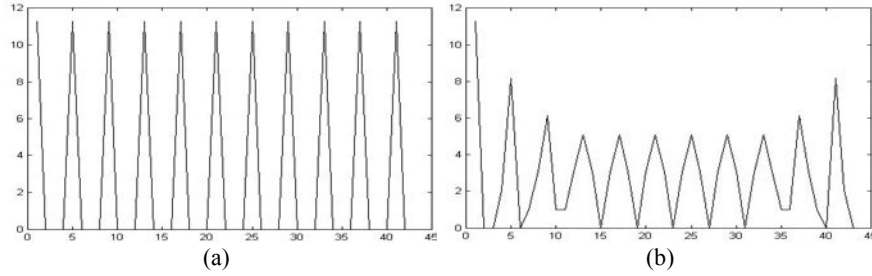


Fig. 4. Autocorrelation of vectors u and v

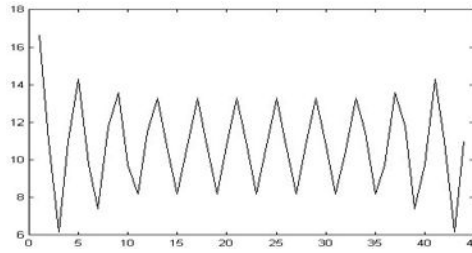


Fig. 5. Autocorrelation of vector w

The above chart shows that the autocorrelation value at position 5 is now 14.3, denoting that the implied periodicity might actually be stronger than the one implied by the autocorrelation of v . Additionally, we can insert values other than 0.5 before and after the aces, depending whether one wants to increase the probability, and therefore the contribution, of the possibly shifted aces. It is totally up to the user or the domain expert to alter this according to his knowledge about the nature of the data. Furthermore, one can also increase the area around every ace to be covered with values between 0 and 1. Replacing zeroes around an ace like $[0.2, 0.6, 1, 0.6, 0.2]$ would be similar to using a triangular membership function in a fuzzification process. The main advantage is that the computational cost of our approach is much smaller than the one of a fuzzy algorithm.

Finally, we should make clear that the estimate provided by APPD is a reliable indication of the strength of a periodicity, and not a frequency estimate, like the one produced by PPD. It is not evidence but a serious hint that could provide the user with useful insight about the data. One should combine the two methods in order to mine for weak periodicities in a time series. If the increase of the autocorrelation value is significant then it is highly possible that its actual confidence is greater than the one produced by the first method.

APPD's computational complexity is exactly the same as PPD's. It engages at the preprocessing stage, during the first scan of the data, when the binary vectors are created. One can create both sets of vectors during the same scan and then run the autocorrelation step twice, avoiding thus another scan over the data on the disk.

5 Conclusions and Further Work

In this paper we presented a method for efficiently discovering a set of candidate periods in a large time series. Our algorithm can be used as a filter to discover the candidate periods without any previous knowledge of the data along with an acceptable estimate of the confidence of a candidate periodicity. It is useful when dealing with data whose period is not known or when mining for unexpected periodicities. Algorithms such as Han's described in [9] can be used to extract the periodic patterns. We tried our method against various data sets and it proved to speed up linearly against different alphabets and different numbers of time points. We also verified its expected completeness using Han's algorithm.

We also proposed a method for capturing approximate periodicities in a time series. Our method is an extension to the partial periodicity detection algorithm, at the preprocessing stage. We provide the user with a reliable strength measure for approximate periodicities. Its usefulness lies on the fact that in real world data several instances of a periodic pattern or symbol, might not be accurately distributed over the time series. It adds no computational overhead to the previous algorithm, since it can be integrated into the first scan of the data, at the preprocessing stage.

We implemented and tested our algorithm using a main memory FFT algorithm, however, a disk-based FFT algorithm [14, 15] would be more appropriate for handling larger time series that do not fit in the main memory. Interesting extension of our work would be the development of an algorithm to perform over other kinds of temporal data such as distributed.

6 References

1. R. Agrawal, C. Faloutsos, and A. Swami, Efficient Similarity Search in Sequence Databases. In *Proc. of the 4th Int. Conf. on Foundations of Data Organization and Algorithms*, Chicago, Illinois, October 1993.
2. R. Agrawal, K. Lin, H. S. Sawhney, and K. Shim. Fast Similarity Search in the Presence of Noise, Scaling, and Translation in Time-Series Databases. In *Proc. of the 21st Int. Conf. on Very Large Databases*, Zurich, Switzerland, September 1995.
3. R. Agrawal and R. Srikant. Mining Sequential Patterns. In *Proc. of 1995 Int. Conf. on Data Engineering*, Taipei, Taiwan, March 1995.
4. H. Mannila, H. Toivonen, and A. I. Verkamo. Discovering Frequent Episodes in Sequences. In *Proc. of the 1st Int. Conf. on Knowledge Discovery and Data Mining*, Montreal, Canada, August 1995.
5. K. Chan and A. Fu. Efficient Time-Series Matching by Wavelets. In *Proc. of 1999 Int. Conf. on Data Engineering*, Sydney, Australia, March 1999.
6. C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. Fast Subsequence Matching in Time-Series Databases. In *Proc. of the 1994 ACM SIGMOD Int. Conf. on Management of Data*, Minneapolis, Minnesota, May 1994.

7. E. Keogh, K. Chakrabarti, M. Pazzani and S. Mehrotra. Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases. *Springer-Verlag, Knowledge and Information Systems* (2001) p. 263–286.
8. H. Toroslu and M. Kantarcioglu. Mining Cyclically Repeated Patterns. *Springer Lecture Notes in Computer Science* 2114, p. 83 ff., 2001.
9. J. Han, G. Dong, and Y. Yin. Efficient Mining of Partial Periodic Patterns in Time Series Databases. In *Proc. of 1999 Int. Conf. on Data Engineering*, Sydney, Australia, March 1999.
10. W. G. Aref, M. G. Elfeky and A. K. Elmagarmid. Incremental, Online and Merge Mining of Partial Periodic Patterns in Time-Series Databases. Submitted for journal publication. Purdue Technical Report, 2001.
11. Orenstein J. A. Redundancy in Spatial Databases, *Proc. ACM SIGMOD Int. Conf. on Management of Data*, Portland, USA, 1989, pp. 294-305.
12. Blake, C. L. & Merz, C. J. (1998) UCI Repository of Machine Learning Databases, University of California, Department of Information and Computer Science.
13. Christos Berberidis, Aref G. Walid, Mikhail Atallah, Ioannis Vlahavas, Ahmed K. Elmagarmid, “Multiple and Partial Periodicity Mining in Time Series Databases”, F. van Harmelen (ed.): ECAI 2002. Proceedings of the 15th European Conference on Artificial Intelligence, IOS Press, Amsterdam, 2002.
14. Numerical Recipes in C: The Art of Scientific Computing. External Storage or Memory-Local FFTs. pp 532-536. Copyright 1988-1992 by Cambridge University Press.
15. J. S. Vitter. External Memory Algorithms and Data Structures: Dealing with Massive Data. *ACM Computing Surveys*, Vol. 33, No. 2, June 2001.

The Need for Low Bias Algorithms in Classification Learning from Large Data Sets

Damien Brain and Geoffrey I. Webb

School of Computing and Mathematics, Deakin University Geelong
Victoria, 3217, Australia
{dbrain, webb}@deakin.edu.au

Abstract. This paper reviews the appropriateness for application to large data sets of standard machine learning algorithms, which were mainly developed in the context of small data sets. Sampling and parallelisation have proved useful means for reducing computation time when learning from large data sets. However, such methods assume that algorithms that were designed for use with what are now considered small data sets are also fundamentally suitable for large data sets. It is plausible that optimal learning from large data sets requires a different type of algorithm to optimal learning from small data sets. This paper investigates one respect in which data set size may affect the requirements of a learning algorithm — the bias plus variance decomposition of classification error. Experiments show that learning from large data sets may be more effective when using an algorithm that places greater emphasis on bias management, rather than variance management.

1 Introduction

Most approaches to dealing with large data sets within a classification learning paradigm attempt to increase computational efficiency. Given the same amount of time, a more efficient algorithm can explore more of the hypothesis space than a less efficient algorithm. If the hypothesis space contains an optimal solution, a more efficient algorithm has a greater chance of finding that solution (assuming the hypothesis space cannot be exhaustively searched within a reasonable time). However, a more efficient algorithm results in more or faster search, not better search. If the learning biases of the algorithm are inappropriate, an increase in computational efficiency may not equate to an improvement in prediction performance.

A critical assumption underlies many current attempts to tackle large data sets by creating algorithms that are more efficient [1, 2, 3, 4, 5, 6]: that the learning biases of existing algorithms are suitable for use with large data sets. Increasing the efficiency of an algorithm assumes that the existing algorithm only requires more time, rather than a different method, to find an acceptable solution.

Since many popular algorithms (e.g. C4.5 [7], CART [8], neural networks [9], k-nearest neighbor [10]) were developed using what are now considered small data sets (hundreds to thousands of instances), it is possible that they are tailored to more effective learning from small data sets than large. It is possible that if data set sizes common today were common when these algorithms were developed, the evolution of such algorithms may have proceeded down a different path.

So far, few approaches to dealing with large data sets have attempted to create totally new algorithms designed specifically for today's data set sizes. In fact, many "new" algorithms are actually more efficient means of searching the hypothesis space while finding the same solution. Examples include RainForest [11], and ADTree [12]. It would seem logical that new algorithms, specifically designed for use with large data sets, are at least worth exploring.

This is not to say that efficiency is unimportant. There is little value in an algorithm that can produce miraculously good models, but takes so long to do so that the models are no longer useful. There must be a balance between efficiency and accuracy. This again lends support to the utility of algorithms fundamentally designed for processing large data sets.

The paper is set out as follows. Section 2 looks at issues relating to learning from large data sets. Section 3 details experiments performed and presents and discusses associated results. Conclusions and possible areas for further work are outlined in Section 4.

2 Learning from Large Data Sets

We have hypothesized that effective learning from large data sets may require different strategies to effective learning from small data sets. How then, might such strategies differ? This section examines multiple facets of this issue.

2.1 Efficiency

Certainly, efficiency is of fundamental importance. Small data set algorithms can afford to be of order $O(n^3)$ or higher, as with small data set sizes a model can still be formed in a reasonable time. When dealing with large data sets, however, algorithms of order $O(n^2)$ can be too computationally complex to be realistically useable. Therefore, algorithms for use with large data sets must have a low order of complexity, preferably no higher than $O(n)$.

It is worth noting that many algorithms can be made more efficient through parallelisation [13]. Splitting processing between multiple processors can be expected to reduce execution time, but only when splitting is possible. For example, the popular Boosting algorithms such as AdaBoost [14] and Arc-x4 [15] would seem prime candidates for parallelisation, as multiple models are produced. Unfortunately, this is not so, as the input of a model depends on the output of previous models (although parallelising the model building algorithms may still be possible). Bagging [16] and MultiBoost [17] are, on the other hand, suitable for parallelisation.

However, techniques such as parallelisation (or, for that matter, sampling) only reduce execution time. They do not make an algorithm fundamentally more suitable for large data sets.

2.2 Bias and Variance

What other fundamental properties of machine learning algorithms are required for learning from large data sets? This research focuses on the bias plus variance decomposition of error as a possible method of designing algorithms for use with large data sets.

The bias of a classification learning algorithm is a measure of the error that can be attributed to the central tendency of the models formed by the learner from different samples. The variance is a measure of the error that can be attributed to deviations from the central tendency of the models formed from different samples.

Unfortunately, while there is a straight-forward and generally accepted measure of these terms in the context of regression (prediction of numeric values), it is less straight-forward to derive an appropriate measure in a classification learning context. Alternative definitions include those of Kong & Dietterich [18], Kohavi & Wolpert [19], James & Hastie [20], Friedman [21], and Webb [17]. Of these numerous definitions we adopt Kohavi & Wolpert's definition [19] as it appears to be the most commonly employed in experimental machine learning research.

2.3 Bias and Variance and Data Set Size

We assume in the following that training data is an iid sample. As the data set size increases, the expected variance between different samples can be expected to decrease. As the differences between alternative samples decreases, the differences between the alternative models formed from those samples can also be expected to decrease. As differences between the models decrease, differences between predictions can also be expected to decrease. In consequence, when learning from large data sets we should expect variance to be lower than when learning from small data sets.

2.4 Management of Bias and Variance

It is credible that the effectiveness of many of a learning algorithm's strategies can be primarily attributed either to a reduction of bias or a reduction of variance. For example, there is evidence that decision tree pruning is primarily effective due to an ability to reduce variance [22]. If learning from small data sets requires effective variance management, it is credible that early learning algorithms, focusing on the needs of small data sets, lent more weight to strategies that are effective at variance management than those that are effective at bias management.

It is important to note that better management of bias does not necessarily equate to lower error due to bias (the same holds for variance). It can be trivially shown that an algorithm with better bias management can have worse predictive performance than an algorithm with less bias management. Therefore, the level of bias and variance management should be viewed as a good guide to performance, not a guarantee.

Both bias and variance management are important. However, if, as has been discussed above, variance can be expected to decrease as training set size increases regardless of the level of variance management, then it would seem logical that more focus can be placed on bias management without significant expected loss of accuracy due to an increase in variance error. The following experiments investigate whether this is true.

2.5 Hypothesis

There are two parts to the hypothesis. The first is that as training set size increases variance will decrease. The second is that as training set size increases, variance will become a less significant part of error. This is based on the stronger expectation for variance to decrease as training size increases than bias. Therefore, it seems plausible that the proportion of decrease in variance will be greater than that for bias.

3 Experiments

Experiments were performed to provide evidence towards the hypothesis. As discussed previously, different algorithms have different bias plus variance profiles. Thus, algorithms with a range of bias plus variance profiles were selected for testing. The first was Naïve Bayes, selected due to its extremely high variance management and extremely low bias management. The second algorithm was the decision tree exemplar C4.5. The many options of C4.5 allow small but important changes to the induction algorithm, altering the bias plus variance profile. It was therefore possible to investigate multiple profiles using the same basic algorithm. This helps in ensuring that any differences in trends found in different profiles are due to the differences in the profiles, not differences in the basic algorithm. The variants investigated were C4.5 with its default options (including pruning), C4.5 without pruning, and C4.5 without pruning and with the minimum number of instances per leaf set to 1. The MultiBoost [17] “meta-algorithm” was also used (with standard C4.5 as its base algorithm) as it has been shown to reduce both bias and variance. Table 1 details the algorithms used, and their associated expected bias plus variance profiles.

Pruning of decision trees has been shown to reduce variance [22]. Therefore, growing trees without pruning should reduce variance management. Reducing the number of instances required at a decision leaf in C4.5 should also result in lower variance management.

3.1 Methodology

Experiments were performed as follows. A data set was divided into three parts. One part was used as the hold-out test set. The training set was randomly sampled without replacement from the remaining two parts. A model was created and tested on the hold-out set. This sub-process was then repeated using each of the other two parts as the hold-out test set. This guarantees that each instance is classified once. The whole process was repeated ten times. Each instance is therefore classified precisely ten

Table 1. Selected algorithms and their bias plus variance profiles

ALGORITHM	BIAS PLUS VARIANCE PROFILE
NAÏVE BAYES	Very high variance management, very little bias management
C4.5	Medium variance management, medium bias management
MULTIBOOST C4.5	More bias and variance management than C4.5
C4.5 WITHOUT PRUNING	Less variance management than C4.5
C4.5 WITHOUT PRUNING, MINIMUM OF 1 INSTANCE AT LEAF	Very little variance management

times as a test instance, and used up to twenty times as a training instance. Training set sample sizes were powers of two - ranging from 32 instances to the highest power of 2 that was less than two-thirds of the number of instances in the entire data set.

Seven freely available data sets from the UCI Machine Learning Repository [23] were used (outlined in Table 2).

Data sets were required to be: a) useable for classification, b) large enough for use with the methodology, so as to provide a sufficiently large maximum training set size, and c) publicly available.

Table 2. Description of data sets

DATA SET	NUMBER OF INSTANCES	CONT ATTRS	DISC ATTRS	CLASSES
ADULT	48,842	6	8	2
CENSUS INCOME	199,523	7	33	2
CONNECT-4	67,557	0	42	3
COVER TYPE	581,012	10	44	7
IPUMS	88,443	60	0	13
SHUTTLE	58,000	9	0	7
WAVEFORM	1,600,000	21	0	3

3.2 Results

Graphs show the relation of bias or variance to data set size for all algorithms used. Note that the error for Naïve-Bayes on the waveform data set increases dramatically at training set size 32,768. This occurred for all measures, and was investigated with no clear reason for such behavior found.

3.2.1 Variance

See Figure 1(a-g). In general, all algorithms follow the trend to decrease in variance as training set size increases for all data sets. The one exception is Naïve Bayes on the Census-Income data, where there are substantial increases in variance.

3.2.2 Bias

See Figure 2(a-g). For all data sets all algorithms except Naïve-Bayes tend to decrease in bias as training set size increases. Naïve-Bayes, an algorithm with very little bias management, increases in bias for all data sets except waveform. Although no hypothesis was offered regarding the trend of bias alone, this suggests that bias management is extremely important.

3.2.3 Ratio of Bias to Variance

See Figure 3(a-g). Note that results are presented as the proportion of bias of overall error, rather than a direct relation of bias to variance for simplification of scales. The results show that varying training set size can have different effects on bias and variance profiles. To evaluate the effect of increasing training set size on the relative importance of bias and variance, we look at the difference in the ratio of bias to variance between the smallest and the largest training set size for each data set. If the ratio increases then bias is increasing in the amount it dominates the final error term. If the ratio decreases then the degree to which variance dominates the final error term is increasing.

The second part of the hypothesis is that variance will become a larger portion of the error with increasing training set size. The comparison found that of the 35 comparisons, 28 were in favor of the hypothesis, with only 7 against. This is significant at the 0.05 level, using a one-tailed sign test ($p=0.0003$).

3.3 Summary

The results show a general trend for variance to decrease with increased training set size. The trend is certainly not as strong as that for bias. However, this trend exists with all algorithms used. Even unpruned C4.5 with minimum leaf instance of one, an algorithm with extremely little variance management, shows the trend. This suggests that variance management may not be of extreme importance in an algorithm when dealing with large data sets. This is not to suggest that variance management is unnecessary, since more variance management can still be expected to result in less variance error. However, these results do suggest that, as expected, variance will naturally decrease with larger training set sizes. The results also support the second part of the hypothesis; that bias can be expected to become a larger portion of error.

3.4 Does Lower Variance Management Imply Higher Bias Management?

It might be thought that management of bias and variance are interlinked so that approaches to reduce bias will increase variance and vice versa. This hypothesis was evaluated with respect to our experiments by examining the effects on bias and variance of the variants of C4.5. The bias and variance of each of the three variants (unpruned, unpruned with minimum leaf size of 1, and MultiBoosting) were compared to the bias and variance of C4.5 with default settings. The number of times the signs of the differences differed (190) was compared with the number of times the

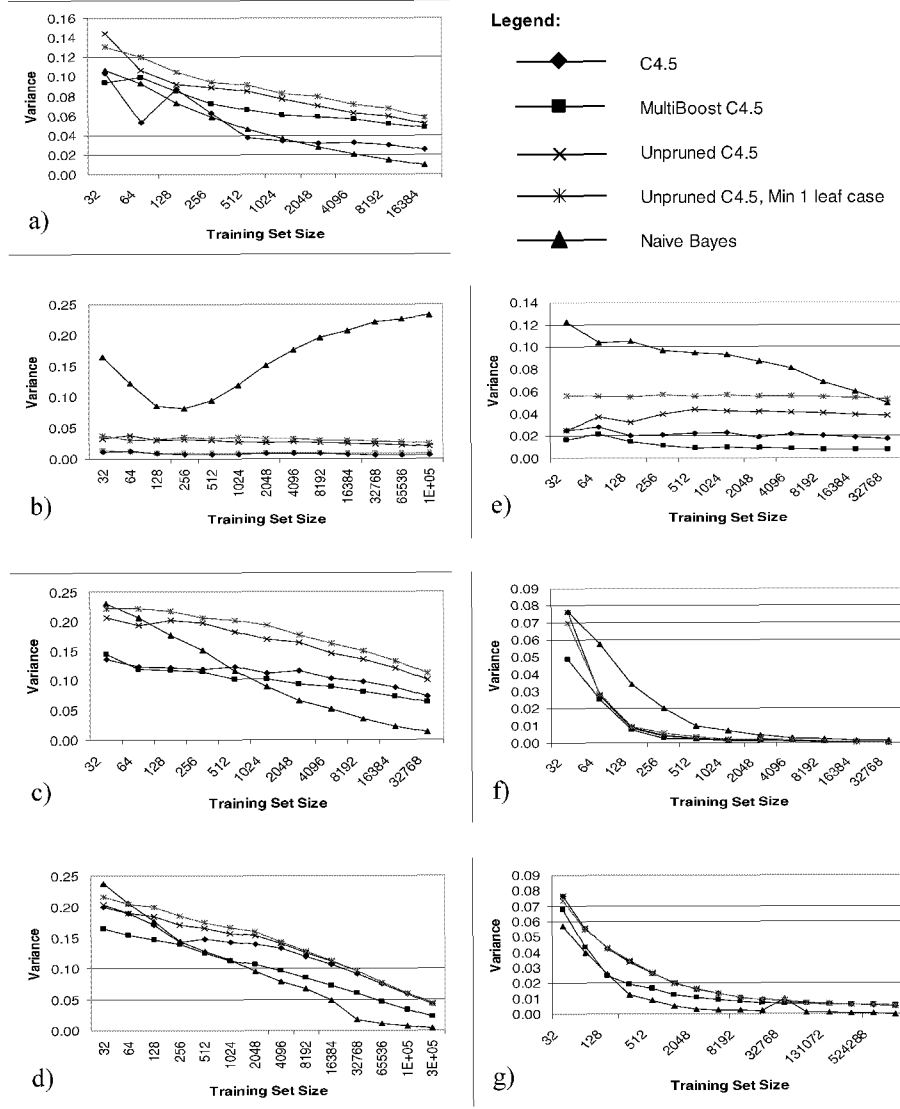


Fig. 1. Variance of algorithms on data sets for different training set sizes. Data sets are a) Adult, b) Census Income, c) Connect-4, d) Cover Type, e) IPUMS, f) Shuttle, g) Waveform

signs were the same (59), with occasions where there was no difference (9) ignored. A one-tailed binomial sign test ($p < 0.0001$) indicates that this outcome indicates a significantly greater chance that a decrease in variance will correspond with an increase in bias and vice versa than that they will both increase or decrease in unison as a result of a modification to a learning algorithm.

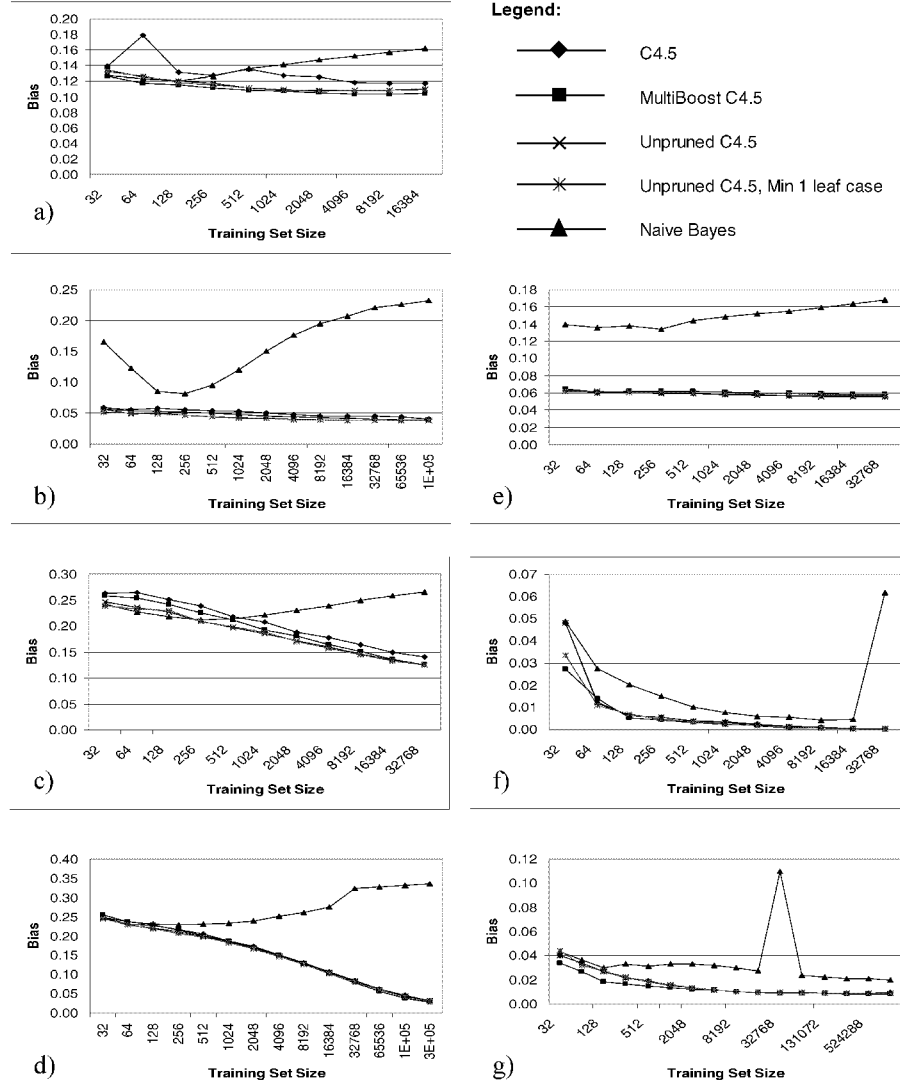


Fig. 2. Bias of algorithms on data sets for different training set sizes. Data sets are a) Adult, b) Census Income, c) Connect-4, d) Cover Type, e) IPUMS, f) Shuttle, g) Waveform

This might be taken as justification for not aiming to manage bias in preference to managing variance at large data set sizes, as managing bias will come at the expense of managing variance. However, while the directions of the effects on bias and variance tend to be the opposite, the magnitudes also differ. The mean absolute difference between the variance of C4.5 with default settings and one of its variants was 0.0221. The mean difference between the bias of C4.5 with default settings and

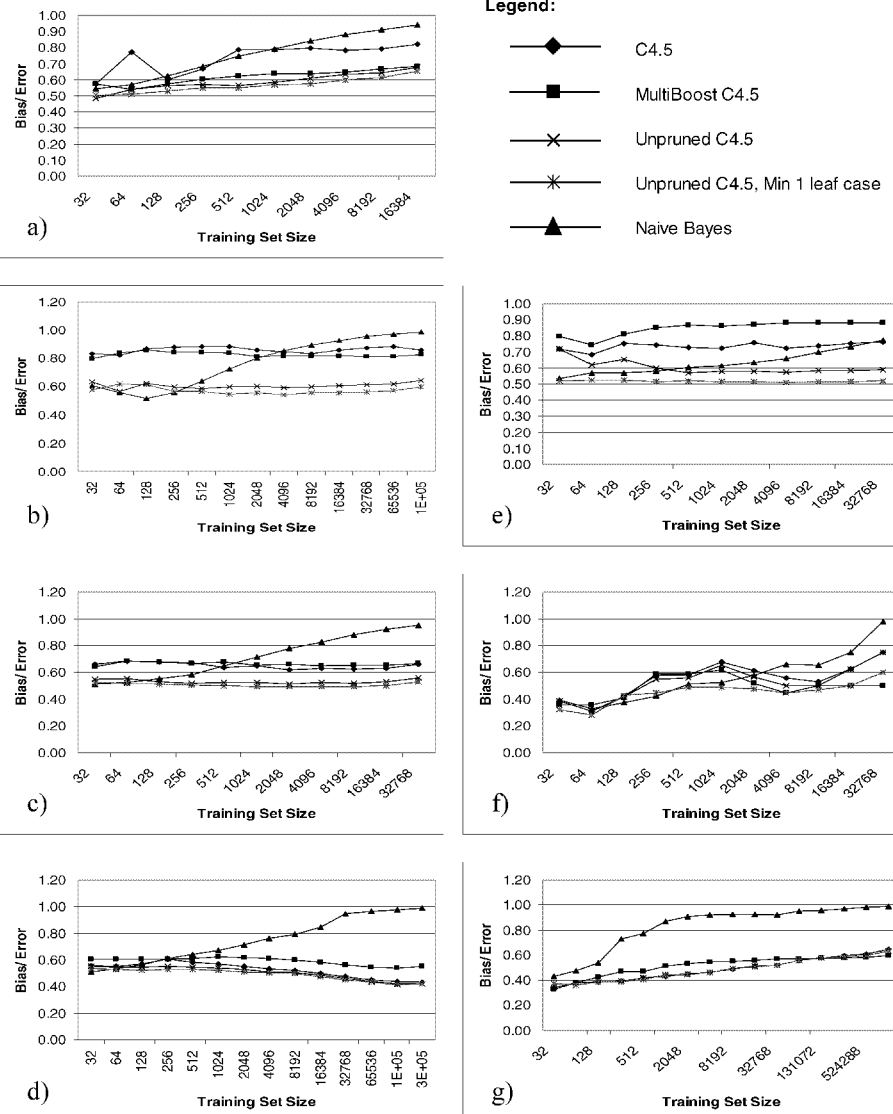


Fig. 3. Ratio of bias to variance of algorithms on data sets for different training set sizes. Data sets are a) Adult, b) Census Income, c) Connect-4, d) Cover Type, e) IPUMS, f) Shuttle, g) Waveform

one of its variants was 0.0101. A one-tailed matched-pair t-test indicates that the effect on variance of the variants of C4.5 is greater than the effect on bias. This adds credibility to the hypothesis that current machine learning algorithms reflect their small data set origins by incorporating primarily variance management measures.

4 Conclusions and Further Work

This paper details experiments performed to investigate whether a) the statistical expectations of bias and variance do indeed apply to classification learning, and b) if bias becomes a larger portion of error as training set size increases. The results support both parts of the hypothesis. Variance, in general, decreases as training set size increases. This appears to be irrespective of the bias plus variance profile of the algorithm. Bias also generally decreases, with more regularity than variance. The one notable exception to this is Naïve-Bayes, an algorithm that employs little bias management. This somewhat surprising result alone suggests that bias management is indeed an important factor in learning from large data sets. An analysis of the impact on bias and variance of changes to a learning algorithm suggest that measures that decrease variance can be expected to increase bias and vice versa. However, the magnitudes of these changes differ markedly, variance being affected more than bias. This suggests that the measures incorporated in standard learning algorithms do indeed relate more to variance management than bias management. The results also show that as training set size increases bias can be expected to become a larger portion of error.

Unfortunately, creating algorithms that focus on bias management seems to be a difficult task. We can, however, identify some methods that may be expected to lower bias. One possibility is to create algorithms with a “larger than usual” hypothesis space. For example, it could be expected that an algorithm that can create non-axis-orthogonal partitions should have less bias than an algorithm that can only perform axis-orthogonal partitions. The drawback of this is an increase in search. Another option might be to introduce a large random factor into the creation of a model. This could be expected to convert at least some of the bias error into variance error. However, the way in which randomization should be included does not appear obvious.

These experiments are by no means exhaustive. Thus, there is scope for continued investigation using a wider range of algorithms, and more and larger data sets.

The data sets used in this research are not considered particularly large by today’s standards. Unfortunately, hardware constraints limited the size of the data sets used. However, even with the data sets employed in this study, trends are apparent. It is reasonable to expect that with massive data sets these trends should continue, and possibly become stronger.

We have shown that the importance of bias management grows as data set size grows. We have further presented evidence that current algorithms are oriented more toward management of variance than management of bias. We believe that the strong implication of this work is that classification learning error from large data sets may be further reduced by the development of learning algorithms that place greater emphasis on reduction of bias.

References

1. Provost, F., Aronis, J.: Scaling Up Inductive Learning with Massive Parallelism. *Machine Learning*, Vol. 23. (1996) 33-46
2. Provost, F., Jensen, D., Oates, T.: Efficient Progressive Sampling. *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*. ACM Press, New York (1999) 22-32
3. Shafer, J., Agrawal, R., Mehta, M.: SPRINT: A Scalable Parallel Classifier for Data Mining. *Proceedings of the Twenty-Second VLDB Conference*. Morgan Kaufmann, San Francisco (1996) 544-555
4. Catlett, J.: Peephaling: Choosing Attributes Efficiently for Megainduction. *Proceedings of the Ninth International Conference on Machine Learning*. Morgan Kaufmann, San Mateo (1992) 49-54
5. Cohen, W.: Fast Effective Rule Induction. *Proceedings of the Twelfth International Conference on Machine Learning*. Morgan Kaufmann, San Francisco (1995) 115-123
6. Aronis, J., Provost, F.: Increasing the Efficiency of Data Mining Algorithms with Breadth-First Marker Propagation. *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*. AAAI Press, Menlo Park (1997) 119-122
7. Quinlan, J. R.: C4.5: Programs for Machine Learning. Morgan Kaufmann, San Mateo (1993)
8. Breiman, L., Freidman, J. H., Olshen, R. A., Stone, C. J.: *Classification and Regression Trees*. Wadsworth International, Belmont (1984)
9. Hecht-Nielsen, R.: *Neurocomputing*. Addison-Wesley, Menlo Park (1990)
10. Cover, T. M., Hart, P. E.: Nearest Neighbor Pattern Classification. *IEEE Transactions on Information Theory*, Vol. 13. (1967) 21-27
11. Gehrke, J., Ramakrishnan, R., Ganti, V.: RainForest □ A Framework for Fast Decision Tree Induction. *Proceedings of the Twenty-fourth International Conference on Very Large Databases*. Morgan Kaufmann, San Mateo (1998)
12. Moore, A., Lee, M. S.: Cached Sufficient Statistics for Efficient Machine Learning with Large Datasets. *Journal of Artificial Intelligence Research*, Vol. 8. (1998) 67-91
13. Chattrachit, J., Darlington, J., Ghanem, M., Guo, Y., Huning, H., Kohler, M., Sutiwaraphun, J., To, H. W., Yang, D.: Large Scale Data Mining: Challenges and Responses. *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*. AAAI Press, Menlo Park (1997)
14. Freund, Y., Schapire, R. E.: A Decision-Theoretic Generalization of On-line Learning and an Application to Boosting. *Journal of Computer and System Sciences*, Vol. 55. (1997) 95-121
15. Breiman, L.: Arcing Classifiers. Technical Report 460. Department of Statistics, University of California, Berkeley (1996)
16. Breiman, L.: Bagging Predictors. *Machine Learning*, Vol. 24. (1996) 123-140.
17. Webb, G. (2000). MultiBoosting: A Technique for Combining Boosting and Wagging. *Machine Learning*, Vol. 40, (2000) 159-196

18. Kong, E. B., Dietterich, T. G.: Error-Correcting Output Coding Corrects Bias and Variance. Proceedings of the Twelfth International Conference on Machine Learning. Morgan Kaufmann, San Mateo (1995)
19. Kohavi, R., Wolpert, D. H.: Bias Plus Variance Decomposition for Zero-One Loss Functions. Proceedings of the Thirteenth International Conference on Machine Learning. Morgan Kaufmann, San Francisco (1996)
20. James, G, Hastie, T.: Generalizations of the bias/variance decomposition for prediction error. Technical Report. Department of Statistics, Stanford University (1997)
21. Friedman, J. H.: On Bias, Variance, 0/1-Loss, and the Curse-of-Dimensionality. Data Mining and Knowledge Discovery, Vol. 1. (1997) 55-77
22. Bauer, E., Kohavi, R.: An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants. Machine Learning, Vol. 36. (1999) 105-142
23. Blake, C. L., Merz, C. J. UCI Repository of Machine Learning Databases [<http://www.ics.uci.edu/~mlearn/MLRepository.html>]. Department of Information and Computer Science, University of California, Irvine

Mining All Non-derivable Frequent Itemsets

Toon Calders^{1,*} and Bart Goethals²

¹ University of Antwerp, Belgium

² University of Limburg, Belgium

Abstract. Recent studies on frequent itemset mining algorithms resulted in significant performance improvements. However, if the minimal support threshold is set too low, or the data is highly correlated, the number of frequent itemsets itself can be prohibitively large. To overcome this problem, recently several proposals have been made to construct a concise representation of the frequent itemsets, instead of mining all frequent itemsets. The main goal of this paper is to identify redundancies in the set of all frequent itemsets and to exploit these redundancies in order to reduce the result of a mining operation. We present deduction rules to derive tight bounds on the support of candidate itemsets. We show how the deduction rules allow for constructing a minimal representation for all frequent itemsets. We also present connections between our proposal and recent proposals for concise representations and we give the results of experiments on real-life datasets that show the effectiveness of the deduction rules. In fact, the experiments even show that in many cases, first mining the concise representation, and then creating the frequent itemsets from this representation outperforms existing frequent set mining algorithms.

1 Introduction

The frequent itemset mining problem [1] is by now well known. We are given a set of items $\mathcal{I}$ and a database $\mathcal{D}$ of subsets of $\mathcal{I}$, together with a unique identifier. The elements of $\mathcal{D}$ are called transactions. An *itemset* $I \subseteq \mathcal{I}$ is some set of items; its *support* in $\mathcal{D}$, denoted by $\text{support}(I, \mathcal{D})$, is defined as the number of transactions in $\mathcal{D}$ that contain all items of I ; and an itemset is called *s-frequent* in $\mathcal{D}$ if its support in $\mathcal{D}$ exceeds s . $\mathcal{D}$ and s are omitted when they are clear from the context. The goal is now, given a minimal support threshold and a database, to find all frequent itemsets.

The search space of this problem, all subsets of $\mathcal{I}$, is clearly huge. Instead of generating and counting the supports of all these itemsets at once, several solutions have been proposed to perform a more directed search through all patterns. However, this directed search enforces several scans through the database, which brings up another great cost, because these databases tend to be very large, and hence they do not fit into main memory.

* Research Assistant of the Fund for Scientific Research - Flanders (FWO-Vlaanderen)

The standard Apriori algorithm [2] for solving this problem is based on the *monotonicity property*: all supersets of an infrequent itemset must be infrequent. Hence, if an itemset is infrequent, then all of its supersets can be *pruned* from the search-space. An itemset is thus considered potentially frequent, also called a *candidate* itemset, only if all its subsets are already known to be frequent. In every step of the algorithm, all candidate itemsets are generated and their supports are then counted by performing a complete scan of the transaction database. This is repeated until no new candidate itemsets can be generated.

Recent studies on frequent itemset mining algorithms resulted in significant performance improvements. In the early days, the size of the database and the generation of a reasonable amount of frequent itemsets were considered the most costly aspects of frequent itemset mining, and most energy went into minimizing the number of scans through the database. However, if the minimal support threshold is set too low, or the data is highly correlated, the number of frequent itemsets itself can be prohibitively large. To overcome this problem, recently several proposals have been made to construct a concise representation of the frequent itemsets, instead of mining all frequent itemsets [13,3,6,5,14,15,7,11].

Our contributions The main goal of this paper is to present several new methods to identify redundancies in the set of all frequent itemsets and to exploit these redundancies, resulting in a concise representation of all frequent itemsets and significant performance improvements of a mining operation.

1. We present a complete set of *deduction rules* to derive *tight* intervals on the support of candidate itemsets.
2. We show how the deduction rules can be used to construct a *minimal representation* of all frequent itemsets, consisting of all frequent itemsets of which the exact support can not be derived, and present an algorithm that efficiently does so.
3. Also based on these deduction rules, we present an efficient method to find the exact support of all frequent itemsets, that are not in this concise representation, *without scanning the database*.
4. We present *connections* between our proposal and recent proposals for concise representations, such as *free sets* [6], *disjunction-free sets* [7], and *closed sets* [13]. We also show that known tricks to improve performance of frequent itemset mining algorithms, such as used in MAXMINER [4] and PASCAL [3], can be described in our framework.
5. We present several *experiments* on real-life datasets that show the effectiveness of the deduction rules.

The outline of the paper is as follows. In Section 2 we introduce the deduction rules. Section 3 describes how we can use the rules to reduce the set of frequent itemsets. In Section 4 we give an algorithm to efficiently find this reduced set, and in Section 5 we evaluate the algorithm empirically. Related work is discussed in depth in Section 6.

2 Deduction Rules

In all that follows, $\mathcal{I}$ is the set of all items and $\mathcal{D}$ is the transaction database.

We will now describe sound and complete rules for deducing tight bounds on the support of an itemset $I \subseteq \mathcal{I}$, if the supports of all its subsets are given. In order to do this, we will not consider itemsets that are no subset of I , and we can assume that all items in $\mathcal{D}$ are elements of I . Indeed, “projecting away” the other items in a transaction database does not change the supports of the subsets of I .

Definition 1. (*I*-Projection) Let $I \subseteq \mathcal{I}$ be an itemset.

- The *I*-projection of a transaction T , denoted $\pi_I T$, is defined as $\pi_I T := \{i \mid i \in T \cap I\}$.
- The *I*-projection of a transaction database $\mathcal{D}$, denoted $\pi_I \mathcal{D}$, consist of all *I*-projected transactions from $\mathcal{D}$.

Lemma 1. Let I, J be itemsets, such that $I \subseteq J \subseteq \mathcal{I}$. For every transaction database $\mathcal{D}$, the following holds:

$$\text{support}(I, \mathcal{D}) = \text{support}(I, \pi_J \mathcal{D}).$$

Before we introduce the deduction rules, we introduce fractions and covers.

Definition 2. (*I*-Fraction) Let I, J be itemsets, such that $I \subseteq J \subseteq \mathcal{I}$, the *I*-fraction of $\pi_J \mathcal{D}$, denoted by $f_I^J(\mathcal{D})$ equals the number of transactions in $\pi_J \mathcal{D}$ that exactly consist of the set I .

If $\mathcal{D}$ is clear from the context, we will write f_I^J , and if $J = \mathcal{I}$, we will write f_I . The support of an itemset I is then $\sum_{I \subseteq I' \subseteq \mathcal{I}} f_{I'}$.

Definition 3. (Cover) Let $I \subseteq \mathcal{I}$ be an itemset. The cover of I in $\mathcal{D}$, denoted by $\text{Cover}(I, \mathcal{D})$, consists of all transactions in $\mathcal{D}$ that contain I .

Again, we will write $\text{Cover}(I)$ if $\mathcal{D}$ is clear from the context.

Let $I, J \subseteq \mathcal{I}$ be itemsets, and $J = I \cup \{A_1, \dots, A_n\}$. Notice that $\text{Cover}(J) = \bigcap_{i=1}^n \text{Cover}(I \cup \{A_i\})$, and that $|\bigcup_{i=1}^n \text{Cover}(I \cup \{A_i\})| = |\text{Cover}(I)| - f_I^J$. From the well-known *inclusion-exclusion principle* [10, p.181] we learn

$$\begin{aligned} |\text{Cover}(I)| - f_I^J &= \sum_{1 \leq i \leq n} |\text{Cover}(I \cup \{A_i\})| \\ &\quad - \sum_{1 \leq i < j \leq n} |\text{Cover}(I \cup \{A_i, A_j\})| + \dots - (-1)^n |\text{Cover}(J)|, \end{aligned}$$

and since $\text{support}(I \cup \{A_{i_1}, \dots, A_{i_\ell}\}) = |\text{Cover}(I \cup \{A_{i_1}, \dots, A_{i_\ell}\})|$, we obtain

$$\begin{aligned} (-1)^{|J-I|} \text{support}(J) - f_I^J &= \text{support}(I) - \sum_{1 \leq i \leq n} \text{support}(I \cup \{A_i\}) \\ &\quad + \sum_{1 \leq i < j \leq n} \text{support}(I \cup \{A_i, A_j\}) + \dots + (-1)^{|J-I|-1} \sum_{1 \leq i \leq n} \text{support}(J - \{A_i\}) \end{aligned}$$

From now on, we will denote the sum on the right-hand side of this last equation by $\sigma(I, J)$.

Since f_I^J is always positive, we obtain the following theorem.

Theorem 1. *For all itemsets $I, J \subseteq \mathcal{I}$, $\sigma(I, J)$ is a lower (upper) bound on $\text{support}(J)$ if $|J - I|$ is even (odd). The difference $|\text{support}(J) - \sigma(I, J)|$ is given by f_I^J .*

We will refer to the rule involving $\sigma(I, J)$ as $\mathcal{R}_J(I)$ and omit J when clear from the context.

If for each subset $I \subset J$, the support $\text{support}(I, \mathcal{D}) = s_I$ is given, then the rules $\mathcal{R}_J(\cdot)$ allow for calculating lower and upper bounds on the support of J . Let l denote the greatest lower bound we can derive with these rules, and u the smallest upper bound we can derive. Since the rules are sound, the support of J must be in the interval $[l, u]$. In [8], we show also that these bounds on the support of J are *tight*; i.e., for every smaller interval $[l', u'] \subset [l, u]$, we can find a database $\mathcal{D}'$ such that for each subset I of J , $\text{support}(I, \mathcal{D}') = s_I$, but the support of J is not within $[l', u']$.

Theorem 2. *For all itemsets $I, J \subseteq \mathcal{I}$, the rules $\{\mathcal{R}_J(I) \mid I \subseteq J\}$ are sound and complete for deducing bounds on the support of J based on the supports of all subsets of J .*

The proof of the completeness relies on the fact that for all $I \subseteq J$, we have $\text{support}(I, \mathcal{D}) = \sum_{I' \subseteq I \subseteq \mathcal{I}} f_{I'}$. We can consider the linear program consisting of all these equalities, together with the conditions $f_I \geq 0$ for all fractions f_I . The existence of a database $\mathcal{D}'$ that satisfies the given supports is equivalent to the existence of a solution to this linear program in the f_I 's and $\text{support}(J, \mathcal{D}')$. From this equivalence, tightness of the bounds can be proved. For the details of the proof we refer to [8].

Example 1. Consider the following transaction database.

$\mathcal{D} =$	A, B, C	$s_A = 5, \quad s_B = 7, \quad s_C = 7,$
	A, C, D	
	A, B, D	
	C, D	
	B, C, D	
	A, D	
	B, D	
	B, C, D	
	B, C, D	
	A, B, C, D	

Figure 1 gives the rules to determine tight bounds on the support of $ABCD$. Using these deduction rules, we derive the following bounds on $\text{support}(ABCD)$ without counting in the database.

Lower bound: $\text{support}(ABCD) \geq 1$ (Rule $\mathcal{R}(AC)$)
Upper bound: $\text{support}(ABCD) \leq 1$ (Rule $\mathcal{R}(A)$)

$$\left\{ \begin{array}{ll}
\text{support}(ABCD) \geq s_{ABC} + s_{ABD} + s_{ACD} + s_{BCD} - s_{AB} - s_{AC} - s_{AD} & \mathcal{R}_{\{\}} \\
& - s_{BC} - s_{BD} - s_{CD} + s_A + s_B + s_C + s_D - s_{\{\}} \\
\text{support}(ABCD) \leq s_A - s_{AB} - s_{AC} - s_{AD} + s_{ABC} + s_{ABD} + s_{ACD} & \mathcal{R}_A \\
\text{support}(ABCD) \leq s_B - s_{AB} - s_{BC} - s_{BD} + s_{ABC} + s_{ABD} + s_{BCD} & \mathcal{R}_B \\
\text{support}(ABCD) \leq s_C - s_{AC} - s_{BC} - s_{CD} + s_{ABC} + s_{ACD} + s_{BCD} & \mathcal{R}_C \\
\text{support}(ABCD) \leq s_D - s_{AD} - s_{BD} - s_{CD} + s_{ABD} + s_{ACD} + s_{BCD} & \mathcal{R}_D \\
\text{support}(ABCD) \geq s_{ABC} + s_{ABD} - s_{AB} & \mathcal{R}_{AB} \\
\text{support}(ABCD) \geq s_{ABC} + s_{ACD} - s_{AC} & \mathcal{R}_{AC} \\
\text{support}(ABCD) \geq s_{ABD} + s_{ACD} - s_{AD} & \mathcal{R}_{AD} \\
\text{support}(ABCD) \geq s_{ABC} + s_{BCD} - s_{BC} & \mathcal{R}_{BC} \\
\text{support}(ABCD) \geq s_{ABD} + s_{BCD} - s_{BD} & \mathcal{R}_{BD} \\
\text{support}(ABCD) \geq s_{ACD} + s_{BCD} - s_{CD} & \mathcal{R}_{CD} \\
\text{support}(ABCD) \leq s_{ABC} & \mathcal{R}_{ABC} \\
\text{support}(ABCD) \leq s_{ABD} & \mathcal{R}_{ABD} \\
\text{support}(ABCD) \leq s_{ACD} & \mathcal{R}_{ACD} \\
\text{support}(ABCD) \leq s_{BCD} & \mathcal{R}_{BCD} \\
\text{support}(ABCD) \geq 0 & \mathcal{R}_{ABCD}
\end{array} \right.$$

Fig. 1. Tight bounds on $\text{support}(ABCD)$. s_I denotes $\text{support}(I)$

Therefore, we can conclude, without having to rescan the database, that the support of $ABCD$ in $\mathcal{D}$ is exactly 1, while a standard monotonicity check would yield an upper bound of 2.

3 Non-derivable Itemsets as a Concise Representation

Based on the deduction rules, it is possible to generate a summary of the set of frequent itemsets. Indeed, suppose that the deduction rules allow for deducing the support of a frequent itemset I *exactly*, based on the supports of its subsets. Then there is no need to explicitly count the support of I requiring a complete database scan; if we need the support of I , we can always simply derive it using the deduction rules. Such a set I , of which we can perfectly derive the support, will be called a *Derivable Itemset* (DI), all other itemsets are called *Non-Derivable Itemsets* (NDIs). We will show in this section that the set of frequent NDIs allows for computing the supports of all other frequent itemsets, and as such, forms a *concise representation* [12] of the frequent itemsets. To prove this result, we first need to show that when a set I is non-derivable, then also all its subsets are non-derivable. For each set I , let l_I (u_I) denote the lower (upper) bound we can derive using the deduction rules.

Lemma 2. (Monotonicity) *Let $I \subseteq \mathcal{I}$ be an itemset, and $i \in \mathcal{I} - I$ an item. Then $2|u_{I \cup \{i\}} - l_{I \cup \{i\}}| \leq 2 \min(|\text{support}(I) - l_I|, |\text{support}(I) - u_i|) \leq |u_I - l_I|$. In particular, if I is a DI, then also $I \cup \{i\}$ is a DI.*

Proof. The proof is based on the fact that $f_J^I = f_J^{I \cup \{i\}} + f_{J \cup \{i\}}^{I \cup \{i\}}$. From Theorem 1 we know that f_J^I is the difference between the bound calculated by $\mathcal{R}_I(J)$ and

the real support of I . Let now J be such that the rule $\mathcal{R}_I(J)$ calculates the bound that is closest to the support of I . Then, the width of the interval $[l_I, u_I]$ is at least $2f_J^I$. Furthermore, $\mathcal{R}_{I \cup \{i\}}(J)$ and $\mathcal{R}_{I \cup \{i\}}(J \cup \{i\})$ are a lower and an upper bound on the support of $I \cup \{i\}$ (if $|I \cup \{i\} - (J \cup \{i\})|$ is odd, then $|I \cup \{i\} - J|$ is even and vice versa), and these bounds on $I \cup \{i\}$ differ respectively $f_J^{I \cup \{i\}}$ and $f_{J \cup \{i\}}^{I \cup \{i\}}$ from the real support of $I \cup \{i\}$. When we combine all these observations, we get: $u_{I \cup \{i\}} - l_{I \cup \{i\}} \leq f_J^{I \cup \{i\}} + f_{J \cup \{i\}}^{I \cup \{i\}} = f_J^I \leq \frac{1}{2}(u_I - l_I)$. $\square$

This lemma gives us the following valuable insights.

Corollary 1. *The width of the intervals exponentially shrinks with the size of the itemsets.*

This remarkable fact is a strong indication that the number of large NDIs will be very small. This reasoning will be supported by the results of the experiments.

Corollary 2. *If I is a NDI, but it turns out that $\mathcal{R}_I(J)$ equals the support of I , then all supersets $I \cup \{i\}$ of I will be a DI, with rules $\mathcal{R}_{I \cup \{i\}}(J)$ and $\mathcal{R}_{I \cup \{i\}}(J \cup \{i\})$.*

We will use this observation to avoid checking all possible rules for $I \cup \{i\}$. This avoidance can be done in the following way: whenever we calculate bounds on the support of an itemset I , we remember the lower and upper bound l_I, u_I . If I is a NDI; i.e., $l_I \neq u_I$, then we will have to count its support. After we counted the support, the tests $\text{support}(I) = l_I$ and $\text{support}(I) = u_I$ are performed. If one of these two equalities obtains, we know that all supersets of I are derivable, without having to calculate the bounds.

Corollary 3. *If we know that I is a DI, and that rule $\mathcal{R}_I(J)$ gives the exact support of I , then $\mathcal{R}_{I \cup \{i\}}(J \cup \{i\})$ gives the exact support for $I \cup \{i\}$.*

Suppose that we want to build the entire set of frequent itemsets starting from the concise representation. We can then use this observation to improve the performance of deducing all supports. Suppose we need to deduce the support of a set I , and of a superset J of I ; instead of trying all rules to find the exact support for J , we know in advance, because we already evaluated I , which rule to choose. Hence, for any itemset which is known to be a DI, we only have to compute a single deduction rule to know its exact support.

From Lemma 2, we easily obtain the following theorem, saying that the set of NDIs is a concise representation. We omit the proof due to space limitations.

Theorem 3. *For every database $\mathcal{D}$, and every support threshold s , let $\text{NDI}(\mathcal{D}, s)$ be the following set:*

$$\text{NDI}(\mathcal{D}, s) := \{(I, \text{support}(I, \mathcal{D})) \mid l_I \neq u_I\}.$$

$\text{NDI}(\mathcal{D}, s)$ is a concise representation for the frequent itemsets, and for each itemset J not in $\text{NDI}(\mathcal{D}, s)$, we can decide whether J is frequent, and if J is frequent, we can exactly derive its support from the information in $\text{NDI}(\mathcal{D}, s)$.

4 The NDI-Algorithm

Based on the results in the previous section, we propose a level-wise algorithm to find all frequent NDIs. Since derivability is monotone, we can prune an itemset if it is derivable. This gives the NDI-algorithm as shown below. The correctness of the algorithm follows from the results in Lemma 2.

```

NDI( $\mathcal{D}, s$ )
   $i := 1$ ; NDI :=  $\{\}$ ;  $C_1 := \{\{i\} \mid i \in \mathcal{I}\}$ ;
  for all  $I$  in  $C_1$  do  $I.l := 0$ ;  $I.u := |\mathcal{D}|$ ;
  while  $C_i$  not empty do
    Count the supports of all candidates in  $C_i$  in one pass over  $\mathcal{D}$ ;
     $F_i := \{I \in C_i \mid \text{support}(I, \mathcal{D}) \geq s\}$ ;
    NDI := NDI  $\cup$   $F_i$ ;
     $Gen := \{\}$ ;
    for all  $I \in F_i$  do
      if  $\text{support}(I) \neq I.l$  and  $\text{support}(I) \neq I.u$  then
         $Gen := Gen \cup \{I\}$ ;
     $PreC_{i+1} := \text{AprioriGenerate}(Gen)$ ;
     $C_{i+1} := \{\}$ ;
    for all  $J \in PreC_{i+1}$  do
      Compute bounds  $[l, u]$  on support of  $J$ ;
      if  $l \neq u$  then  $J.l := l$ ;  $J.u := u$ ;  $C_{i+1} := C_{i+1} \cup \{J\}$ ;
     $i := i + 1$ 
  end while
  return NDI

```

Since evaluating all rules can be very cumbersome, in the experiments we show what the effect is of only using a couple of rules. We will say that we use rules *up to depth* k if we only evaluate the rules $\mathcal{R}_J(I)$ for $|I - J| \leq k$. The experiments show that in most cases, the gain of evaluating rules up to depth k instead of up to depth $k - 1$ typically quickly decreases if k increases. Therefore, we can conclude that in practice most pruning is done by the rules of limited depth.

5 Experiments

For our experiments, we implemented an optimized version of the Apriori algorithm and the NDI algorithm described in the previous section. We performed our experiments on several real-life datasets with different characteristics, among which a dataset obtained from a Belgian retail market, which is a sparse dataset of 41 337 transaction over 13 103 items. The second dataset was the BMS-Webview-1 dataset donated by Z. Zheng et al. [16], containing 59 602 transactions over 497 items. The third dataset is the dense census-dataset as available in the UCI KDD repository [9], which we transformed into a transaction database by creating a different item for every attribute-value pair, resulting

in 32 562 transactions over 22 072 items. The results on all these datasets were very similar and we will therefore only describe the results for the latter dataset.

Figure 2 shows the average width of the intervals computed for all candidate itemsets of size k . Naturally, the interval-width of the singleton candidate itemsets is 32 562, and is not shown in the figure. In the second pass of the NDI-algorithm, all candidate itemsets of size 2 are generated and their intervals deduced. As can be seen, the average interval size of most candidate itemsets of size 2 is 377. From then on, the interval sizes decrease exponentially as was predicted by Corollary 1.

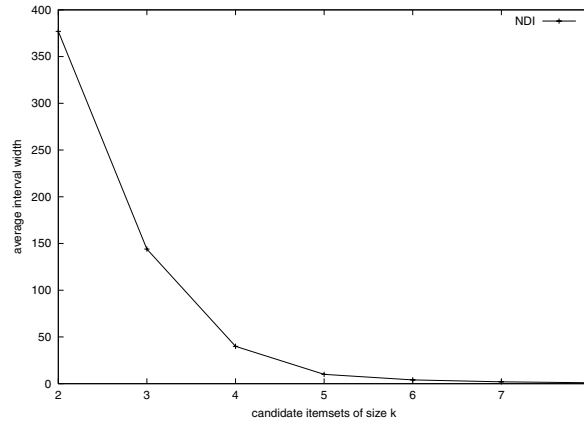


Fig. 2. Average interval-width of candidate itemsets

Figure 3 shows the size of the concise representation of all NDIs compared to the total number of frequent patterns as generated by Apriori, for varying minimal support thresholds. If this threshold was set to 0.1%, there exist 990 097 frequent patterns of which only 162 821 are non-derivable. Again this shows the theoretical results obtained in the previous sections.

In the last experiment, we compared the strength of evaluating the deduction rules up to a certain depth, and the time needed to generate all NDIs w.r.t. the given depth. Figure 4 shows the results. On the x-axis, we show the depth up to which rules are evaluated. We denoted the standard Apriori monotonicity check by 0, although it is actually equivalent to the rules of depth 1. The reason for this is that we also used the other optimizations described in Section 3. More specifically, if the lower or upper bound of an itemset equals its actual support, we can prune its supersets, which is denoted as depth 1 in this figure. The left y-axis shows the number of NDIs w.r.t. the given depth and is represented by the line ‘concise representation’. The line ‘NDI’ shows the time needed to generate these NDIs. The time is shown on the right y-axis. The ‘NDI+DI’ line shows the time needed to generate all NDIs plus the time needed to derive all DIs, resulting

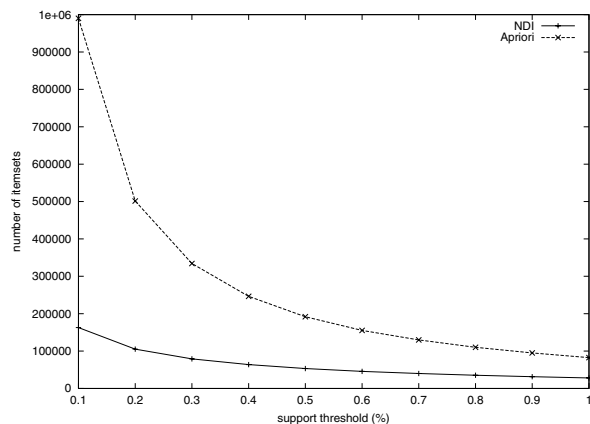


Fig. 3. Size of concise representation

in all frequent patterns. As can be seen, the size of the concise representation drops quickly only using the rules of depth 1 and 2. From there on, higher depths result in a slight decrease of the number of NDIs. From depth 4 on, this size stays the same, which is not that remarkable since the number of NDIs of these sizes is also small. The time needed to generate these sets is best if the rules are only evaluated up to depth 2. Still, the running time is almost always better than the time needed to generate all frequent itemsets (depth 0), and is hardly higher for higher depths. For higher depths, the needed time increases, which is due to the number of rules that need to be evaluated. Also note that the total time required for generating all NDIs and deriving all DIs is also better than generating all frequent patterns at once, at depth 1,2,and 3. This is due to the fact that the NDI algorithm has to perform less scans through the transaction database. For larger databases this would also happen for the other depths, since the derivation of all DIs requires no scan through the database at all.

6 Related Work

6.1 Concise Representations

In the literature, there exist already a number of concise representations for frequent itemsets. The most important ones are *closed itemsets*, *free itemsets*, and *disjunction-free itemsets*. We compare the different concise representations with the NDI-representation.

Free sets [6] or *Generators* [11] An itemset I is called *free* if it has no subset with the same support. We will denote the set of all frequent free itemsets with $FreqFree$. In [6], the authors show that freeness is anti-monotone; the subset of

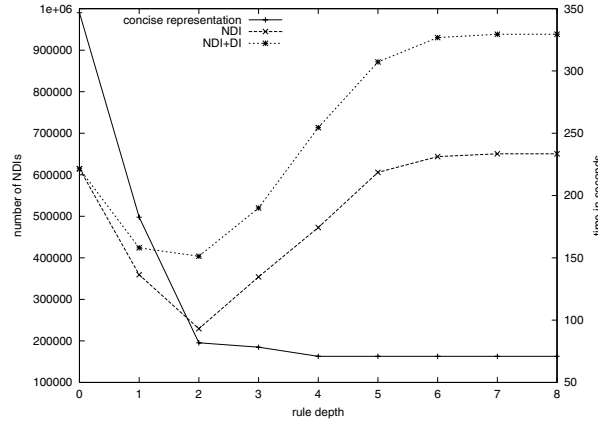


Fig. 4. Strength of deduction rules

a free set must also be free. *FreqFree* itself is not a concise representation for the frequent sets, unless if the set $\text{Border}(\text{FreqFree}) := \{I \subseteq \mathcal{I} \mid \forall J \subset I : J \in \text{FreqFree} \wedge I \notin \text{FreqFree}\}$ is added [6]. We call the concise representation consisting of these two sets *ConFreqFree*. Notice that free sets [6] and generators [13,11] are the same.

Disjunction-free sets [7] or *disjunction-free generators* [11] *Disjunction-free* sets are essentially an extension of free sets. A set I is called disjunction-free if there does not exist two items i_1, i_2 in I such that $\text{support}(I) = \text{support}(I - \{i_1\}) + \text{support}(I - \{i_2\}) - \text{support}(I - \{i_1, i_2\})$. This rule is in fact our rule $\mathcal{R}_I(I - \{i_1, i_2\})$. Notice that free sets are a special case of this case, namely when $i_1 = i_2$. We will denote the set of frequent disjunction-free sets by *FreqDFree*. Again, disjunction-freeness is anti-monotone, and *FreqDFree* is not a concise representation of the set of frequent itemsets, unless we add the border of *FreqDFree*. We call the concise representation containing these two sets *ConFreqDFree*.

Closed itemsets [13] Another type of concise representation that received a lot of attention in the literature [5,14,15] are the closed itemsets. They can be introduced as follows: the *closure* of an itemset I is the largest superset of I such that its support equals the support of I . This superset is unique and is denoted by $cl(I)$. An itemset is called *closed* if it equals its closure. We will denote the set of all frequent closed itemsets by *FreqClosed*. In [13], the authors show that *FreqClosed* is a concise representation for the frequent itemsets.

In the following proposition we give connections between the different concise representations.

Proposition 1. *For every dataset and support threshold, the following inequalities are valid.*

1. The set of frequent closed itemsets is always smaller or equal in cardinality than the set of frequent free sets.
2. The set of NDIs is always a subset of *ConFreqDFree*.

Proof. 1. We first show that $Closed = cl(Free)$.

- $\subseteq$ Let C be a closed set. Let I be a smallest subsets of C such that $cl(I) = C$. Suppose I is not a free set. Then there exist $J \subset I$ such that $support(J) = support(I)$. This rule however implies that $support(J) = support(C)$. This is in contradiction with the minimality of I .
- $\supseteq$ Trivial, since cl is idempotent.

This equality implies that cl is always a surjective function from *Free* to *Closed*, and therefore, $|Free| \geq |Closed|$.

2. Suppose I is not in *ConFreqDFree*. If I is not frequent, then the result is trivially satisfied. Otherwise, this means that I is not a frequent free set, and that there is at least one subset J of I that is also not a frequent free set (otherwise I would be in the border of *FreqDFree*.) Therefore, there exist $i_1, i_2 \in J$ such that $support(J) = support(J - \{i_1\}) + support(J - \{i_2\}) - support(J - \{i_1, i_2\}) = \sigma(J, J - \{i_1, i_2\})$. We now conclude, using Lemma 2, that I is a derivable itemset, and thus not in NDI.

□

Other possible inclusions between the described concise representations do not satisfy, i.e., for some datasets and support thresholds we have $|NDI| < |Closed|$, while other datasets and support thresholds have $|Closed| < |NDI|$. We omit the proof of this due to space limitations. We should however mention that even though *FreqDFree* is always a superset of NDI, in the experiments the gain of evaluating the extra rules is often small. In many cases the reduction of *ConFreqDFree*, which corresponds to evaluating rules up to depth 2 in our framework, is almost as big as the reduction using the whole set of rules. Since our rules are complete, this shows that additional gain is in many cases unlikely.

6.2 Counting Inference

MAXMINER [4] In MAXMINER, Bayardo uses the following rule to derive a lower bound on the support of an itemset:

$$support(I \cup \{i\}) \leq support(I) - \sum_{j \in T} drop(J, j)$$

with $T = I - J$, $J \subset I$, and $drop(J, j) = support(J) - support(J \cup \{j\})$. This derivation corresponds to repeated application of rules $\mathcal{R}_I(I - \{i_1, i_2\})$.

PASCAL [3] In their PASCAL-algorithm, Bastide et al. use counting inference to avoid counting the support of all candidates. The rule they are using to avoid counting is based on our rule $\mathcal{R}_I(I - \{i\})$. In fact the PASCAL-algorithm corresponds to our algorithm when we only check rules up to depth 1, and do not prune derivable sets. Instead of counting the derivable sets, we use the derived

support. Here the same remark as with the *ConFreqDFree*-representation applies; although PASCAL does not use all rules, in many cases the performance comes very close to evaluating all rules, showing that for these databases PASCAL is nearly optimal.

References

1. R. Agrawal, T. Imilienski, and A. Swami. Mining association rules between sets of items in large databases. In *Proc. ACM SIGMOD Int. Conf. Management of Data*, pages 207–216, Washington, D. C., 1993. 74
2. R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. VLDB Int. Conf. Very Large Data Bases*, pages 487–499, Santiago, Chile, 1994. 75
3. Y. Bastide, R. Taouil, N. Pasquier, G. Stumme, and L. Lakhal. Mining frequent patterns with counting inference. *ACM SIGKDD Explorations*, 2(2):66–74, 2000. 75, 84
4. R. J. Bayardo. Efficiently mining long patterns from databases. In *Proc. ACM SIGMOD Int. Conf. Management of Data*, pages 85–93, Seattle, Washington, 1998. 75, 84
5. J.-F. Boulicaut and A. Bykowski. Frequent closures as a concise representation for binary data mining. In *Proc. PaKDD Pacific-Asia Conf. on Knowledge Discovery and Data Mining*, pages 62–73, 2000. 75, 83
6. J.-F. Boulicaut, A. Bykowski, and C. Rigotti. Approximation of frequency queries by means of free-sets. In *Proc. PKDD Int. Conf. Principles of Data Mining and Knowledge Discovery*, pages 75–85, 2000. 75, 82, 83
7. A. Bykowski and C. Rigotti. A condensed representation to find frequent patterns. In *Proc. PODS Int. Conf. Principles of Database Systems*, 2001. 75, 83
8. T. Calders. Deducing bounds on the frequency of itemsets. In *EDBT Workshop DTDM Database Techniques in Data Mining*, 2002. 77
9. S. Hettich and S. D. Bay. *The UCI KDD Archive*. [<http://kdd.ics.uci.edu>]. Irvine, CA: University of California, Department of Information and Computer Science, 1999. 80
10. D. E. Knuth. *Fundamental Algorithms*. Addison-Wesley, Reading, Massachusetts, 1997. 76
11. M. Kryszkiewicz. Concise representation of frequent patterns based on disjunction-free generators. In *Proc. IEEE Int. Conf. on Data Mining*, pages 305–312, 2001. 75, 82, 83
12. H. Mannila and H. Toivonen. Multiple uses of frequent sets and condensed representations. In *Proc. KDD Int. Conf. Knowledge Discovery in Databases*, 1996. 78
13. N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Discovering frequent closed itemsets for association rules. In *Proc. ICDT Int. Conf. Database Theory*, pages 398–416, 1999. 75, 83
14. J. Pei, J. Han, and R. Mao. Closet: An efficient algorithm for mining frequent closed itemsets. In W. Chen, J. F. Naughton, and P. A. Bernstein, editors, *ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, Dallas, TX, 2000. 75, 83
15. M. J. Zaki and C. Hsiao. ChARM: An efficient algorithm for closed association rule mining. In *Technical Report 99-10, Computer Science, Rensselaer Polytechnic Institute*, 1999. 75, 83

16. Z. Zheng, R. Kohavi, and L. Mason. Real world performance of association rule algorithms. In *Proc. KDD Int. Conf. Knowledge Discovery in Databases*, pages 401–406. ACM Press, 2001. [80](#)

Iterative Data Squashing for Boosting Based on a Distribution-Sensitive Distance

Yuta Choki and Einoshin Suzuki

Division of Electrical and Computer Engineering
Yokohama National University, Japan
{choki,suzuki}@slab.dnj.ynu.ac.jp

Abstract. This paper proposes, for boosting, a novel method which prevents deterioration of accuracy inherent to data squashing methods. Boosting, which constructs a highly accurate classification model by combining multiple classification models, requires long computational time. Data squashing, which speeds-up a learning method by abstracting the training data set to a smaller data set, typically lowers accuracy. Our SB (Squashing-Boosting) loop, based on a distribution-sensitive distance, alternates data squashing and boosting, and iteratively refines an SF (Squashed-Feature) tree, which provides an appropriately squashed data set. Experimental evaluation with artificial data sets and the KDD Cup 1999 data set clearly shows superiority of our method compared with conventional methods. We have also empirically evaluated our distance measure as well as our SF tree, and found them superior to alternatives.

1 Introduction

Boosting represents a learning method which constructs a highly accurate classification model by combining multiple classification models, each of which is called a weak hypothesis [4]. It is possible to reduce computational time of boosting by using data squashing [3], which decreases the number of examples in the data set, typically at the sacrifice of accuracy. In order to circumvent this problem, we propose a method of utilizing the probability distribution over training examples provided by boosting methods, here AdaBoost.M2 [4], to data squashing. Data squashing and boosting episodes are alternated in order to employ the examples' weights determined by AdaBoost.M2 to determine thresholds used by the data squashing algorithm. Moreover, we consider distribution of examples in the process by using our projected SVD distance as the distance measure for data squashing. Effects of the iterative data squashing and the distance measure are empirically evaluated through experiments with artificial and real-world data sets.

This paper is structured as follows. In section 2, we review boosting especially AdaBoost.M2 [4], which we employ throughout this paper. Section 3 explains previous research for fast learning based on data squashing. In section 4, we propose our SB (Squashing-Boosting) loop, and evaluate it through experiments in section 5. Section 6 describes concluding remarks.

2 Boosting

The goal of boosting is to construct a “strong learner” which demonstrates high accuracy by combining a sequence of “weak learners” each of which has accuracy slightly higher than random learning. AdaBoost.M2 deals with a classification problem with no less than 3 classes, and constructs each weak hypothesis by transforming the original classification problem to a binary classification problem in terms of an original class. AdaBoost.M2 utilizes, maintains, and modifies an example weight for each example and a model weight for each weak hypothesis. An example weight represents the degree of importance for the example in constructing a weak hypothesis, and is initialized uniformly before learning the first weak hypothesis. An example weight is increased when the obtained weak hypothesis misclassifies the example, and vice versa. A model weight represents the degree of correctness of the corresponding weak hypothesis. We describe a brief outline of AdaBoost.M2 below.

A training data set $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_m, y_m)$ consists of m examples, where the domain of a class y_i is described as $\{1, 2, \dots, c\}$, and $\mathbf{x}_i$ is a vector in an n -dimensional space. An example weight of $(\mathbf{x}_i, y_i)$ is represented as $D_t(i, y)$, where t is the number of rounds and $t = 1, 2, \dots, T$. An initial value for an example weight $D_1(i, y)$ is given by

$$D_1(i, y) = \frac{1}{mc}. \quad (1)$$

An example weight is updated based on the prediction $h_t(\mathbf{x}, y)$ of a weak hypothesis h_t , which is obtained by a weak learning algorithm, for the class y of an instance $\mathbf{x}$. Here $h_t(\mathbf{x}, y)$ outputs 1 or -1 as a predicted class. In this paper, we employ a decision stump which represents a decision tree of depth one as a weak learner. In AdaBoost.M2, a pseudo-loss ϵ_t of a weak hypothesis h_t is obtained for all examples $i = 1, 2, \dots, m$ and all classes $y = 1, 2, \dots, c$.

$$\epsilon_t = \frac{1}{2} \sum_{i=1}^m \sum_{y=1}^c (1 - h_t(\mathbf{x}_i, y_i) + h_t(\mathbf{x}_i, y)) \quad (2)$$

From this, β_t is obtained as follows.

$$\beta_t = \frac{\epsilon_t}{1 - \epsilon_t} \quad (3)$$

The example weight is updated to $D_{t+1}(i, y)$ based on β_t , where Z_t represents the add-sum of all example weights and is employed to normalize example weights.

$$D_{t+1}(i, y) = \frac{D_t(i, y)}{Z_t} \beta_t^{\frac{1}{2}(1+h_t(\mathbf{x}_i, y_i)-h_t(\mathbf{x}_i, y))} \quad (4)$$

$$\text{where } Z_t = \sum_{i=1}^m D_t(i, y) \beta_t^{\frac{1}{2}(1+h_t(\mathbf{x}_i, y_i)-h_t(\mathbf{x}_i, y))} \quad (5)$$

AdaBoost.M2 iterates this procedure T times to construct T weak hypotheses. The final classification model, which is given by (6), predicts the class of each example by a weighted vote of T weak hypotheses, where a weight of a weak hypothesis h_t is given by $\log(1/\beta_t)$.

$$h_{\text{fin}}(\mathbf{x}) = \arg \max_y \sum_{t=1}^T \left(\log \frac{1}{\beta_t} \right) h_t(\mathbf{x}, y) \quad (6)$$

Experimental results show that AdaBoost.M2 exhibits high accuracy. However, it is relatively time-consuming even if it employs a decision stump as a weak learner since its time complexity is given by $O(2^c T m n)$, where n is the number of attributes.

3 Fast Learning Based on Data Squashing

3.1 BIRCH

The main stream of conventional data mining research has concerned how to scale up a learning/discovery algorithm to cope with a huge amount of data. Contrary to this approach, data squashing [3] concerns how to scale down such data so that they can be dealt by a conventional algorithm. Here we show a data squashing method used in BIRCH [11], which represents a fast clustering [5] algorithm. We will modify this data squashing method in the next section as a basis of our method.

Data reduction methods can be classified into feature selection [6] and instance selection [7]. In machine learning, feature selection has gained greater attention since it is more effective in improving time-efficiency. We, however, have adopted instance selection since crucial information for classification is more likely to be lost with feature selection than instance selection, and instance selection can deal with massive data which do not fit in memory.

BIRCH takes a training data set $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m$ as input, and outputs its partition $\gamma_1, \gamma_2, \dots, \gamma_{n+1}$, where each of $\gamma_1, \gamma_2, \dots, \gamma_n$ represents a cluster, and γ_{n+1} is a set of noise. A training data set is assumed to be so huge that it is stored on a hard disk, and cannot be dealt by a global clustering algorithm. Data squashing, which transforms a given data set to a much smaller data set by abstraction, can be considered to speed up learning in this situation. BIRCH squashes the training data set stored on a hard disk to obtain a CF (clustering feature) tree, and applies a global clustering algorithm to squashed examples each of which is represented by a leaf of the tree.

A CF tree represents a height-balanced tree which is similar to a B+ tree [2]. A node of a CF tree represents a CF vector, which corresponds to an abstracted expression of a set of examples. For a set of examples $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{N_\phi}$ to be squashed, a CF vector $\mathbf{CF}_\phi$ consists of the number N_ϕ of examples, the add-sum vector $\mathbf{LS}_\phi$ of examples, and the squared-sum SS_ϕ of attribute values of examples.

$$\mathbf{CF}_\phi = (N_\phi, \mathbf{LS}_\phi, SS_\phi) \quad (7)$$

$$\mathbf{LS}_\phi = \sum_{i=1}^{N_\phi} \mathbf{x}_i \quad (8)$$

$$SS_\phi = \sum_{i=1}^{N_\phi} \|\mathbf{x}_i\|^2 \quad (9)$$

Since the CF vector satisfies additivity and can be thus updated incrementally, BIRCH requires only one scan of the training data set. Moreover, various inter-cluster distance measures can be calculated with the corresponding two CF vectors only. This signifies that the original data set need not be stored, and clustering can be performed with their CF vectors only.

A CF tree is constructed with a similar procedure for a B+ tree. When a new example is read, it follows a path from the root node to a leaf, then nodes along this path are updated. Selection of an appropriate node in this procedure is based on a distance measure which is specified by a user. The example is assigned to its closest leaf if the distance between the new example and the examples of the leaf is below a given threshold L . Otherwise the new example becomes a novel leaf.

3.2 Application of Data Squashing to Classification and Regression

DuMouchel proposed to add moments of higher orders to the CF vector, and applied his data squashing method to regression [3]. Pavlov applied data squashing to support vector machine: a classifier which maximizes margins of training examples under a similar philosophy to boosting [9]. Nakayasu substituted a product-sum matrix for the CF vector, and applied their method to Bayesian classification [8]. They proposed a tree structure similar to the CF tree, and defined the squared add-sum of eigenvalues of the covariance matrix for each squashed example as information loss.

4 Proposed Method

4.1 SB Loop

Data squashing, which we explained in the last section, typically represents a single squashing of a training data set based on a distance measure. Several pieces of work including that of Nakayasu [8] consider how examples are distributed, and can be considered to squash a data set more appropriately than an approach based on a simple distance measure. However, we believe that a single squashing can consider distribution of examples only insufficiently.

In order to cope with this problem, we propose to squash the training data set iteratively. Since a boosting procedure outputs a set of example weights each

of which represents difficulty of prediction of the corresponding example, we considered to use them in data squashing. By using these example weights, we can expect that examples which are difficult to be predicted would be squashed moderately, and examples which are easy to be predicted would be squashed excessively. Alternatively, our approach can be viewed as a speed-up of AdaBoost.M2 presented in section 2 with small degradation of accuracy.

Note that a simple application of a CF tree, which was originally proposed for clustering, would squash examples belonging to different classes to an identical squashed example. We believe that such examples should be processed separately, and thus propose an SF (Squashed-Feature) tree which separates examples belonging to different classes in its root node, and builds a CF tree from each child node. Figure 1 shows an example of an SF tree for a 3-class classification problem.

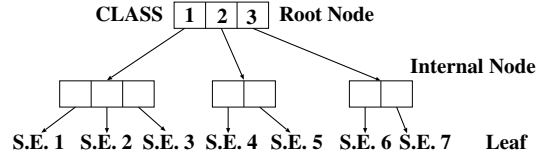


Fig. 1. An example of an Squashed-Feature tree, where an S.E. represents a squashed example

Our approach below iteratively squashes the training data set based on the set of example weights which are obtained from a boosting procedure. Since this squashing and boosting procedure is iterated so that the training data set is squashed appropriately, we call our approach an SB (Squashing-Boosting) loop.

1. Initial data squashing

Given m examples $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_m, y_m)$, obtain p squashed examples $(\mathbf{x}_{\text{sub } 1}, y_{\text{sub } 1}), (\mathbf{x}_{\text{sub } 2}, y_{\text{sub } 2}), \dots, (\mathbf{x}_{\text{sub } p}, y_{\text{sub } p})$ by constructing an SF tree. The threshold L for judging whether an example belongs to a leaf is uniformly settled to L_0 ¹.

2. For $\theta = 1$ until $\theta = \Theta$ step 1

- (a) Application of boosting

Apply AdaBoost.M2 to $(\mathbf{x}_{\text{sub } 1}, y_{\text{sub } 1}), (\mathbf{x}_{\text{sub } 2}, y_{\text{sub } 2}), \dots, (\mathbf{x}_{\text{sub } p}, y_{\text{sub } p})$, and obtain example weights $D_T(1, y_{\text{sub } 1}), D_T(2, y_{\text{sub } 2}), \dots, D_T(p, y_{\text{sub } p})$ of the final round T and a classification model.

- (b) Update of thresholds

For a leaf which represents a set of examples $(\mathbf{x}_{\text{sub } i}, y_{\text{sub } i})$, update its

¹ As we explained in section 3.1, BIRCH employs a threshold L to judge whether an example belongs to a leaf, i.e. a squashed example.

threshold $L(\theta, \mathbf{x}_{\text{sub } i})$ to $L(\theta + 1, \mathbf{x}_{\text{sub } i})$.

$$L(\theta + 1, \mathbf{x}_{\text{sub } i}) = L(\theta, \mathbf{x}_{\text{sub } i}) \frac{D_1(i, y)}{D_T(i, y)} \log a(\theta, i) \quad (10)$$

where $D_1(i, y)$ is given by (1), and $a(\theta, i)$ represents the number of examples which are squashed into the leaf i .

(c) Data Squashing

Construct a novel SF tree from the training examples. In the construction, if a leaf has a corresponding leaf in the previous SF tree, use $L(\theta + 1, \mathbf{x}_{\text{sub } i})$ as its threshold. Otherwise, use L_0 as its threshold.

3. Output the current classification model.

In each iteration, a squashed example with a large example weight is typically divided since we employ a smaller threshold for the corresponding leaf node. On the other hand, a squashed example with a small example weight is typically merged with other squashed examples since we employ a larger threshold for the corresponding leaf node. We show a summary of our SB loop in figure 2.

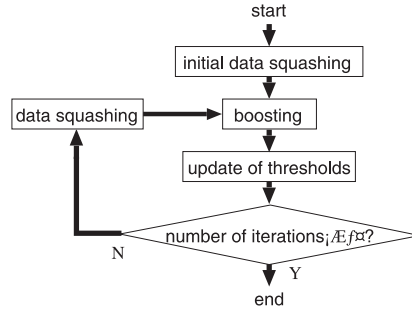


Fig. 2. SB (Squashing-Boosting) loop

4.2 Projected SVD Distance

BIRCH employs a distance measure such as average cluster distance and Euclidean distance in constructing a CF tree [11]. These distance measures typically fail to represent distribution of examples since they neglect interactions among attributes.

In order to circumvent this problem, we propose to store the number of examples N_ϕ , an average vector $\boldsymbol{\mu}_\phi$, and a quasi-product-sum matrix W_ϕ in a node ϕ of our SF tree, where $\boldsymbol{\mu}_\phi = \sum_{i=1}^{N_\phi} \mathbf{x}_i / N_\phi$. A quasi-product-sum matrix, which is given by (11), is updated when a novel example is squashed into its corresponding leaf. The update is done by adding the product-sum matrix of the novel example to the quasi-product-sum matrix. A quasi-product-sum matrix of

an internal node is given by the add-sum of the quasi-product-sum matrices of its child nodes.

$$W_\phi = \begin{pmatrix} g_{11\phi} & \cdots & g_{1j\phi} & \cdots & g_{1m\phi} \\ \vdots & & \ddots & & \vdots \\ g_{i1\phi} & & g_{ij\phi} & & g_{im\phi} \\ \vdots & & & \ddots & \vdots \\ g_{m1\phi} & \cdots & g_{mj\phi} & \cdots & g_{mm\phi} \end{pmatrix} \quad (11)$$

$$g_{ij\phi} = \begin{cases} \sum_k g_{ijk} \\ \text{(for an internal node, where } k \text{ represents an identifier of its child nodes)} \\ x_{fi}x_{fj} + g'_{ij\phi} \\ \text{(for a novel example, where } x_{fi} \text{ represents an attribute value of an attribute } i \text{ for an inputted example } f, \text{ and } g'_{ij\phi} \text{ represents the original value of a squashed example } \phi) \end{cases} \quad (12)$$

Our projected SVD distance $\Delta(\mathbf{x}_i, k)$ between an example $\mathbf{x}_i$ and a squashed example k is defined as follows.

$$\Delta(\mathbf{x}_i, k) = (\mathbf{x}_i - \boldsymbol{\mu}_k)^t S_k^{-1} (\mathbf{x}_i - \boldsymbol{\mu}_k) \quad (13)$$

where S_k represents the quasi-covariance matrix obtained from W_k .

$$S_k = \begin{pmatrix} Cov(11k) & \cdots & Cov(1jk) & \cdots & Cov(1mk) \\ \vdots & & \ddots & & \vdots \\ Cov(i1k) & & Cov(ijk) & & Cov(imk) \\ \vdots & & & \ddots & \vdots \\ Cov(m1k) & \cdots & Cov(mjk) & \cdots & Cov(mm k) \end{pmatrix} \quad (14)$$

$$\text{where } Cov(ijk) = \frac{g_{ijk}}{N_k} - E(ik)E(jk) \quad (15)$$

$$\text{and } E(ik) \text{ is the } i\text{th element of } \boldsymbol{\mu}_k \quad (16)$$

Our projected SVD distance requires the inverse matrix of S , and we use singular value decomposition [10] for this problem. In the method, S is represented as a product of three matrices as follows.

$$S = U \cdot \begin{pmatrix} z_1 & & & \mathbf{0} \\ & z_2 & & \\ & & \cdots & \\ \mathbf{0} & & & z_n \end{pmatrix} \cdot V \quad (17)$$

where U and V are orthogonal matrices. Consider two vectors $\mathbf{x}$, $\mathbf{b}$ which satisfy $S \cdot \mathbf{x} = \mathbf{b}$. If S is singular, there exists a vector $\mathbf{x}'$ which satisfies $S \cdot \mathbf{x}' = \mathbf{0}$.

In general, there are an infinite number of $\mathbf{x}$ which satisfies $S \cdot \mathbf{x} = \mathbf{b}$, and we choose the one with minimum $\|\mathbf{x}\|^2$ as a representative. For this we use

$$\mathbf{x} = V \cdot [\text{diag}(1/z_j)]U^T \cdot \mathbf{b}, \quad (18)$$

where we settle $1/z_j = 0$ if $z_j = 0$, and $\text{diag}(1/z_j)$ represents a diagonal matrix of which j th element is $1/z_j$. This is equivalent to obtaining $\mathbf{x}$ which minimizes $\|S \cdot \mathbf{x} - \mathbf{b}\|$, i.e. an approximate solution for $S \cdot \mathbf{x} = 0$ [10].

5 Experimental Evaluation

5.1 Experimental Condition

We employ artificial data sets as well as real-world data sets in the experiments. Each of our artificial data sets contains, as classes, four normal distributions with equal variances and the covariances are 0. We show means and variances of the classes in table 1. We varied the number of attributes 3, 5, 10. Each class contains 5000 examples. In the experiments for evaluating our SF tree, the number of examples for each class was settled to 500 in order to investigate on the cases with a small number of examples.

Table 1. Means and variances of classes in the artificial data sets, where μ_i represents the mean of an attribute i for each class

class	μ_1	μ_2	μ_3	μ_4	μ_5	μ_6	μ_7	μ_8	μ_9	μ_{10}	variance
1	-6	2	-9	3	10	5	-4	-10	2	9	7
2	7	-2	0	10	3	-4	4	-7	3	7	9
3	-2	-3	-9	-6	3	8	8	1	-2	-3	5
4	-5	-5	8	5	1	-7	6	6	7	-6	8

We employed the KDD Cup 1999 data set [1], from which we produced several data sets. Since it is difficult to introduce a distance measure of data squashing for a nominal attribute and binary attributes can be misleading in calculating a distance, we deleted such attributes before the experiments. As the result, each data set contains 12 attributes instead of 43. We selected the normal-access class and the two most frequent fraudulent-access classes, and defined a 3-class classification problem. We have generated ten data sets by choosing 10000, 20000, $\dots$ 90000, and 97278 examples from each class.

We measured classification accuracy and computational time using 5-fold cross-validation. For artificial data sets, we have chosen boosting without data squashing and boosting with a single data squashing in order to investigate on effectiveness of our approach. We also evaluated our projected SVD distance by

comparing it with average cluster distance and Euclidean distance. The threshold L was settled so that the number of squashed examples becomes approximately 3% of the number of examples, the number of iterations in boosting was settled to $T = 100$, and the number of iterations of data squashing in our approach was $\Theta = 3$. For real-world data sets, we compared our projected SVD distance with average cluster distance. We omitted Euclidean distance due to its poor performance in the artificial data sets.

In the experiments for evaluating tree structures in data squashing, we employed our SF tree and a tree which squashes examples without class information. The experiments were performed with average cluster distance and Euclidean distance since we considered that the small number of examples favors simple distance measures. Since the latter tree can squash examples of different classes into an example, the class of a squashed example was determined by a majority vote. In these experiments, we settled L so that the number of squashed examples becomes approximately 10% of the number of examples, and we used $T = 100$, $\Theta = 5$.

5.2 Experimental Results and Analysis

Artificial Data Sets We show the results with our projected SVD distance in figure 3. From the figure, we see that our SB loop, compared with boosting with single data squashing, exhibits higher accuracy (approximately 8 %) though its computational time is 5 to 7 times longer for almost all data sets. These results show that a single data squashing fails to squash data appropriately, while our SB loop succeeds in doing so by iteratively refining the squashed data sets. Moreover, degradation of accuracy for our approach, compared with boosting without data squashing is within 3 % except for the case of 10 attributes, and our method is 5 to 6 times faster. These results show that our data squashing is effective in speeding-up boosting with a little sacrifice in accuracy.

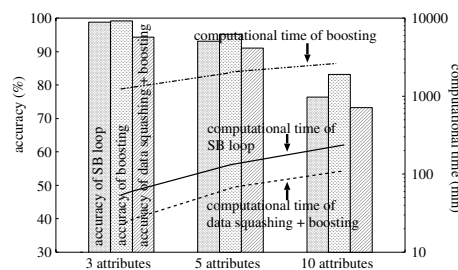


Fig. 3. Effect of SB loop with projected SVD distance for the artificial data sets

We also show the results with average cluster distance and Euclidean distance in figure 4. The figure shows that our approach is subject to large degradation of accuracy compared with boosting, especially when Euclidean distance

is employed. These results justify our projected SVD distance, which reflects distribution of examples to distance.

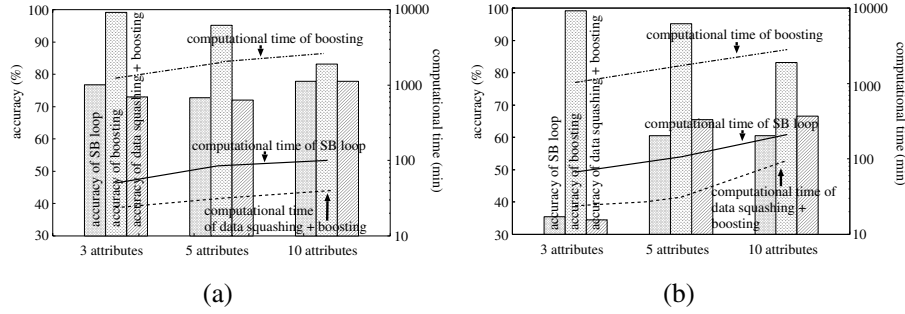


Fig. 4. Effect of SB loop for the artificial data sets with average inter-cluster distance (a) and Euclidean distance (b)

Real-World Data Sets We show experimental results with our projected SVD distance and average cluster distance in figure 5. In both cases, compared with boosting with single data squashing, SB loop exhibits approximately 8 % of improvement in accuracy in average though its computational time is approximately 4 to 6 times longer. Compared with boosting without data squashing, when our projected SVD distance is employed, our approach shortens computational time at most to 1/35 with a small degradation in accuracy. Moreover, the accuracy of our SB loop is no smaller than 92 % when our projected SVD distance is employed. The good performance of our approach can be explained from characteristics of the data set. In the data set, two attributes have large variances which are more than 10000 times greater than the variances of the other attributes. Therefore, data squashing is practically performed in terms of these attributes, and is relatively easier than the cases with the artificial data sets. Moreover, these attributes are crucial in classification since our approach sometimes improves accuracy of boosting without data squashing.

Effectiveness of an SF Tree We show results with the artificial data sets in terms of tree structures and distance measures in figure 6. Regardless of distance measures, our SF tree ((a) and (c)) typically exhibits high accuracy with our SB loop. We attribute the reason to appropriate data squashing. On the contrary, neglecting class information in data squashing ((b) and (d)) typically lowers accuracy, especially when we use data squashing iteratively. We consider that these results justify our SF tree.

In terms of average computational time, our SF tree is approximately five times longer than the tree which neglects class information with average cluster

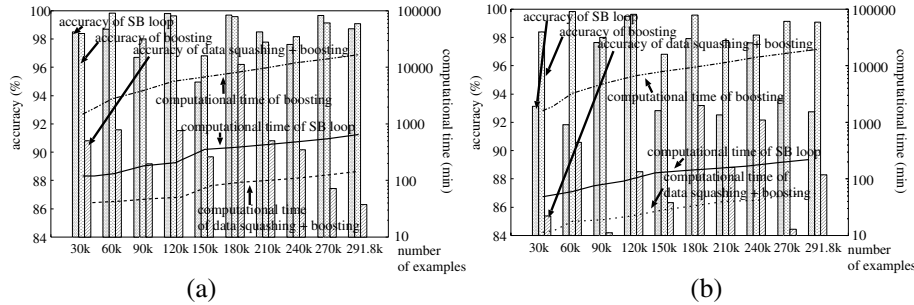


Fig. 5. Results of the KDD Cup 1999 data with projected SVD distance (a) and average cluster distance (b)

distance, and is almost equivalent to the tree which neglects class information with Euclidean distance. This can be explained by the number of squashed examples since our SF tree has, in average, approximately 4.95 and 1.15 times of squashed examples than the tree which neglects class information.

6 Conclusion

The main stream of conventional data mining research has concerned how to scale up a learning/discovery algorithm to cope with a huge amount of data. Contrary to this approach, data squashing [3] concerns how to scale down such data so that they can be dealt by a conventional algorithm.

Our objective in this paper represents a speed-up of boosting based on data squashing. In realizing the objective, we have proposed a novel method which iteratively squashes a given data set using example weights obtained in boosting. Moreover, we have proposed, in data squashing, the projected SVD distance measure, which tries to reflect distribution of examples to distance. Lastly, our SF tree considers class information in data squashing unlike the CF tree used in BIRCH [11].

We experimentally compared our approach with boosting without data squashing and boosting with a single data squashing using both artificial and real-world data sets. Results show that our approach speeds-up boosting 5 to 6 times while its degradation of accuracy was typically less than approximately 3 % for artificial data sets. Compared with boosting with a single data squashing, our approach requires 5 to 7 times of computational time, but improves accuracy approximately 8 % in average. For the real-world data sets from the KDD Cup 1999 data set, our projected SVD distance exhibits approximately 8 % of higher accuracy in average compared with average cluster distance, while the required computational time is almost the same. Considering class information in our SF tree improves accuracy approximately 2.4 % to 27 % in average when the number of examples is small.

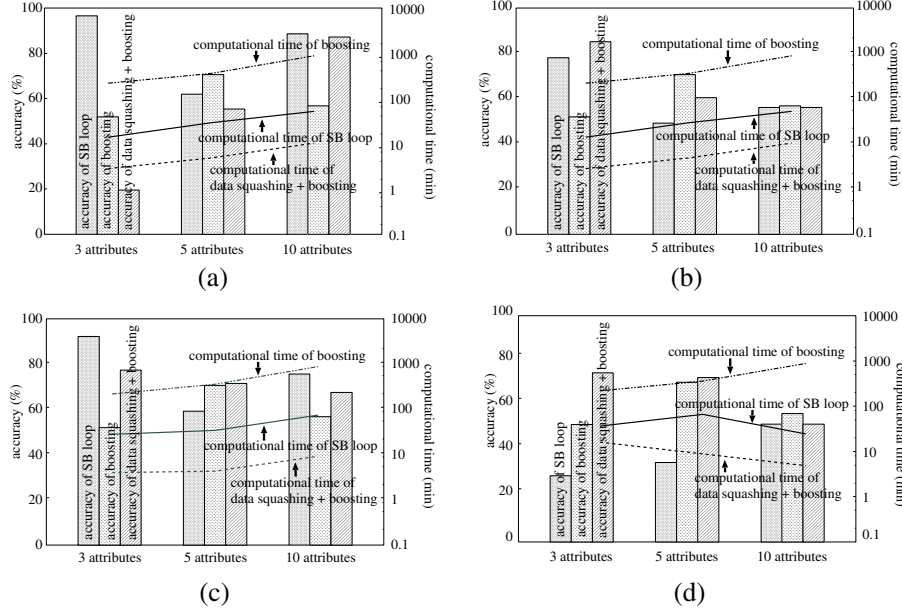


Fig. 6. Effectiveness of an SF tree with average cluster distance (a), a tree which ignores class information with average cluster distance (b), an SF tree with Euclidean distance (c), and a tree which ignores class information with Euclidean distance (d)

References

1. Bay, S.: *UCI KDD Archive*, <http://kdd.ics.uci.edu/>, Dept. of Information and Computer Sci., Univ. of California Irvine. (1999). 93
2. Comer, D.: The Ubiquitous B-Tree, *ACM Computing Surveys*, Vol. 11, No. 2, pp. 121–137 (1979). 88
3. DuMouchel, W. et al.: Squashing Flat Files Flatter, *Proc. Fifth ACM Int'l Conf. on Knowledge Discovery and Data Mining (KDD)*, pp. 6–15 (1999). 86, 88, 89, 96
4. Freund, Y. and Schapire, R. E.: Experiments with a New Boosting Algorithm, *Proc. Thirteenth Int'l Conf. on Machine Learning (ICML)*, pp. 148–156 (1996). 86
5. Kaufman, L. and Rousseeuw, P. J.: *Finding Groups in Data*, Wiley, New York (1990). 88
6. Liu, H. and Motoda, H.: *Feature Selection*, Kluwer, Norwell, Mass. (1998). 88
7. Liu, H. and Motoda, H. (eds.): *Instance Selection and Construction for Data Mining*, Kluwer, Norwell, Mass. (2001). 88
8. Nakayasu, T., Suematsu, N., and Hayashi, A.: Learning Classification Rules from Large-Scale Databases, *Proc. 62th Nat'l Conf. of Information Processing Society of Japan*, Vol. 2, pp. 23–24 (2001, in Japanese). 89
9. Pavlov, D., Chudova, D., and Smyth, P.: Towards Scalable Support Vector Machines Using Squashing, *Proc. Sixth ACM Int'l Conf. on Knowledge Discovery and Data Mining (KDD)*, pp. 295–299 (2000). 89

10. Press, W. H. et al.: *Numerical Recipes in C - Second Edition*, Cambridge Univ. Press, Cambridge, U. K. (1992). 92, 93
11. Zhang, T., Ramakrishnan, R., and Livny, M.: BIRCH: An Efficient Data Clustering Method for Very Large Databases, *Proc. 1996 ACM SIGMOD Int'l Conf. on Management of Data (SIGMOD)*, pp. 103–114 (1996). 88, 91, 96

Finding Association Rules with Some Very Frequent Attributes

Frans Coenen and Paul Leng

Department of Computer Science, The University of Liverpool
Chadwick Building, P.O. Box 147, Liverpool L69 3BX, England
{frans,phl}@csc.liv.ac.uk

Abstract. A key stage in the discovery of Association Rules in binary databases involves the identification of the “frequent sets”, i.e. those sets of attributes that occur together often enough to invite further attention. This stage is also the most computationally demanding, because of the exponential scale of the search space. Particular difficulty is encountered in dealing with very densely-populated data. A special case of this is that of, for example, demographic or epidemiological data, which includes some attributes with very frequent instances, because large numbers of sets involving these attributes will need to be considered. In this paper we describe methods to address this problem, using methods and heuristics applied to a previously-presented generic algorithm, Apriori-TFP. The results we present demonstrate significant performance improvements over the original Apriori-TFP in datasets which include subsets of very frequently-occurring attributes.

Keywords: Association Rules, Frequent sets, Dense data

1 Introduction

Association rules [2] are observed relationships between database attributes, of the form “if the set of attributes A is found in a record, then it is likely that B will be found also”. More formally, an association rule R takes the form $A \rightarrow B$, where A, B are disjoint subsets of the attribute set. Usually, a rule is thought to be “interesting” if at least two properties apply. First, the *support* for the rule, that is, the number of records within which the association can be observed, must exceed some minimum threshold value. Then, if this is the case, the *confidence* in the rule, which is the ratio of its support to that of its antecedent, must also exceed a required threshold value. Other measures, (e.g. *lift* [6], or *conviction* [7]) have also been proposed to provide further definition of the interest in a potential rule. All work on association-rule mining, however, has recognised that the identification of the *frequent* sets, the support for which exceeds the required threshold, is a necessary stage, and also that this is computationally the most demanding, because of the inherently exponential nature of the search space.

Most work on association rule discovery has focused in particular on its application in supermarket shopping-basket analysis. This, however, is not the most

demanding problem domain. In other applications, such as census data, there may be many attributes (for example “female”, “married”, etc.) which occur in a very high proportion of records. The correspondingly high frequency of combinations including these attributes gives rise to very large *candidate sets* of attributes that potentially exceed the support threshold, causing severe problems for methods such as Apriori [3]. The problem can be reduced by setting the support threshold at a sufficiently high level, but this will risk eliminating potentially interesting combinations of less common attributes.

In this kind of data, associations involving only the most common attributes are likely to be obvious and therefore not genuinely interesting. In this paper, therefore, we describe a method which seeks to identify only those frequent sets which involve at least one “uncommon” attribute. This reduction makes it possible to employ heuristics which can reduce the computational cost significantly. We describe the algorithms we have used, and present results demonstrating the performance gains achieved in dealing with data which includes a proportion of very frequent attributes.

2 Finding Frequent Sets

We will begin by reviewing the well-known and seminal “Apriori” algorithm of [3]. Apriori examines, on successive passes of the data, a candidate set C_k of attribute sets the members of which are all those sets of k attributes which remain in the search space. Initially, the set C_1 consists of the individual attributes. Then, the k th cycle proceeds as follows (for $k=1,2,..$ until $C_k =$ empty set):

1. Perform a pass over the database to compute the support for all members of C_k .
2. From this, produce the set L_k of frequent sets of size k .
3. Derive from this the candidate set C_{k+1} , using the *downward closure* property, i.e. that all the k -subsets of any member of C_{k+1} must be members of L_k .

Apriori and related algorithms work reasonably well when the records being examined are relatively sparsely populated, i.e. few items occur very frequently, and most records include only a small number of items. When this is not so, however, the time for the algorithm increases exponentially. The principal cost arises in step 1, above, which requires each database record to be examined, and all its subsets that are members of the current candidate set to be identified. Clearly, the time for this procedure will in general depend both on the number of attributes in the record and on the number of candidates in the current set C_k . If the database is densely-populated, the size of the candidate sets may become very large, especially in the early cycles of the algorithm, before the “downward closure” heuristic begins to take effect. Also, any record including a large number of attributes may require a large subset of the candidate set to be examined: for

example, the extreme case of a record containing all attributes will require all candidates in the current set to be inspected.

A number of strategies have been adopted to reduce some of the inherent performance costs of association-rule mining in large, dense, databases. These include methods which reduce the scale of the task by working initially with a subset or sample of the database [14],[15]; methods which look for *maximal* frequent sets without first finding all their frequent subsets [4],[5]; methods which redefine the candidate set dynamically [7], [10], and methods which are optimised for dealing with main-memory-resident data [1]. No method, however, offers a complete solution to the severe scaling of the problem for dense data.

A particular case that causes difficulty for Apriori and related methods occurs when the attribute set includes a possibly quite small subset of very frequently-occurring attributes. Suppose, for example, there is a subset F of attributes each of which is present in about 50% of all records. Then if the support threshold is set at 0.5%, which may be necessary to find interesting combinations of scarce attributes, it is likely that most of the combinations of attributes in F will still be found in L_7 . Not only will this lead to a large number of database passes to complete the count, but also the candidate sets will be inflated by the continuing presence of these combinations. For example, if F contains only 20 attributes, C_8 is likely to include about 125,000 candidates which are combinations of these only, and for 40 attributes, the size of C_8 may exceed 76×10^6 .

We have described previously [12] a method we have developed which reduces two of the performance problems of Apriori and related algorithms: the high cost of dealing with a record containing many attributes, and the cost of locating relevant candidates in a large candidate-set. In the following section we will briefly summarise this method, before going on to describe adaptations of this approach to address the problems of datasets such as the one outlined above.

3 Computing via Partial Support

The methods we use begin by using a single database pass to restructure the data into a form more useful for subsequent processing, while at the same time beginning the task of computing support-counts. Essentially, each record in the database takes the form of a set of attributes i that are represented in the record. We reorganise these sets to store them in lexicographic order in the form of a set-enumeration tree [13]. Records that are duplicated in the database occur only once in the tree, and are stored with an associated incidence-count. As the tree is being constructed, it is also easy and efficient to add to this count, for each set i stored, the number of times i occurs as a subset of a record which *follows* i in the set ordering. We use the term *P-tree* to refer to this set-enumeration tree with its associated partial support-counts. A detailed description of the algorithm for building the *P-tree* is given in [9]. The construction is simple and efficient, and both the tree size and construction time scale linearly with the database size and density.

The P -tree thus constructed contains all the sets present as records in the original data, linked into a structure in which each subtree contains all the lexicographically following supersets of its parent node, and the count stored at a parent node incorporates the counts of its subtree nodes. The concept is similar to that of the FP -tree of [10], with which it shares some properties, but the P -tree has a simpler and more general form which offers some advantages. In particular, it is easy and convenient to convert the P -tree to an equivalent tabular form, explained below. Thus, although for simplicity our experiments use data for which the P -tree is store-resident, the structures are simple enough also to enable straightforward and efficient implementations in cases when this is not so. Results presented in [12] also show the memory requirement for the P -tree to be significantly less than that for the FP -tree.

The generality of the P -tree makes it possible to apply variants of many existing methods to this to complete the summation of required support-totals. We have experimented with a method which applies the Apriori procedure outlined in the previous section to the tabulated P -tree nodes, rather than to the records in the original database. We store candidates whose support is to be counted in a second set-enumeration structure, the T -tree, arranged in the opposite order to that of the P -tree. Each subtree of the T -tree stores predecessor-supersets of its root node. The significance of this is that it localises efficiently those candidates which need to be considered when we examine the subsets of a P -tree node. The algorithm we use, which we call Apriori-TFP, is described in detail in [12]. Apriori-TFP follows the Apriori methodology of performing repeated passes of the data, in each of which the support for candidates of size k is counted. The candidates are stored on the T -tree, which is built level by level as each pass is performed, and pruned at the end of the pass to remove candidates found not to be frequent. Each pass involves a complete traversal of the P -tree. Because the order in which this happens is irrelevant, it is possible to store the P -tree in a node-by-node tabular form. An entry in this table representing a set $ABDFG$, say, present in the tree as a child of $ABDF$, would be stored in a form $ABDF.G$, (with an associated count). When this entry is examined in the second pass, for example, this count is added to the totals stored in the T -tree for the pairs AG , BG , DG and FG , all of which, assuming they remain in the candidate set, will be found in the branch of the T -tree rooted at G . Notice that, in contrast with Apriori, we do not need to count the other subsets AB , BD , etc., at this point, as the relevant totals will be included in the count for the parent set $ABDF$ and counted when that node and its ancestors are processed. It is these properties that lead to the performance gain from the method. The advantage gained increases with increasing density of the data.

4 Heuristics for Dense Data

Notwithstanding the gains achievable from using the P -tree, very dense data poses severe problems for this as for other methods. Its performance can be improved by ordering the tree by descending frequency of attributes [8]. The ex-

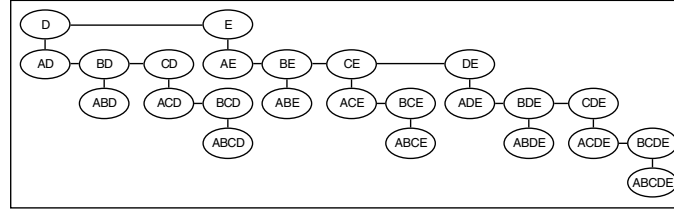


Fig. 1. Incomplete T-tree for attributes $\{(A, B, C), D, E\}$

tent of the improvement depends on the nature of the data; for the “mushroom” dataset of [16], for example, the gain was about 50%. This heuristic points us to a strategy for dealing with data including some very frequent attributes. Suppose that there is a subset F of such attributes, and let us assume that we have no interest in deriving rules which include only members of F . We still, of course, wish to investigate rules which associate subsets of F with one or more of the less common attributes, and for this purpose will need to compute the support for at least some of the sets in the power set of F .

We begin by constructing the P -tree as before, with an ordering of attributes that places the members of F first. From this, we construct an *incomplete* T -tree, to include only candidate sets that contain at least one attribute not in F . The form of this tree is illustrated in Figure 1, for a set of attributes $\{A, B, C, D, E\}$, of which A , B and C are in the set F . Note again that the tree is ordered so that each subtree contains only the lexicographically preceding supersets of its parent node. The branches rooted at A , B and C , which would be present in the full T -tree, are in this case omitted.

Apart from the omission of the sets derived only from F , the tree shown in Figure 1 is complete. In the actual implementation, however, it is constructed so as to contain, finally, only the frequent sets. The algorithm Apriori-TFP for doing this builds the tree level by level via successive passes of the P -tree table. In each pass, the level k currently being considered contains the candidate set C_k defined as for Apriori. The support for each set in C_k is counted and attached to the corresponding node of the T -tree. At the end of the pass, sets found not to be frequent are removed, leaving the sets in L_k on the tree, and the next level of the tree is built using the Apriori heuristic.

Although the tree we have illustrated will not contain the support-counts for the members of the “very frequent” subset F , we may still require to know these values. Firstly, when a level is added to the tree, we wish to include nodes only if all their subsets are frequent, including those subsets that contain only members of F and thus are not in the tree. Finally, also, we will need to know the support of all of the subsets of F that are included as subsets of sets in the T -tree so that we can compute the confidence for each possible association that may result from sets in the T -tree. Because of the way we have constructed the P -tree, however, there is a very efficient procedure for computing these support-totals exhaustively, provided F is small enough for all these counts to be contained

in an array in main memory. Associated with each set i in the P -tree is an incomplete support-count Q_i . A “brute-force” algorithm to compute the final “total” support-counts T_i may be described thus:

Algorithm ETFP (Exhaustive Total- from Partial- supports)

```

for each node  $j$  in  $P$ -tree do
  if  $j \subseteq F$  then
    begin  $k = j$  - parent ( $j$ );
      for each subset  $i$  of  $j$  with  $k \subseteq i$  do
        add  $Q_j$  to  $T_i$ ;
    end
end

```

Notice again that because the incomplete support-count Q_i for a parent node i incorporates the counts for its children, we need only consider subsets of a node that are not subsets of its parent. If the counts T_i can be contained in a simple array (initialised to zero), the count-update step is trivial. Also note that because of the ordering of the P -tree, the sets to be counted are clustered at the start, so not all the tree need be traversed to find them.

We can now describe methods for computing all the support-totals we need to determine the frequent sets. In every case, we begin by constructing the P -tree.

Method 1

1. Use Algorithm ETFP to count the support for all subsets of F .
2. Convert the P -tree to tabular form, omitting subsets of F .
3. Use Algorithm Apriori-TFP to find all the frequent sets that include at least one attribute not in F , storing these in an incomplete T -tree of the form illustrated in Figure 1.

As each new level is added to the T -tree during step 3, we include candidates only if all their subsets are frequent, which can be established by examining the support counts stored at the current level of the T -tree and, for the subsets of F , in the array created in step 1. However, because of the high probability that the latter are indeed frequent, it may be more efficient to assume this; thus:

Method 2

As Method 1, but in step 3, assume that all subsets of F are frequent. This may occasionally result in candidates being added unnecessarily to the tree, but will reduce the number of checks required as each new level is added.

Notice that in fact these methods do find all the frequent sets, including those that contain only attributes from F . In comparison with the original Apriori-TFP, the cost of building the T -tree is reduced both because of the smaller number of candidates in the reduced T -tree, and because when examining sets in the P -tree table, only those subsets containing a “scarce” attribute need be considered (and subsets of F are left out of the table). These gains more than

compensate for the cost of exhaustively counting support for the very frequent attributes by the efficient ETFP algorithm, provided the size of F is relatively small. If there are more than about 20 very frequent attributes, however, the number of combinations of these becomes too great for exhaustive counting to be feasible. In this case, a third method may be applied — method 3.

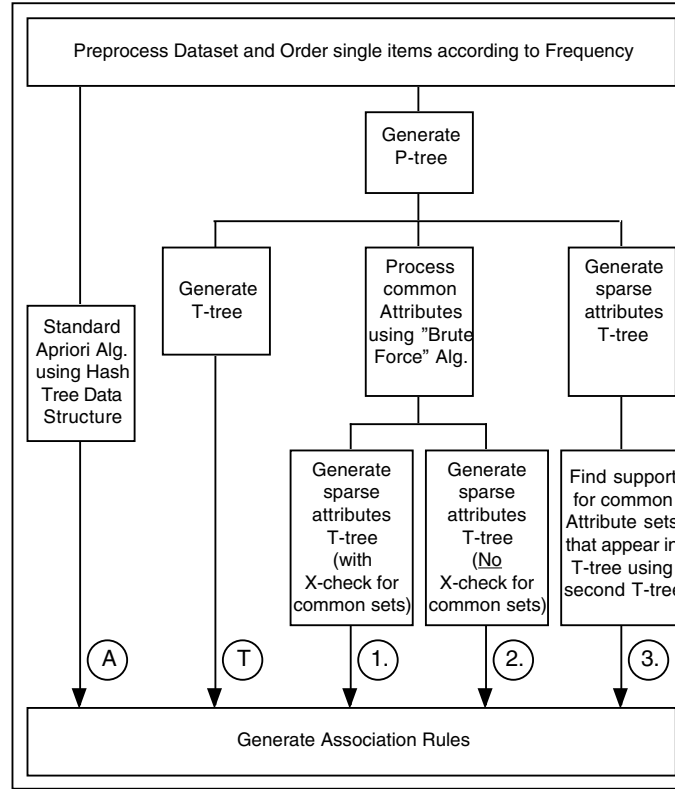


Fig. 2. Summary of methods used

Method 3

1. Convert the P -tree to tabular form.
2. Use Algorithm Apriori-TFP, as before, to find all the frequent sets that include at least one attribute not in F , storing these in a T -tree. As with Method 2, we assume all subsets of F are (probably) frequent.
3. Traverse the T -tree to extract all the subsets of F that are included in frequent sets with scarce attributes, storing these on a second T -tree.
4. Perform a single pass of the part of the P -tree table that contains the subsets of F , to count the support for those that are left in the second T -tree.

The effect of this is to count the support of subsets of F only if they are associated in frequent sets with the “scarce” attributes. Because this will not be so for most of the larger subsets, this is likely to be feasible even when F is too large to be exhaustively enumerated. The result at step 2 of this method is similar to what would be achieved using the multiple support threshold algorithm MSapriori [11], with the higher support threshold set at 100%. In this case, however, MSapriori would not count the support for the subsets of F needed to complete the derivation of rules. In other respects, also, MSapriori shares the problems of the original Apriori, with further overheads added by the need to process multiple support thresholds.

The three methods above are summarised in Figure 2, in comparison with the “standard” Apriori algorithm (labelled A), and our original Apriori-TFP algorithm (labelled T). In the following section we will compare the performance of the 5 methods outlined in Figure 2.

5 Results

To examine the performance of the methods we have described, we have applied them to datasets generated using the QUEST generator described in [3]. This defines parameters N , the number of attributes of the data, T , the average number of attributes present in a record, and I , the largest number of attributes expected to be found in a frequent set. For the purpose of these experiments, we began by generating a dataset of 250,000 records with $N = 500$, $T = 10$, and $I = 5$. This gives rise to a relatively sparse dataset, typical of that used in experiments on shopping-basket data. To create a more challenging dataset, we also performed a second generation of 250,000 records, with $N = 20$, $T = 10$, and $I = 5$. The two sets were merged, record-by-record, to create a final dataset of 250,000 records with 520 attributes, within which the first 20 attributes are each likely to occur in about 50% of all records. In a real case, of course, the distinction between the “very common” and “less common” is likely to be less clear-cut, and may be influenced by a subjective assessment of which attributes are of most interest. For methods 1 and 2, however, the number of attributes which can be included in the set F is limited by the size of the array needed to store all their combinations, a practical limit of about 20. We examine below the case in which F is larger than this.

Figure 3 illustrates the performance of the basic Apriori-TFP algorithm on this dataset, in comparison with the original Apriori and with the FP-growth algorithm of [10], for varying support thresholds down to 1%. All methods are our own implementations (in Java), intended to give as fair a comparison as we are able of the approaches. The graphs show the total time required in each case to determine the frequent sets, for Apriori (labelled A), FP-growth (labelled F) and Apriori-TFP (labelled T). In the latter case, this includes the time required to construct the P -tree. As we have shown in earlier work [12], Apriori-TFP strongly outperforms the original Apriori. In the present case, the difference is particularly extreme, because of the higher density of data produced as a

result of the inclusion of the 20 “very frequent” attributes. The consequent large candidate sets, with the typical record-length of 20 attributes, requires in Apriori large numbers of subsets to be counted and a high level of expensive hash-tree traversal. In our implementation, the time required for this becomes prohibitive for the lower support thresholds. As we have shown in [8], the improvement shown by Apriori-TFP is maximised when, as in this case, the most common attributes are placed at the start of the set order.

The comparison with FP-growth is much closer, as we would expect from methods which share similar advantages. This is shown more clearly in Figure 4, which also illustrates the performance of the three variants we described in the previous section, for the dataset described above, (this time with a linear scale on the graph). All methods show the relatively sharp increase in time required as the support threshold is dropped and the number of candidates to be considered increases correspondingly. As can be seen, FP-growth matches the basic Apriori-TFP method for higher support thresholds, but the latter begins to perform better at lower thresholds. We believe this is because of the overheads of the relatively complex recursive tree-generation required by FP-growth, compared with the rather simple iteration of Apriori-TFP.

Curve 1 in Figure 4 shows, in comparison, the time taken when the combinations of the 20 most common attributes are counted exhaustively (Method 1). For high support thresholds, this is slower than Apriori-TFP, because we are counting many combinations which are not in fact frequent. However, for support thresholds below about 3%, in this data, this is more than compensated by the more efficient counting method of ETFP, leading to a significant performance gain. The performance of Method 2 (Curve 2 in Figure 4) is similar; in fact, Method 2 slightly outperformed Method 1, although the difference was not significant and is not apparent in the graph. At these support thresholds, the

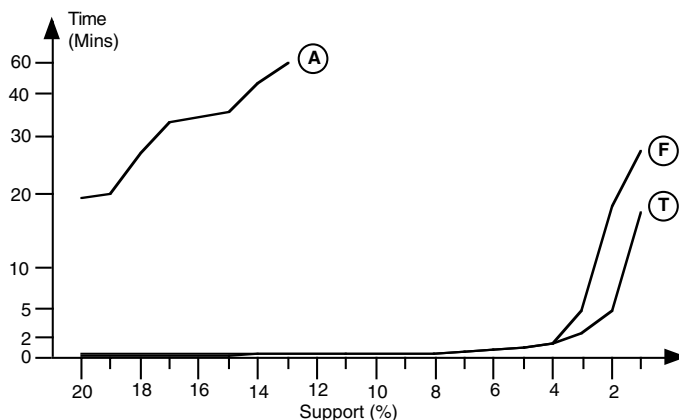


Fig. 3. Comparison of Apriori (A), FP-growth (F) and Apriori-TFP (T) (*T10.I5.D250000.N20* merged with *T10.I5.D250000.N500*)

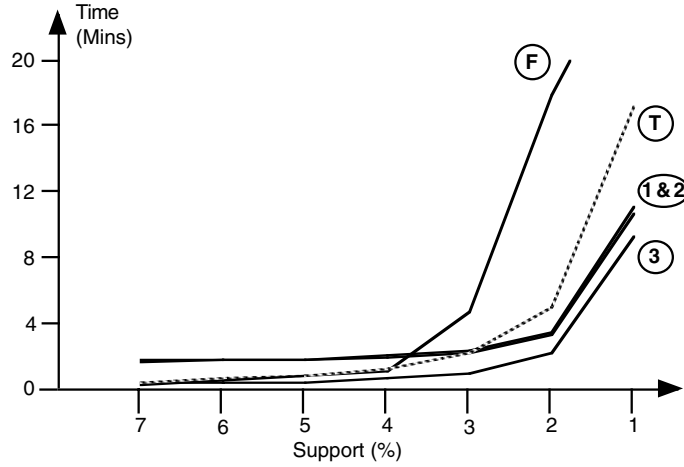


Fig. 4. Performance for *T10.I5.D250000.N20* merged with *T10.I5.D250000.N500*

assumption that all relevant combinations of the “very frequent” attributes are frequent is almost always correct, but the gain from this is slight because, with relatively small candidate sets, the cost of cross-checking is not a major factor.

Curve 3 in Figure 4 shows the performance of Method 3, in which the support for combinations of the 20 most frequent attributes was counted only when they appeared in frequent sets with less common attributes. Here, the results show a consistent performance gain over all the other methods.

Finally, we examine the performance of our methods in cases where the number of “very frequent” attributes is greater. For this purpose, we again began with the dataset of 250,000 records with $N = 500$, $T = 10$, and $I = 5$. In this case, we merged this with a second set of 250,000 records with $N = 40$, $T = 20$, and $I = 10$. The result is to create a set of records with 540 attributes, the first 40 of which have a frequency of about 50%, and for which the average record-length is about 30 attributes. This relatively dense data is far more demanding for all methods. Figure 5 shows the results of our experiments with this data. Again, the curve labelled T illustrates the performance of the basic Apriori-TFP (in all cases, we have continued the curves only as far as is feasible within the time-frame of the graph).

For more than about 20 attributes, the “Brute Force” algorithm ETFP becomes infeasible, so Curve 1 shows the performance of Method 1, in which only 20 of the 40 most common attributes are counted in this way. This still shows a performance improvement over Apriori-TFP. However, because another 20 of the very frequent attributes are counted in the T -tree, the size of the candidate sets leads, as with other methods, to severe performance scaling as the support threshold is reduced. The same applies to Method 2, which offers a further small advantage as the support threshold is lowered, because with large candidate sets

the amount of checking involved becomes significant. Even so, the gain from this is very slight (less than 2% improvement at the 7% support threshold).

Curve 3 illustrates Method 3, also (for comparison) with only 20 of the 40 most common attributes excluded from the initial T -tree. As we would expect, the curve is similar to that of Figure 4; although the method outperforms the others, the time taken increases rapidly for low support thresholds. This is, of course, because of the very large number of combinations of the very common attributes which are being counted. The final curve, 4, however, shows the results of excluding all 40 very frequent attributes from the initial count. In this case, the initial construction of the T -tree counts only those frequent sets which include at least one of the 500 less common attributes. For support thresholds down to about 3%, there are relatively few of these, so the time to count these sets is low, as is the time required finally to count the support of the combinations of common attributes which are subsets of these frequent sets. Only at a support threshold of 1% does the time start to rise rapidly (although this will still be far less than for the other methods described).

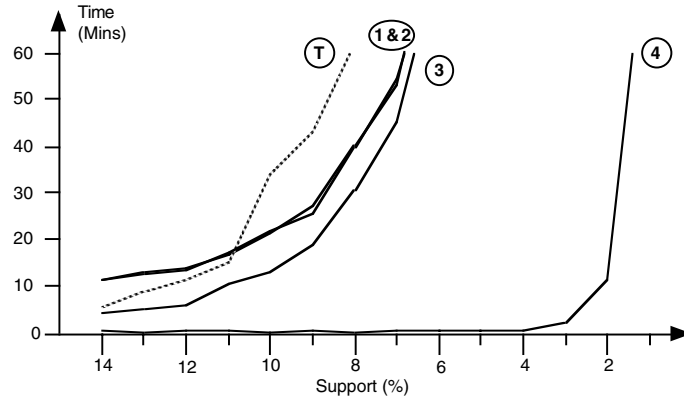


Fig. 5. Performance for $T20.I10.D250000.N40$ merged with $T10.I5.D250000.N500$

6 Conclusions

Much of the work reported in the literature on the discovery of Association rules has focused on the case of shopping-basket analysis, characterised by very large numbers of database attributes (i.e. items available for purchase) but relatively low data density (frequency of individual items and sets of items). It is well-understood that all methods find increasing difficulty in coping with more densely-populated data. We have previously described a method, Apriori-TFP, which performs well in comparison with others, but is also subject to this adverse performance scaling when dealing with high-density data.

In this paper we have described some developments of our approach, optimised to deal with data that includes a proportion of very frequent attributes. This kind of data may be typical, for example, of demographic survey data and epidemiological data, in which some categorical attributes relating to age, gender, etc, have very frequent instances, while others which may be very relevant to the epidemiology are rather infrequent. In this case, we require support thresholds to be set low enough to identify the interesting sets including these scarce attributes.

Our results show, unsurprisingly, that both Apriori-TFP and other methods face problems with this data at low support thresholds. We have shown that the performance of Apriori-TFP can be improved significantly, although not dramatically, by the use of a heuristic which employs exhaustive counting for the most frequent attributes. However, if we make the (not unreasonable) assumption that the only sets that are of interest are those including at least one scarce attribute, then a much more effective adaptation of Apriori-TFP becomes possible. We have shown that this method strongly outperforms Apriori-TFP, enabling identification of the interesting sets in this kind of dense data even at low thresholds.

References

1. Agarwal, R., Aggarwal, C. and Prasad, V. Depth First Generation of Long Patterns. Proc ACM KDD 2000 Conference, Boston, 108-118, 2000 [101](#)
2. Agrawal, R. Imielinski, T. Swami, A. Mining Association Rules Between Sets of Items in Large Databases. SIGMOD-93, 207-216. May 1993 [99](#)
3. Agrawal, R. and Srikant, R. Fast Algorithms for Mining Association Rules. Proc 20th VLDB Conference, Santiago, 487-499. 1994 [100](#), [106](#)
4. Bayardo, R. J. Efficiently Mining Long Patterns from Databases. Proc ACM-SIGMOD Int Conf on Management of Data, 85-93, 1998 [101](#)
5. Bayardo, R. J., Agrawal, R. and Gunopulos, D. Constraint-based rule mining in large, dense databases. Proc 15th Int Conf on Data Engineering, 1999 [101](#)
6. Berry, M. J. and Linoff, G. S. Data Mining Techniques for Marketing, Sales and Customer Support. John Wiley and sons, 1997 [99](#)
7. Brin, S., Motwani, R., Ullman, J. D. and Tsur, S. Dynamic itemset counting and implication rules for market basket data. Proc ACM SIGMOD Conference, 255-256, 1997 [99](#), [101](#)
8. Coenen, F. and Leng, P. Optimising Association Rule Algorithms Using Itemset Ordering. In Research and Development in Intelligent Systems XVIII, (Proc ES2001), eds M. Bramer, F Coenen and A Preece, Springer, Dec 2001, 53-66 [102](#), [107](#)
9. Goulbourne, G., Coenen, F. and Leng, P. Algorithms for Computing Association Rules using a Partial-Support Tree. J. Knowledge-Based Systems 13 (2000), 141-149. (also Proc ES'99.) [101](#)
10. Han, J., Pei, J. and Yin, Y. Mining Frequent Patterns without Candidate Generation. Proc ACM SIGMOD 2000 Conference, 1-12, 2000 [101](#), [102](#), [106](#)
11. Liu, B., Hsu, W. and Ma, Y. Mining association rules with multiple minimum supports. Proc. KDD-99, ACM, 1999, 337-341 [106](#)

12. Coenen, F., Goulbourne, G. and Leng, P. Computing Association Rules Using Partial Totals. Proc PKDD 2001, eds L. De Raedt and A Siebes, LNAI 2168, August 2001, 54-66 101, 102, 106
13. Rymon, R. Search Through Systematic Set Enumeration. Proc. 3rd Int'l Conf. on Principles of Knowledge Representation and Reasoning, 1992, 539-550 101
14. Savasere, A., Omiecinski, E. and Navathe, S. An efficient algorithm for mining association rules in large databases. Proc 21st VLDB Conference, Zurich, 432-444. 1995 101
15. Toivonen, H. Sampling large databases for association rules. Proc 22nd VLDB Conference, 134-145. Bombay, 1996 101
16. UCI Machine Learning Repository Content Summary. <http://www.ics.uci.edu/mlearn/MLSummary.html> 103

Unsupervised Learning: Self-aggregation in Scaled Principal Component Space^{*}

Chris Ding¹, Xiaofeng He¹, Hongyuan Zha², and Horst Simon¹

¹ NERSC Division, Lawrence Berkeley National Laboratory
University of California, Berkeley, CA 94720

² Department of Computer Science and Engineering
Pennsylvania State University, University Park, PA 16802
{chqding,xhe,hdsimon}@lbl.gov
zha@cse.psu.edu

Abstract. We demonstrate that data clustering amounts to a dynamic process of *self-aggregation* in which data objects move towards each other to form clusters, revealing the inherent pattern of similarity. Self-aggregation is governed by connectivity and occurs in a space obtained by a nonlinear scaling of principal component analysis (PCA). The method combines dimensionality reduction with clustering into a single framework. It can apply to both square similarity matrices and rectangular association matrices.

1 Introduction

Organizing observed data into groups or clusters is the first step in discovering coherent patterns and useful structures. This unsupervised learning process (data clustering) is frequently encountered in science, engineering, commercial data mining and information processing. There exists a large number of data clustering methods [13,5] for different situations. In recent decades, unsupervised learning methods related to the principal component analysis (PCA)[14] has been increasingly widely used: the low-dimensional space spanned by the principal components is effective in revealing structures of the observed high-dimensional data. PCA is a coordinate rotation such that the principal components span the dimensions of largest variance. The *linear* transformation preserves the local properties and global topologies, and can be efficiently computed. However, PCA is not effective in revealing nonlinear structures [9,16,17,23,20,21]. To overcome the short-comings of *linear* transformation of PCA, nonlinear PCAs have been proposed, such as principal curves [9], auto-associative networks [16], and kernel PCA [21]. But they do not possess the self-aggregation property. Recently, nonlinear mappings [23,20] have been developed. But they are not primarily concerned with data clustering.

^{*} LBNL Tech Report 49048, October 5, 2001. Supported by Department of Energy (Office of Science, through a LBNL LDRD) under contract DE-AC03-76SF00098

Here we introduce a new concept of *self-aggregation* and show that a nonlinear scaling of PCA leads to a low-dimensional space in which data objects self-aggregate into distinct clusters, revealing inherent patterns of similarity, in contrast to existing approaches. Thus data clustering becomes a *dynamic* process, performing nonlinear dimensionality reduction and cluster formation simultaneously; the process is governed by the connectivity among data objects, similar to dynamic processes in recurrent networks [12,10].

2 Scaled Principal Components

Associations among data objects are mostly quantified by a similarity metric. The scaled principal component approach starts with a nonlinear (non-uniform) scaling of the similarity matrix $W = (w_{ij})$, where $w_{ij} = w_{ji} \geq 0$ measures the similarity, association, or correlation between data objects i, j . The scaling factor $D = (d_i)$ is a diagonal matrix with each diagonal element being the sum of the corresponding row ($d_i = \sum_j w_{ij}$). Noting that $W = D^{1/2}(D^{-1/2}WD^{-1/2})D^{1/2}$, we apply PCA or spectral decomposition on the scaled matrix $\widehat{W} = D^{-1/2}WD^{-1/2}$ instead of on W directly, leading to

$$W = D^{1/2}(\sum_k \mathbf{z}_k \lambda_k \mathbf{z}_k^T)D^{1/2} = D \sum_k \mathbf{q}_k \lambda_k \mathbf{q}_k^T D \quad (1)$$

Here we call $\mathbf{q}_k = D^{-1/2}\mathbf{z}_k$ the *scaled principal components* ($\mathbf{q}_k, \mathbf{z}_k$ are n -vectors¹); they are obtained by solving the eigenvalue system

$$D^{-1/2}WD^{-1/2}\mathbf{z} = \lambda\mathbf{z}. \quad (2)$$

or equivalently, solving

$$W\mathbf{q} = \lambda D\mathbf{q}. \quad (3)$$

Self-aggregation

The K -dimensional space spanned by the first K scaled principal components (SPCA space) has an interesting self-aggregation property enforced by within-cluster association (connectivity). This property is first noted in [2].

First, we consider the case where clusters are well separated, i.e., no overlap (no connectivity) exists among the clusters.

Theorem 1. When overlaps among K clusters are zero, the K scaled principal components $(\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_K) = Q_K$ get the same maximum eigenvalue: $\lambda_1 = \dots = \lambda_K = 1$. Each $\mathbf{q}_k$ is a multistep (piecewise-constant) function (assuming objects within a cluster are indexed consecutively). In the SPCA space spanned by Q_K , all objects within the same cluster self-aggregate into a single point. $\square$

¹ Here bold-face lowercase letters are vectors of size n , with $q_k(i)$ as the i th element of $\mathbf{q}_k$. Matrices are denoted by uppercase letters.

Proof. Now $W = (W_{pq})$ is block diagonal: $W_{pq} = 0, p \neq q$. Assume $K = 3$. Define basis vectors:

$$\mathbf{x}^{(k)} = (0 \cdots 0, D_{kk}^{1/2} \mathbf{e}_k, 0 \cdots 0)^T, \quad (4)$$

where $s_{pq} = \sum_{i \in G_p} \sum_{j \in G_q} w_{ij}$, $D_{pq} = \text{diag}(W_{pq} \mathbf{e}_q)$, and $\mathbf{e}_k = (1, \dots, 1)^T$ with the size of cluster G_k . $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \mathbf{x}^{(3)}$ are eigenvectors of Eq.(2) with $\lambda^{(0)} = 1$. For any K real numbers $\mathbf{c} = (c_1, c_2, \dots, c_K)^T$, $\mathbf{z} = X_K \mathbf{c} = c_1 \mathbf{x}^{(1)} + \cdots + c_K \mathbf{x}^{(K)}$ is also an eigenvector of Eq.(2) with $\lambda^{(0)} = 1$. The corresponding scaled principal component

$$\mathbf{q} = D^{-1/2} \mathbf{z} = (c_1 \mathbf{e}_1 / s_{11}^{1/2}, \dots, c_K \mathbf{e}_K / s_{KK}^{1/2})^T, \quad (5)$$

is a K -step piece-wise constant function. Clearly, all data objects within the same cluster have *identical* elements in $\mathbf{q}$. The coordinate of object i in the K -dim SPCA space is $\mathbf{r}_i = (q_1(i), \dots, q_K(i))^T$. Thus objects within a cluster are located at (self-aggregate into) the same point. $\square$

Scaled principal components are not *unique* when no overlap between clusters exist. For a set of K scaled principal components $(\mathbf{q}_1, \dots, \mathbf{q}_K) = Q_K$, and another arbitrary $K \times K$ orthonormal matrix R , $Q_K R$ are also a valid set of scaled principal components. However, the expansion of Eq.(1) is unique, because $\mathbf{q}_k \mathbf{q}_k^T$ is unique. Thus, self-aggregation of cluster member is equivalent to the fact that $Q_K Q_K^T$ has a block diagonal structure,

$$Q_K Q_K^T = \text{diag}(\mathbf{e}_1 \mathbf{e}_1^T / s_{11}, \dots, \mathbf{e}_K \mathbf{e}_K^T / s_{KK}), \quad (6)$$

where elements within the same diagonal block all have the same value. In graph theory, the scaled PCA represents each cluster as a complete graph (clique). For this reason, the truncated SPCA expansion

$$W_K = D \sum_{k=1}^K \mathbf{q}_k \mathbf{q}_k^T D = D Q_K Q_K^T D \quad (7)$$

is particularly useful in discovering cluster structure. Here we retain only first K terms and set $\lambda_k = 1$ which is crucial for enforcing the cluster structure later.

Second, we consider the case when overlaps among different clusters exist. We apply perturbation analysis by writing $\widehat{W} = \widehat{W}^{(0)} + \widehat{W}^{(1)}$, where $\widehat{W}^{(0)}$ is the similarity matrix for the zero-overlap case considered above, and $\widehat{W}^{(1)}$ accounts for the overlap among clusters and is treated as a perturbation.

Theorem 2. At the first order, the K scaled principal components and their eigenvalues have the form

$$\mathbf{q} = D^{-1/2} X_K \mathbf{y}, \quad \lambda = 1 - \zeta,$$

where $\mathbf{y}$ and ζ satisfy the eigensystem $\Gamma \mathbf{y} = \zeta \mathbf{y}$. The matrix Γ has the form $\Gamma = \Omega^{-1/2} \bar{\Gamma} \Omega^{-1/2}$, where

$$\bar{\Gamma} = \begin{pmatrix} h_{11} & -s_{12} & \cdots & -s_{1K} \\ -s_{21} & h_{22} & \cdots & -s_{2K} \\ \vdots & \vdots & \cdots & \vdots \\ -s_{K1} & -s_{K2} & \cdots & h_{KK} \end{pmatrix} \quad (8)$$

$h_{kk} = \sum_{p \neq k} s_{kp}$ (p sums over all indices except k) and $\Omega = \text{diag}(s_{11}, \dots, s_{KK})$. This analysis is accurate to order $\|\widehat{W}^{(1)}\|^2 / \|\widehat{W}^{(0)}\|^2$ for eigenvalues and to order $\|\widehat{W}^{(1)}\| / \|\widehat{W}^{(0)}\|$ for eigenvectors. $\square$

The proof is a bit involved and is omitted here. Several features of SPCA can be obtained from Theorem 2:

Corollary 1. SPCA expansion $W_K = DQ_K Q_K^T D = D^{1/2} X_K X_K^T D^{1/2}$ has the same block diagonal form of Eq.(6) within the accuracy of Theorem 1.

Corollary 2. The first scaled principal component is $\mathbf{q}_1 = D^{-1/2} X_K \mathbf{y}_1 = (1, \dots, 1)^T$ with $\lambda_1 = 1$. λ_1 and $\mathbf{q}_1$ are also the exact solutions to the original Eq.(3).

Corollary 3. The second principal component for $K = 2$ is

$$\mathbf{q}_2 = D^{-1/2} X_2 \mathbf{y}_2 = \left(\sqrt{\frac{s_{22}}{s_{11}}} \mathbf{e}_1, -\sqrt{\frac{s_{11}}{s_{22}}} \mathbf{e}_2 \right)^T. \quad (9)$$

The eigenvalue is

$$\lambda_2 = 1 - (s_{12}/s_{11} + s_{12}/s_{22}). \quad (10)$$

The diagonal block structure of the SPCA expansion W_K (Corollary 1) implies that objects within the same cluster will self-aggregate as in Theorem 1. We can also see this more intuitively. A scaled principal component $\mathbf{q} = (q(1), \dots, q(n))^T$, as an eigenvector of Eq.(3), can be equivalently obtained by minimizing the objective function

$$\min_{\mathbf{q}} \frac{\sum_{ij} w_{ij} [q(i) - q(j)]^2}{\sum_i d_i [q(i)]^2}. \quad (11)$$

Thus adjacent objects have close coordinates such that $[q(i) - q(j)]^2$ is small for non-zero w_{ij} : the larger w_{ij} is, the closer $q(i)$ is to $q(j)$.

To illustrate the above analysis, we provide the following example and applications.

Example 1. A dataset of 3 clusters with substantial random overlap between the clusters. All edge weights are 1. The similarity matrix and results are shown in Fig.1, where nonzero matrix elements are shown as dots. The exact λ_2 and approximate $\tilde{\lambda}_2$ from Theorem 2 are close:

$$\lambda_2 = 0.300, \quad \tilde{\lambda}_2 = 0.268.$$

The SPCA expansion $W_K = DQ_K Q_K^T D$ reveals the correct block structure clearly due to self-aggregation: in W_K connections between different clusters are substantially suppressed while connections within clusters are substantially enhanced. Thus W_K is much sharper than the original weight matrix W . In SPCA space using coordinates $\mathbf{r}_i = (q_1(i), \dots, q_3(i))^T$, objects within the same cluster become almost on top of each other (not shown) as the result of self-aggregation.

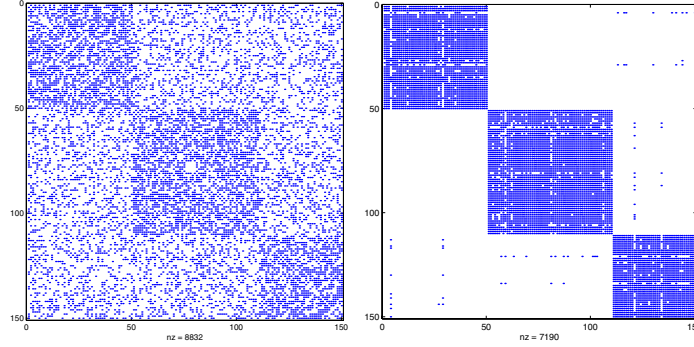


Fig. 1. Left: similarity matrix W . Diagonal blocks represent weights inside clusters and off-diagonal blocks represent overlaps between clusters. Right: Computed W_K

Application 1. In DNA micro-array gene expression profiling, responses of thousands of genes from tumor tissues are simultaneously measured. We SPCA to gene expression profiles of lymphoma cancer data from Alizadeh et al. [1]. Discovered clusters clearly correspond to normal or cancerous subtypes identified by human expertise. 100 most informative genes (defines the Euclidean space) are selected out of the original 4025 genes based on the F -statistic. Pearson correlation c_{ij} is computed and similarity $w_{ij} = \exp(c_{ij}/\langle c \rangle)$, where $\langle c \rangle = 0.1$. Three cancer and three normal subtypes are shown with symbols explained in Figure 2B (the number of samples in each subtype is shown in parentheses). This is a difficult problem due to large variances in cluster sizes. Self-aggregation is evident in Figures 2B and 2C.

Besides the self-aggregation, the nonlinearity in SPCA can alter the topology in a useful way to reveal structures which are otherwise difficult to detect using standard PCA. Thus the SPCA space is a more useful space to explore the structures.

Application 2. 1000 points form two interlocking rings (but not touching each other) in 3D Euclidean space. The similarities between data points are computed same as in Application 1. In SPCA space, rings are separated. Objects self-aggregate into much thinner rings (shown in right panel of Figure 2).

Dynamic Aggregation

The self-aggregation process can be repeated to obtain sharper clusters. W_K is the low-dimensional projection that contains the essential cluster structure. Combining this structure with the original similarity matrix, we obtain a new similarity matrix containing sharpened cluster information:

$$W^{(t+1)} = (1 - \alpha)W_K^{(t)} + \alpha W^{(t)}, \quad (12)$$

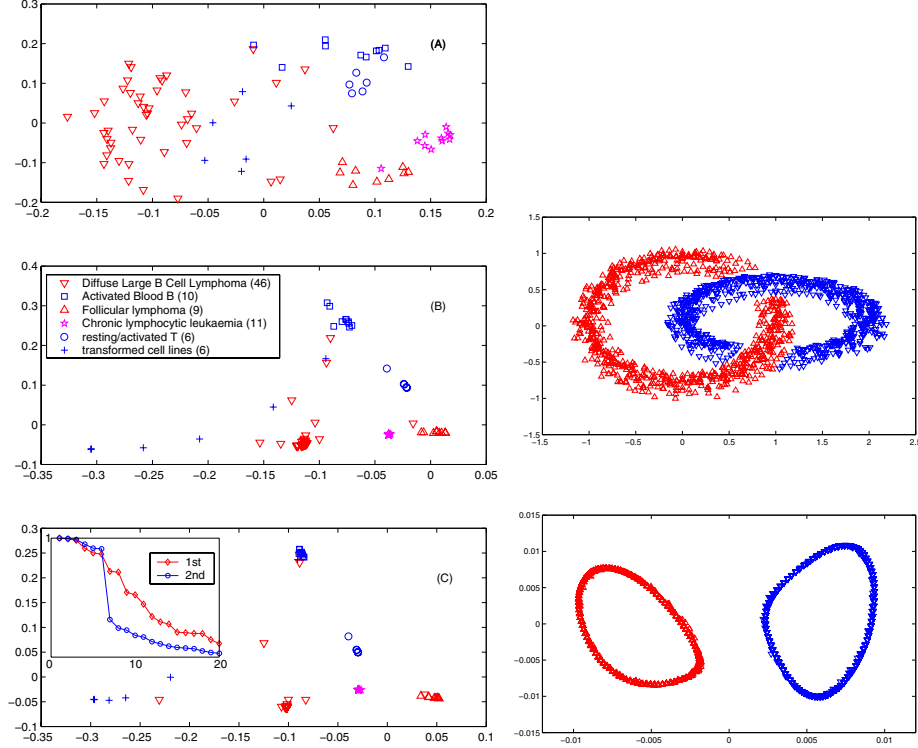


Fig. 2. Left: Gene expression profiles in original Euclidean space (A), in SPCA space (B), and in SPCA space after one iteration of Eq. 13 (C). In all 3 panels, objects are plotted in 2D-view spanned by the first two PCA components. Cluster structures become clearer due to self-aggregation. The insert in (C) shows the eigenvalues of the 1st and 2nd SPCA.

Right: Data objects in 3D Euclidean space (top) and in SPCA space (bottom)

where $W_K^{(t)}$ is the SPCA representation (Eq. 7) of $W^{(t)}$, the weight matrix at t -th iteration, $\alpha = 0.5$, and $W^{(1)} = W$.

Applying SPCA on $W^{(2)}$ leads to further aggregation (see Figure 2C). The eigenvalues of the 1st and 2nd SPCA are shown in the insert in Figure 1C. As iteration proceeds, a clear gap is developed, indicating that clusters becoming more separated.

Noise Reduction

SPCA representation W_K has noises. For example, W_K has sometimes negative weights $(W_K)_{ij}$ whereas we expect them to be nonnegative for learning. However, a nice property of SPCA provides a solution. The structure of W_K is determined by QQ^T . When data contains K well separated clusters, QQ^T has a diagonal block structure and every elements in the block are identical (Eq. 6).

When clusters are not well separated but can be meaningfully distinguished, QQ^T has approximately the same block-diagonal form (Corollary 1). This property allows us to interpret QQ^T as the probability that two objects i, j belong to the same cluster:

$$p_{ij} = (W_K)_{ij} / (W_K)_{ii}^{1/2} (W_K)_{jj}^{1/2}.$$

which is the same as $p_{ij} = (QQ^T)_{ij} / (QQ^T)_{ii}^{1/2} (QQ^T)_{jj}^{1/2}$. To reduce noise in the above dynamic aggregation, we set

$$(W_K)_{ij} = 0 \quad \text{if} \quad p_{ij} < \beta, \quad (13)$$

where $0 < \beta < 1$ and we chose $\beta = 0.8$. Noise reduction is an integral part of SPCA. In general, the method is stable: the final results are insensitive to α, β . The above dynamic aggregation process repeatedly projects data into SPCA space and the self-aggregation forces data objects towards the attractors of the dynamics. The attractors are the desired clusters which are well separated and their principal eigenvalues approach 1 (see insert in Fig.1C). Usually, after one or two iterations of self-aggregation in SPCA, the cluster structure becomes evident.

3 Mutual Dependence

In many learning and information processing tasks, we look for inter-dependence among different aspects (attributes) of the same data objects. In gene expression profiles, certain genes express strongly when they are from tissues of a certain phenotype, but express mildly when they are from other phenotypes[1]. Thus it is meaningful to consider gene-gene correlations as characterized by their expressions across all tissue samples, in addition to sample-sample correlations we usually study.

In text processing, such as news articles, the content of an article is determined by the word occurrences, while the meaning of words can be inferred through their occurrences across different news articles. This kind of association between a data object (tissues, news articles) and its attributes (expressions of different genes, word occurrences) is represented by the asymmetric data association matrix. Here we restrict our consideration to the cases where all entries of association matrix B are non-negative, and therefore can be viewed as the probability of association (conditional probability) between column objects (news articles or tissue samples) and row objects (words or genes). This kind of data is sometimes called a contingency table. In graph theory, B is the weight matrix for a bipartite graph. Clustering row and column objects simultaneously amounts to clustering the bipartite graph as shown in Figure 3.

SPCA applies to these inter-dependence problems (bipartite graphs) as well. We introduce nonlinear scaling factors, diagonal matrices D_r (each element is the sum of a row) and D_c (each element is the sum of a column). Let $B =$

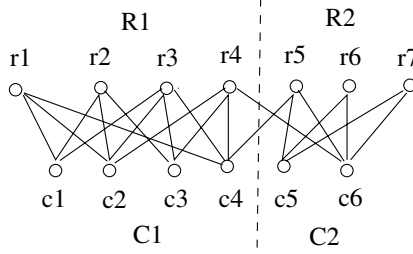


Fig. 3. A bipartite graph with row-nodes and column-nodes. The dashed line indicates a possible clustering

$D_r^{1/2}(D_r^{-1/2}BD_c^{-1/2})D_c^{1/2}$. Applying PCA on $\hat{B} = D_r^{-1/2}BD_c^{-1/2}$, we obtain

$$B = D_r^{1/2}\left(\sum_k \mathbf{u}_k \lambda_k \mathbf{v}_k^T\right)D_c^{1/2} = D_r \sum_k \mathbf{f}_k \lambda_k \mathbf{g}_k^T D_c. \quad (14)$$

Scaled principal components are $\mathbf{f}_k = D_r^{-1/2}\mathbf{u}_k$ for row objects and $\mathbf{g}_k = D_c^{-1/2}\mathbf{v}_k$ for column objects.

Scaled principal components here have the same self-aggregation and related properties as in §2. First, the singular vectors $\mathbf{u}_k$ and $\mathbf{v}_k$ and the singular values λ_k are determined though

$$(\hat{B}\hat{B}^T)\mathbf{u} = \lambda^2\mathbf{u}, \quad (\hat{B}^T\hat{B})\mathbf{v} = \lambda^2\mathbf{v}. \quad (15)$$

They can be viewed as simultaneous solutions to Eq.(3), with

$$W = \begin{pmatrix} & B \\ B^T & \end{pmatrix}, \quad D = \begin{pmatrix} D_r & \\ & D_c \end{pmatrix}, \quad \mathbf{z} = \begin{pmatrix} \mathbf{u} \\ \mathbf{v} \end{pmatrix},$$

as can be easily verified. Therefore, all conclusions of Theorems 1 and 2 for undirect graphs can readily extended over to the bipartite case here.

When K clusters are well separated (no overlap among clusters), we have

Theorem 3. For well separated clusters, row objects within the same cluster will self-aggregate in the SPCA space spanned by $(\mathbf{f}_1, \dots, \mathbf{f}_K) = F_K$, while column objects within the same cluster will self-aggregate in the SPCA space spanned by $(\mathbf{g}_1, \dots, \mathbf{g}_K) = G_K$. $\square$

When clusters overlap, a theorem almost identical to Theorem 2 can be established for bipartite graphs. The corollaries following Theorem 2 can be nearly identically extended to the bipartite graphs. We briefly summarize the results here. Let

$$\mathbf{q}_k = \begin{pmatrix} \mathbf{f}_k \\ \mathbf{g}_k \end{pmatrix} = D^{-\frac{1}{2}} \begin{pmatrix} \mathbf{u}_k \\ \mathbf{v}_k \end{pmatrix}, \quad Q_K = \begin{pmatrix} F_K \\ G_K \end{pmatrix}, \quad \text{and} \quad Q_K Q_K^T = \begin{pmatrix} F_K F_K^T & F_K G_K^T \\ G_K F_K^T & G_K G_K^T \end{pmatrix}.$$

The low-dimensional SPCA expansion $B_K = D_r \sum_{k=1}^K \mathbf{f}_k \mathbf{g}_k^T D_c = D_r F_K G_K^T D_c$ gives the sharpened association between words and documents, the diagonal

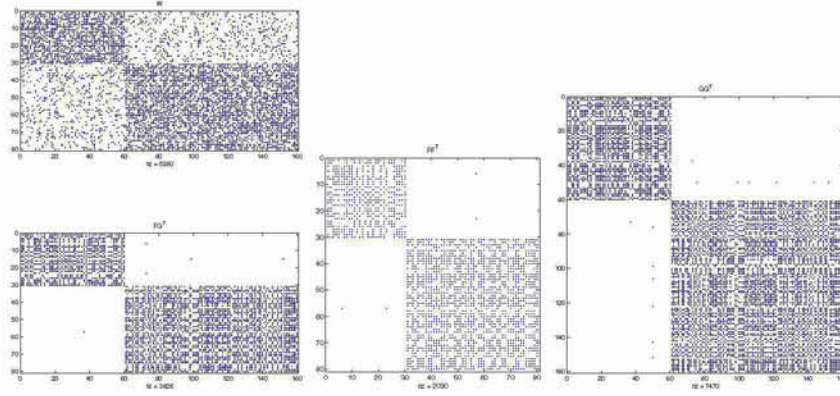


Fig. 4. Left-top: association (weight) matrix of a bipartite graph of 2 dense clusters (diagonal blocks) with random overlaps (off-diagonal blocks). Left-bottom: $F_K G_K^T$ for sharpened associations. Middle: $G_K G_K^T$ for clustering column objects. Right: $F_K F_K^T$ for clustering row objects

block structure of $F_K F_K^T$ gives the clusters on row objects (words) while the diagonal block structure of $G_K G_K^T$ simultaneously gives the clusters on column objects (news articles). We note that Eqs.(14,15) rediscover the correspondence analysis [7] in multivariate statistics from the SPCA point of view.

Example 2. We apply the above analysis to a bipartite graph example with association matrix shown in Fig.4. The bipartite graph has two dense clusters with large overlap between them. The SPCA representations are computed and shown in Fig.4. $F_K G_K^T$ gives a sharpened association matrix where the overlap between clusters (off-diagonal blocks) is greatly reduced. $F_K F_K^T$ reveals the cluster structure for row objects and $G_K G_K^T$ reveals the cluster structure for column objects.

Application 4. Clustering internet newsgroups (see Figure 5). (The newsgroup dataset is from www.cs.cmu.edu/afs/cs/project/theo-11/www/naive-bayes.html) Five newsgroups are used in the dataset (listed in upper right corner with corresponding color). 100 news articles are randomly chosen from each newsgroup. From them 1000 words selected. Standard **tf.idf** weighting are used. Each document (column) is normalized to one. The resulting word-to-document association matrix is the input matrix B . As shown in Figure 5, words aggregate in SPCA word space (spanned by F_K) while news articles are simultaneously clustered in SPCA document space (spanned by G_K) shown by the projection matrix GG^T (the insert). One can see that GG^T indicates some overlap between *computer graphics* and *space science*, which is consistent with the relative closeness of the two corresponding word clusters in word space. The accuracy of clustering is 86%. (We also computed the cosine similarity $W = B^T B$ and use the method in §2 to obtain clusters with 89% accuracy.) This dataset

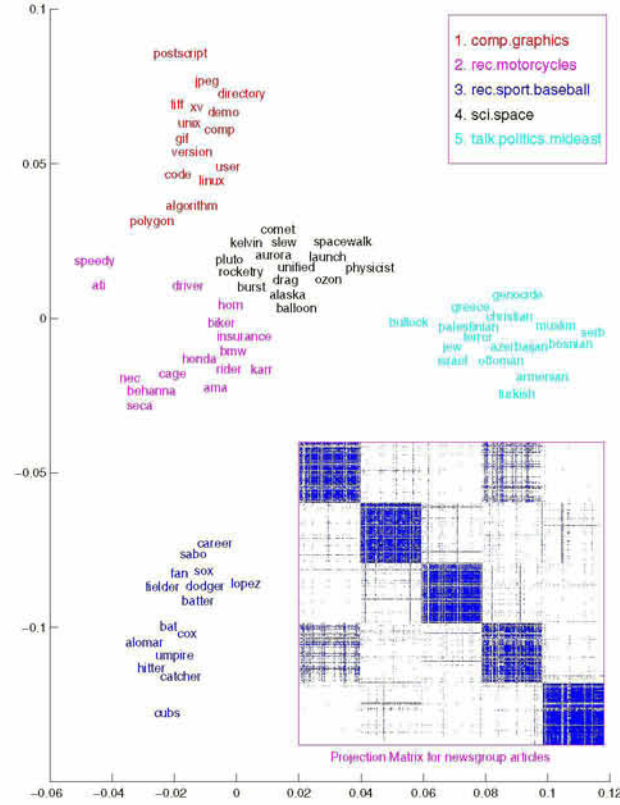


Fig. 5. Words self-aggregate in SPCA word space while internet newsgroups articles are simultaneously clustered. Shown are the top 15 most frequently occurring words from each discovered cluster. (Several words in *motorcycles* are brand names, and several words in *baseball* are players' names.) The insert shows the projection matrix GG^T on clustering news articles

has been extensively studied in [24]; the standard Kmeans clustering gives about 66% accuracy, while two improved methods get 76-80% accuracy.

4 Discussions

In this paper, we assume that objects belonging to the same cluster are consecutively indexed. However, the SPCA framework is independent of the indexing. The diagonal block structure of SPCA representation as the result of cluster member self-aggregation merely indicates the fact that connectivities between different clusters are substantially suppressed while connectivities within a cluster are substantially enhanced. Our main results, in essence, is that if cluster structures in the original dataset can be meaningfully distinguished, such as

Figures 1,2, SPCA makes them much more well-separated so that clusters can be easily detected either by direct visual inspection or by a standard clustering method such as the K-means algorithm.

The key to understand SPCA is the nonlinear scaling factor D . Columns and rows of the similarity matrix are scaled inversely proportional to their weights such that all K principal components get the same maximum eigenvalue of one. This happens independent of cluster sizes, leading to desirable consequences. (i) Outliers are usually far away from other objects and often skew the statistics (means, covariance, etc) in original Euclidean space. However, in SPCA we focus on similarity matrix (instead of distance matrix). Outliers contribute very little to the quantities in Eqs.(4,8) thus do not adversely affects SPCA. But, their small similarities with other objects force them to appear as independent clusters and thus can be easily detected. (ii) Unbalanced clusters (in which the number of objects in each cluster varies substantially) are usually difficult to discover using many other clustering methods, but can be effectively dealt with in SPCA due to the nonlinear scaling. Directly applying PCA on W will be dominated by the large clusters and no self-aggregation will occur.

The scaled PCA has a connection to spectral graph partitioning and clustering [4,6,19,8,22,3,18]. Given a weighted graph G where the weight w_{ij} is the similarity between nodes i, j , one wish to partition it into two subgraphs (clusters) A, B according to the *min-max clustering principle*: the (overlapping) similarity between A and B is minimized while similarities within A or B are maximized[3]. The overlap between A and B is the sum of weights between A and B , $s(A, B) = \sum_{i \in A, j \in B} w_{ij}$. The similarity within cluster A is $s(A, A)$ (sum of all edge weights within A). The similarity within cluster B is $s(B, B)$. Thus the clustering principle of minimizing $s(A, B)$ while maximizing $s(A, A)$ and $s(B, B)$ leads to the min-max cut objective function[3],

$$J_{\text{MMC}} = \frac{s(A, B)}{s(A, A)} + \frac{s(A, B)}{s(B, B)}. \quad (16)$$

The clustering result can be represented by an indicator vector $\mathbf{q}$, where $q(i) = a$ or $-b$ depending on node $i \in A$ or B . (a and b are positive constants.) If one relaxes $q(i)$ from discrete indicators to continuous values in $(-1, 1)$, the solution $\mathbf{q}$ for minimizing J_{MMC} is given by the eigenvector of $(D - W)\mathbf{q} = \zeta D\mathbf{q}$, which is exactly Eq.3 with $\lambda = 1 - \zeta$. This further justifies our SPCA approach for unsupervised learning. In addition, the desired clustering indicator vector $\mathbf{q}$ is precisely recovered in Eq.9 with $a = \sqrt{s_{22}/s_{11}}$ and $b = \sqrt{s_{11}/s_{22}}$ due to Theorem 1; minimizing the min-max cut objective of Eq.16 is equivalent to maximizing the eigenvalue of the second SPCA component given in Eq.10. All these indicate that SPCA is a principled and coherent framework for data clustering. One drawback of the method is the computation is in general $O(n^2)$.

In self-aggregation, data objects move towards each other guided by connectivity which determines the attractors. This is similar to the self-organizing map [15,11], where feature vectors self-organize into a 2D feature map while data objects remain fixed. All these have a connection to recurrent networks [12,10].

In Hopfield network, features are stored as associative memories. In more complicated networks, connection weights are dynamically adjusted to learn or discover the patterns, much like the dynamic aggregation of Eq.(12). Thus it may be possible to construct a recurrent network that implements the self-aggregation. In this network, high dimensional input data are converted into low-dimensional representations in SPCA space and cluster structures emerge as the attractors of dynamics.

References

1. A. A. Alizadeh, M. B. Eisen, et al. Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling. *Nature*, 403:503–511, 2000. 116, 118
2. C. Ding, X. He, and H. Zha. A spectral method to separate disconnected and nearly-disconnected web graph components. In *Proc. ACM Int'l Conf Knowledge Disc. Data Mining (KDD 2001)*, pages 275–280. 113
3. C. Ding, X. He, H. Zha, M. Gu, and H. Simon. A min-max cut algorithm for graph partitioning and data clustering. *Proc. 1st IEEE Int'l Conf. Data Mining*, pages 107–114, 2001. 122
4. W. E. Donath and A. J. Hoffman. Lower bounds for partitioning of graphs. *IBM J. Res. Develop.*, 17:420–425, 1973. 122
5. R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification, 2nd ed.* Wiley, 2000. 112
6. M. Fiedler. Algebraic connectivity of graphs. *Czech. Math. J.*, 23:298–305, 1973. 122
7. M. J. Greenacre. *Theory and Applications of Correspondence Analysis*. Academic press, 1984. 120
8. L. Hagen and A. B. Kahng. New spectral methods for ratio cut partitioning and clustering. *IEEE. Trans. on Computed Aided Desgin*, 11:1074–1085, 1992. 122
9. T. Hastie and W. Stuetzle. Principal curves. *J. Amer. Stat. Assoc.*, 84:502–516, 1989. 112
10. S. S. Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 1998, 2nd ed. 113, 122
11. J. Himberg. A som based cluster visualization and its application for false coloring. *Proc Int'l Joint Conf. Neural Networks*, pages 587–592, 2000. 122
12. J. J. Hopfield. Neural networks and physical systems with emergent collective computation abilities. *Proc. Nat'l Acad Sci USA*, 79:2554–2558, 1982. 113, 122
13. A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31:264–323, 1999. 112
14. I. T. Jolliffe. *Principal Component Analysis*. Springer Verlag, 1986. 112
15. T. Kohonen. *Self-organization and Associative Memory*. Springer-Verlag, 1989. 122
16. M. A. Kramer. Nonlinear principal component analysis using autoassociative neural networks. *AIChE Journal*, 37:233–243, 1991. 112
17. D. D. Lee and H. S. Seung. Learning the parts of objects with nonnegative matrix factorization. *Nature*, 401:788–791, 1999. 112
18. A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. *Proc. Neural Info. Processing Systems (NIPS 2001)*, Dec. 2001. 122
19. A. Pothen, H. D. Simon, and K. P. Liou. Partitioning sparse matrices with eigenvectors of graph. *SIAM Journal of Matrix Anal. Appl.*, 11:430–452, 1990. 122

20. S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290:2323–2326, 2000. [112](#)
21. B. Scholkopf, A. Smola, and K. Muller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319, 1998. [112](#)
22. J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE. Trans. on Pattern Analysis and Machine Intelligence*, 22:888–905, 2000. [122](#)
23. J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2323, 2000. [112](#)
24. H. Zha, C. Ding, M. Gu, X. He, and H. D. Simon. Spectral relaxation for k-means clustering. *Proc. Neural Info. Processing Systems (NIPS 2001)*, Dec. 2001. [121](#)

A Classification Approach for Prediction of Target Events in Temporal Sequences

Carlotta Domeniconi¹, Chang-shing Perng², Ricardo Vilalta², and Sheng Ma²

¹ Department of Computer Science, University of California
Riverside, CA 92521 USA
{carlotta@cs.ucr.edu}

² IBM T.J. Watson Research Center, 19 Skyline Drive
Hawthorne, N.Y. 10532 USA
{perng,vilalta,shengma}@us.ibm.com

Abstract. Learning to predict significant events from sequences of data with categorical features is an important problem in many application areas. We focus on events for system management, and formulate the problem of prediction as a classification problem. We perform co-occurrence analysis of events by means of Singular Value Decomposition (SVD) of the examples constructed from the data. This process is combined with Support Vector Machine (SVM) classification, to obtain efficient and accurate predictions. We conduct an analysis of statistical properties of event data, which explains why SVM classification is suitable for such data, and perform an empirical study using real data.

1 Introduction

Many real-life scenarios involve massive sequences of data described in terms of categorical and numerical features. Learning to predict significant events from such sequences of data is an important problem useful in many application areas.

For the purpose of this study, we will focus on system management events. In a production-network, the ability of predicting specific harmful events can be applied for automatic real-time problem detection. In our scenario, a computer network is under continuous monitoring. Our data reveals that one month of monitoring of a computer network with 750 hosts can generate over 26,000 events, with 164 different types of events. Such high volume of data makes necessary the design of efficient and effective algorithms for pattern analysis.

We take a classification approach to address the problem of event data prediction. The historical sequence of data provides examples that serve as input to the learning process. Our settings allow to capture temporal information through the use of adaptive-length monitor windows. In this scenario, the main challenge consists in constructing examples that capture information that is relevant for the associated learning system. Our approach to address this issue has its foundations in the information retrieval domain.

Latent Semantic Indexing (LSI) [5] is a method for selecting informative subspaces of feature spaces. It was developed for information retrieval to reveal

semantic information from co-occurrences of terms in documents. In this paper we demonstrate how this method can be used for pattern discovery through feature selection for making predictions with temporal sequences. The idea is to start with an initial rich set of features, and cluster them based on feature correlation. The finding of correlated features is carried out from the given set of data by means of SVD. We combine this process with SVM, to obtain efficient and accurate predictions. The resulting classifier, in fact, is expressed in terms of a reduced number of examples, which lie in the feature space constructed through feature selection. Thereby, predictions can be performed efficiently. Besides performing comparative studies, we also take a more theoretical perspective to motivate why SVM learning method is suitable for our problem. Following studies conducted for text data [8], we discover that the success of SVM in predicting events has its foundations on statistical properties of event data.

2 Problem Settings

We assume that a computer network is under continuous monitoring. Such monitoring process produces a sequence of events, where each event is a timestamped observation described by a fixed set of categorical and numerical features. Specifically, an event is characterized by four components: the time at which the event occurred, the type of the event, the host that generated the event, and the severity level. The severity component can assume five different levels: $\{harmless, warning, minor, critical, fatal\}$.

We are interested in predicting events with severity either critical or fatal, which we call *target events*. Such events are rare, and therefore their occurrence is sparse in the sequence generated by the monitoring process. Our goal is then to identify situations that lead to the occurrence of target events. Given the current position in time, by observing the monitoring history within a certain time interval (monitor window), we want to predict if a given target event will occur after a warning interval.

In our formulation of the problem, as we shall see, events will be characterized by their timestamp and type components. In this study, the host component is ignored (some hosts generate too few data). Therefore, we will denote events as two dimensional vectors $\mathbf{e} = (\text{timestamp}, \text{type})$. We will use capital letter $\mathbf{T}$ to denote each target event, which is again a two dimensional vector. We assume that the severity level of target events is either critical or fatal.

3 Related Work

Classical time series analysis is a well studied field that involves identifying patterns (trend analysis), identifying seasonal changes, and forecasting [2]. There exist fundamental differences between time series and event data prediction that render techniques for time series analysis inappropriate in our case. A time series is, in fact, a sequence of real numbers representing measurements of a variable

taken at equal time intervals. Techniques developed for time series require numerical features, and do not support predicting specific target events within a time frame. The nature of event data is fundamentally different. Events are characterized by categorical features. Moreover, events are recorded as they occur, and show different inter-arrival times. Correlations among events are certainly local in time, and not necessarily periodic. New models that capture the temporal nature of event data are needed to properly address the prediction problem in this context.

The problem of mining sequences of events with categorical features has been studied by several researchers [10,16]. Here the focus is on finding frequent subsequences. [16] systematically searches the sequence lattice spanned by the subsequence relation, from the most general to the most specific frequent sequences. The minimum support is a user defined parameter. Temporal information can be considered through the handling of a time window.

[10] focuses on finding all frequent episodes (from a given class of episodes), i.e., collections of events occurring frequently close to each other. Episodes are viewed as partially ordered sets of events. The width of the time window within which the episode must occur is user defined. The user also specifies the number of times an event must occur to qualify as frequent. Once episodes are detected, rules for prediction can be obtained. Clearly, the identified rules depend on the initial class of episodes initially considered, and on the user defined parameters.

Our view of target events and monitor periods is akin to the approach taken in [14], and in [15]. [14] adopts a classification approach to generate a set of rules to capture patterns correlated to classes. The authors conduct a search over the space of monomials defined over boolean vectors. To make the search systematic, pruning mechanisms are employed, which necessarily involve parameter and threshold tuning.

Similarly, [15] sets the objective of constructing predictive rules. Here, the search for prediction patterns is conducted by means of a genetic algorithm (GA), followed by a greedy procedure to screen the set of pattern candidates returned by the GA.

In contrast, in this work, we fully exploit the classification model to solve the prediction problem. We embed patterns in examples through co-occurrence analysis of events in our data; by doing so we avoid having to search in a space of possible solutions, and to conduct post-processing screening procedures. We are able to conduct the classification approach in a principled manner that has its foundations on statistical properties of event data.

4 Prediction as Classification

The overall idea of our approach is as follows. We start with an initial rich set of features which encodes information relative to each single event type; we then cluster correlated components to derive information at pattern level. The resulting feature vectors are used to train an SVM. The choice of conducting SVM classification is not merely supported by our empirical study, but finds its

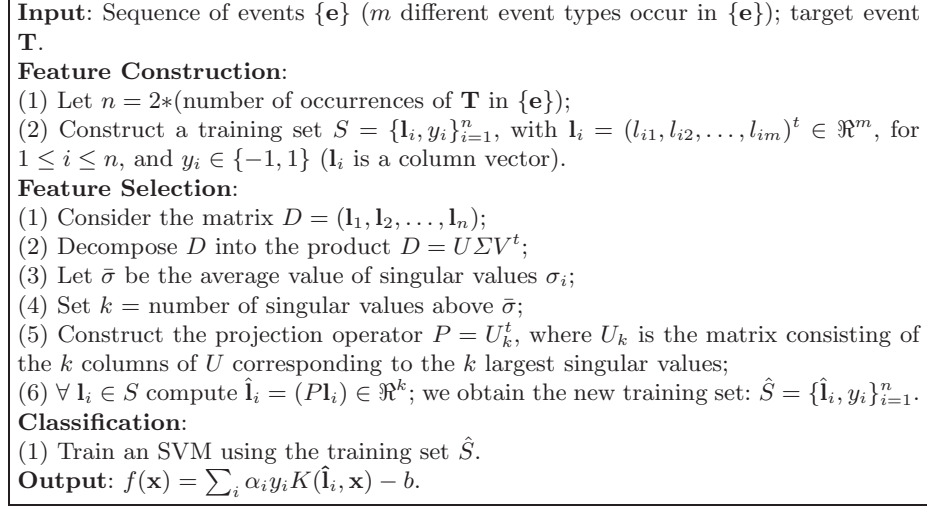


Fig. 1. Summary of the SVD-SVM algorithm

foundation on the structure and on the distribution properties of our data, as we will discuss in section 6. In figure 1 we summarize the whole algorithm, which we call SVD-SVM.

4.1 Feature Construction

We generate a training set of positive and negative examples for each target event $\mathbf{T}$. Specifically, we generate a positive example for each occurrence of $\mathbf{T}$. We do so by observing the monitoring history within a certain time interval, which we call *monitor window*, preceding the occurrence of $\mathbf{T}$, and preceding a *warning window*. The warning window represents the leading time useful to take actions for preventing the target event from happening during the on-line prediction process.

We proceed similarly for the construction of negative examples, monitoring the history of events within windows which are far from occurrences of the target event along the time axis. The rationale behind this choice is to try to minimize the overlapping of features between positive and negative examples. This strategy is a heuristic, and other approaches may be valid as well.

Our first step toward feature selection involves the mapping of temporal sequences (i.e., monitor windows) into vectors $\mathbf{l} \in \mathbb{R}^m$, whose dimensionality m is given by the number of event types in the data. Each component l_i of $\mathbf{l}$ encodes the information relative to event e_i with respect to the monitor window under consideration. In general, the value for l_i could be a function of the timestamps of e_i . Alternatively, l_i could simply encode the number of occurrences, or just the existence of the corresponding event type e_i , within the monitor window.

4.2 Feature Selection

Let us assume we have m different event types, and we take into consideration n monitor windows to generate both positive and negative examples. Then, the feature construction step gives a training set of n m -dimensional vectors: $\mathbf{l}_1, \mathbf{l}_2, \dots, \mathbf{l}_n$, with $\mathbf{l}_i \in \mathbb{R}^m$ for $1 \leq i \leq n$. We denote each $\mathbf{l}_i$ as a column vector: $\mathbf{l}_i = (l_{i1}, l_{i2}, \dots, l_{im})^t$. We can then represent the vectors in the training set as a matrix: $D = (\mathbf{l}_1 \mathbf{l}_2 \dots \mathbf{l}_n)$, whose rows are indexed by the event types, and whose columns are indexed by the training vectors. We call D the *event-by-window* matrix. Its dimensions are $m \times n$.

We extract relevant information from the given training set by performing the SVD of the event-by-window matrix. The vectors are projected into the subspace spanned by the first k singular vectors of the feature space. Hence, the dimension of the feature space is reduced to k , and we can control this dimension by varying k . This process allows us to obtain a vector space in which distances reflect pattern-context information.

Using SVD, we decompose the event-by-window matrix D into the product $D = U\Sigma V^t$, where U and V are square orthogonal matrices, and Σ has the same dimensions as D , but is only non-zero on the leading diagonal. The diagonal contains the (positive) singular values in decreasing order, $\sigma_1 \geq \sigma_2 \dots \geq \sigma_k \geq \dots \geq 0$ (we denote with $\bar{\sigma}$ their average value). The first k columns of U span a subspace of the space of event vectors which maximizes the variation of the examples in the training set. By using this subspace as a feature space, we force the co-occurring event types to share similar directions.

The number of features is reduced; the level of grouping controls the performance, and is determined by the choice of k . In our experiments we exploit the pattern similarity information that characterizes the data by setting k equal to the number of singular values which are above the average $\bar{\sigma}$, in correspondence of the monitor window length that minimizes the error rate (see section 7).

The projection operator onto the first k dimensions is given by $P = U_k^t$, where U_k is the matrix consisting of the first k columns of U . We can then project the vectors $\mathbf{l}_i$ into the selected k dimensions by computing $\hat{\mathbf{l}}_i = (P\mathbf{l}_i) \in \mathbb{R}^k$. This gives us the new k -dimensional vectors $\hat{\mathbf{l}}_i$, for $1 \leq i \leq n$. Assuming we are interested in predicting c target events, the feature selection process produces c training sets: $\{\hat{\mathbf{l}}_i, y_i\}_{i=1}^n$. We use each of these sets to train an SVM, obtaining c classifiers $SVM_1, SVM_2, \dots, SVM_c$.

The meaning of feature selection is twofold. Correlated dimensions are explicitly embedded in the induced space; they represent relevant features for the successive classification step. Thereby, higher prediction accuracy can be achieved. Furthermore, since the dimensionality of the induced feature space is reduced, this phase makes classification (and prediction) more efficient. Of course, the SVD process itself has a cost, but it is performed only once and off-line. For on-line updates, incremental techniques have been developed for computing SVD in linear time in the number of vectors [4,6]. [9] reduces this linear dependence by using a much smaller aggregate data set to update SVD. Furthermore, new optimization approaches that specifically exploit the struc-

ture of the SVM have been introduced for scaling up the learning process [3,12]. Techniques have been developed to also extend the SVM learning algorithm in an incremental fashion [11,13].

5 SVMs for Text Classification

The rationale for using SVMs relies on the fact that event and text data share important statistical properties, that can be tied to the performance of SVMs. Here we discuss such properties for text data, and then show that similar ones hold for event data also.

SVMs have been successfully applied for text classification. In [8], a theoretical learning model that connects SVMs with the statistical properties of text classification tasks has been presented. This model explains why and when SVMs perform well for text classification. The result is an upper bound connecting the expected generalization error of an SVM with the statistical properties of a particular text classification task.

The most common representation for text classification is the bag-of-words model, in which each word of the language corresponds to an attribute. The attribute value is the number of occurrences of that word in a document. It has been shown [8] that such text classification tasks are characterized by the following properties: *High Dimensional Feature Space*. Many text classification tasks have more than 30,000 features. *Heterogenous Use of Words*. There is generally not a small set of words or a single word that sufficiently describes all documents with respect to the classification task. *High Level of Redundancy*. Most documents contain not only one, but multiple features indicating its class. *Sparse Document Vectors*. From the large feature space, only a few words occur in a single document. *Frequency Distribution of Words follows Zipf's Law*. This implies that there is a small number of words that occurs very frequently while most words occur very infrequently [17]. It is possible to connect these properties of text classification tasks with the generalization performance of an SVM [8]. In particular, the listed properties necessarily lead to large margin separation. Moreover, large margin, combined with low training error, is a sufficient condition for good generalization accuracy.

6 Statistical Properties of Event Data

Here we pose the following question: do properties similar to those discussed in section 5 hold for event data also? In order to answer it, we analyzed the frequency distributions of event types for positive and negative examples of training sets relative to different target events. For lack of space, we report the results obtained for only one target event: CRT_URL_Timeout (coded as target event 2), which indicates that a web site is unaccessible. We have obtained similar results for the other target events. The data used are real event data from a production computer network. The data shows 164 different event types, numbered from 1 to 164. Therefore, each example is a 164-dimensional feature vector, with one

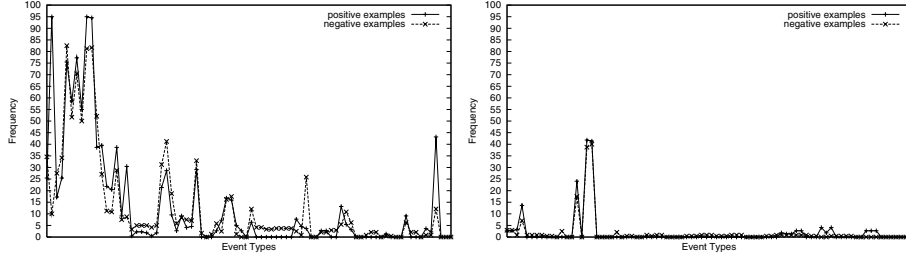


Fig. 2. Frequency of Event Types 1-82 (*left*) and 83-164 (*right*) in positive and negative examples for prediction of Target Event 2

component per event type. Each component encodes the existence of the corresponding event type. The training set for event type 2 contains 460 examples, with roughly the same number of positive and negative instances.

In figure 2, we plot the frequency value of each event type for both positive and negative examples in the training set relative to target event 2. We observe that many event types have very low or zero frequency value, indicating that the 164-dimensional examples are sparse feature vectors. Furthermore, we observe a significant gap in frequency levels between positive and negative examples in correspondence of multiple event types. This shows positive evidence for redundancy of features indicating the class of examples. Also, given the frequency levels of some relevant features, it is highly likely that a significant number of positive examples does not share any of these event types, showing a heterogeneous presence of event types. In order to test the Zipf’s law distribution property, in figure 3 we show the rank-frequency plot for the positive examples relative to target event 2. A similar plot was obtained for the negative examples. The plot shows the occurrence frequency versus the rank, in logarithmic scales. We observe a Zipf-like skewed behavior, very similar to the one obtained for rank-frequency plots of words [1].

We observe that the Zipf distribution does not perfectly fit the plot in figure 3. In fact, in log-log scales, the Zipf distribution gives a straight line, whereas our plot shows a top concavity. It is interesting to point out that the same “parabola” phenomenon has been observed with text data also [1]. In [8], the assumption that term frequencies obey Zipf’s law is used to show that the Euclidean length of document vectors is small for text-classification tasks. This result in turns contributes to bound the expected generalization error tightly. The characteristic that feature vectors are short still holds under the Zipf-like skewed behavior observed for our data. These results provide experimental evidence that statistical properties that hold for text data are valid for event data also. As a consequence, similar theoretical results [8] derived for text data also apply for event data. This establishes the foundations for conducting SVM classification.

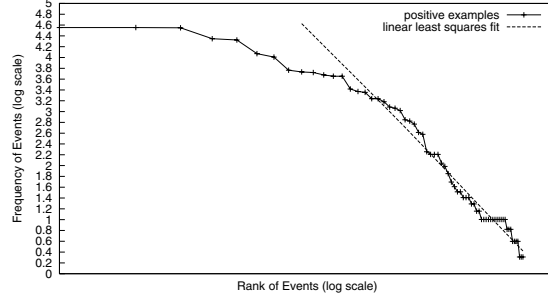


Fig. 3. Rank-frequency plot in logarithmic-logarithmic scales, positive examples for prediction of Target Event 2

7 Experiments on Real Data

In the following we compare different classification methods using real data. We compare the following approaches: (1) **SVD-SVM** classifier. We used *SVM^{light}* [7] with radial basis kernels to build the SVM classifier. We optimized the value of γ in $K(\mathbf{x}_i, \mathbf{x}) = e^{-\gamma \|\mathbf{x}_i - \mathbf{x}\|^2}$, as well as the value of C for the soft-margin classifier, via cross-validation over the training data. (2) **SVM** classifier in original feature space. Again we used *SVM^{light}* with radial basis kernels, and optimized the values of γ and C via cross-validation over the training data. (3) **C4.5** decision tree method in original and reduced feature spaces.

We used real event data from a production computer network. The monitoring process samples events at equal time intervals of one minute length. The characteristics of the data are as follows: the time span covered is of 30 days; the total number of events is 26,554; the number of events identified as critical is 692; the number of events identified as fatal is 16; the number of hosts is 750; the number of event types is 164.

We focus on the prediction of two critical event types: `CRT_URL.Timeout` (coded as type 2), which indicates that a web site is unaccessible, and `OV_Node.Down` (coded as type 94), which indicates that a managed node is down. This choice has been simply determined by the fact that the data contain a reasonable number of occurrences for these two critical event types. In particular, we have 220 occurrences of event type 2, and 352 occurrences of event type 94. We have generated, roughly, an equal number of positive and negative examples for each of the two event types. Specifically, we have constructed 460 examples (220 positive and 240 negative) for event type 2, and 702 examples (352 positive and 350 negative) for event type 94. We have performed 10 2-fold cross-validation to compute error rates.

Feature Construction Processes. We have first conducted an experiment to compare different feature construction processes: (1) **existence**: encodes the

Table 1. Prediction of Event Type 2 using SVD-SVM. Performance results for three different feature construction processes

	ERROR(%)	STD DEV	SEL.DIM.
EXISTENCE	10.6	0.3	61
COUNT	14.7	0.4	40
TEMPORAL	15.1	0.4	66

existence of each event type; (2) **count**: encodes the number of occurrences of each event type; (3) **temporal**: encodes times of occurrences of each event type.

To encode times of occurrences, we partition the monitor window into time slots. Then, for each event type, we generate a binary string with one digit for each time slot: the digit value is one if the event type occurs within the corresponding time slot; otherwise it is zero. We then translate the resulting binary sequence into a decimal number, that uniquely encodes the timestamps (time slots) of the correspondent event type. The collection of the resulting m numbers gives the feature vector.

The results shown in table 1 have been obtained applying the SVD-SVM technique for prediction of event type 2, using a 30 minutes length monitor window and a 5 minutes length warning window. The three columns show: error rate, standard deviation, and number of selected dimensions. The best performance has been obtained when existence is used as feature construction process. The same trend has been observed for target event 94 also. The fact that ignoring the temporal distribution of events within the specified monitor window gives better results, while surprising at first, may be due to patterns that repeat under different event type permutations. Furthermore, patterns may show some variability in number of occurrences of some event types. This explains the superiority of the existence approach versus the count technique. Based on these results, we have adopted the existence feature construction process, and all the results presented in the following make use of such scheme.

Monitor Window Length: Second, we have performed an experiment to determine the proper length of monitor windows. The objective of this experiment is to determine to which extent a target event is temporally correlated to previous events.

In figure 4 (left) we plot the average error rate as a function of the monitor window length, for target events 2 and 94. We have tested monitor windows of length that range from 5 up to 100 minutes (at intervals of 5 minutes). We have used a 5 minutes length warning window. We observe the same trend for both target events, but the extent of correlation with the monitoring history is different for the two. The error rate for event type 2 shows a significant drop for monitor windows of length up to 30 minutes. Then, the error rate slowly decreases, and reaches a minimum at 95 minutes. For event type 94 the drop

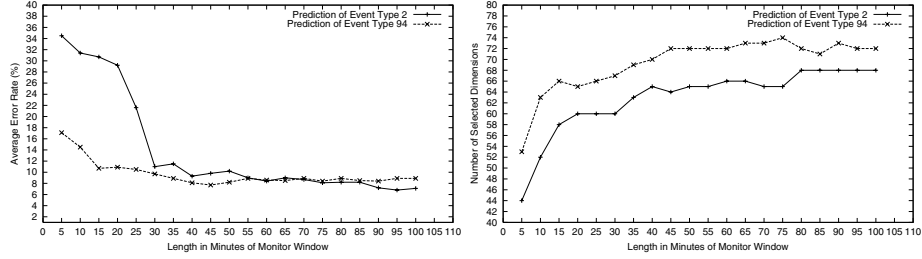


Fig. 4. Prediction of Event Types 2 and 94 using the SVD-SVM technique: (*left*) average error rate (*right*) number of selected dimensions as a function of the length of the monitor window

occurs more rapidly within 15 minutes; then, the error keeps decreasing, and reaches a minimum at 45 minutes.

Interestingly, figure 4 (right) shows an almost mirrored behavior. Here we plot the number of selected dimensions as a function of the monitor window length. In our experiments, the number of selected dimensions corresponds to the number of singular values above the average $\bar{\sigma}$. We observe that the number of such dimensions grows as the length of the monitor window increases, and reaches a stable point when the error rate reaches a minimum (95 minutes for event type 2, and 45 minutes for event type 94).

These results show a nice correlation between the error rate, the number of selected dimensions, and the length of monitor windows: by customizing the length of the monitor window, we are able to capture the useful information, expressed in terms of selected dimensions, i.e., patterns that predict the target, to minimize the prediction error. We emphasize that, although the choice of setting k equal to the number of singular values above the average $\bar{\sigma}$ (figure 1) is a heuristic, we are still able of estimating the intrinsic dimensionality of the data by letting k grow to the value that gives the optimal error rate.

Off-line and On-line Prediction. In table 2 we report the results obtained for the three methods we are comparing. The four columns show: error rate (with standard deviation), error rates for positive and negative examples, and number of selected dimensions. We report the results obtained with C4.5 over the whole feature space, omitting the results obtained over the reduced space, since C4.5 always performed poorly in the second case. For the monitor window length, we have used the optimal values determined in the previous experiment. SVD-SVM and SVM show similar results in both cases, with SVD-SVM selecting 68 (out of 164) dimensions in the first case and 72 in the second. C4.5 is the worst performer in both cases.

We present next the results we have obtained for prediction in an on-line setting. We consider the data preceding a certain timestamp to train a classifier; then we use such classifier for on-line prediction over the remaining time span. To

Table 2. Prediction of Event Types 2 (top 3 rows) and 94

	ERROR	ERROR+	ERROR-	SEL.DIM.
SVD-SVM	6.8 ± 0.2	4.9	8.3	68
C4.5	7.7 ± 1.0	4.8	10.2	164
SVM	7.0 ± 0.2	4.9	8.9	164
SVD-SVM	7.7 ± 0.3	8.0	7.3	72
C4.5	9.3 ± 1.0	9.8	8.2	164
SVM	7.6 ± 0.3	8.4	6.8	164

Table 3. On-line prediction of Event Types 2 (top 3 rows) and 94

	ERROR	ERROR+	ERROR-
SVD-SVM	8.6	12.5	8.5
C4.5	8.4	12.5	8.4
SVM	9.8	12.5	9.8
SVD-SVM	7.2	3.0	7.2
C4.5	34.9	2.4	35.1
SVM	6.6	3.0	6.7

simulate an on-line prediction setting, we consider sliding windows (positioned at each event) of length 95 minutes for event type 2, and of length 45 for event type 94. A warning window of 5 minutes is considered for training in both cases. Therefore, the positive examples for on-line testing are those with an occurrence of the target event within the fifth and sixth minute following the end of the monitor window.

Table 3 shows the results. The number of positive and negative examples used for training is 124 and 160, respectively, for event type 2, and 179, 222 for event type 94. The number of positive and negative examples used for on-line testing is 64 and 9491, respectively, for event type 2, and 165, 19655 for event type 94. Clearly, since target events are rare, the number of tested negative examples is much larger than the positives. We observe that a trivial classifier that always predicts no flaw will make a smaller number of mistakes than SVD-SVM, but it is useless since its recall will always be zero.

On target event 2, all three methods show a similar performance, with SVM being slightly worst (due to a larger number of false positives). On target event 94, SVD-SVM and SVM show a similar performance, whereas C4.5 performs poorly in this case, due to a large number of false positives. By analyzing the tree produced, we see that C4.5 has correctly chosen event type 94 as predictor at the top of the tree (in fact, we did observe that event type 94 is the most relevant predictor of itself). On the other hand, the subtree following the arc

labelled 0 contains event types which are not discriminant; they cause the false positives.

8 Conclusions

We have presented a framework to fully exploit the classification model for event prediction. The accuracy achieved by SVD-SVM throughout our experiments validate the effectiveness of selected features. We have also established the foundations for conducting SVM classification based on statistical properties of event data.

References

1. Bi, Z., Faloutsos, C., Korn, F. (2001). The “DGX” Distribution for Mining Massive, Skewed Data. *International Conference on Knowledge Discovery and Data Mining*. 131
2. Brockwell, P. J., Davis, R. (1996). *Introduction to Time-Series and Forecasting*, Springer-Verlag. 126
3. Cauwenberghs, G., Poggio, T. (2000). Incremental and Decremental Support Vector Machine Learning. *NIPS*. 130
4. Chandrasekharan, S., Manjunath, B. S., Wang, Y. F., Winkeler, J., Zhang, H. (1997). An eigenspace update algorithm for image analysis. *Journal of graphical models and image processing*, 321-332, 59(5). 129
5. Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., Harshman, R. A. (1990). Indexing by latent semantic analysis, *Journal of the American Society for Information Science*, 391-407, 41(6). 125
6. Degroat, R., Roberts, R. (1990). Efficient numerically stabilized rank-one eigenstructure updating. *IEEE transactions on acoustic and signal processing*, 38(2). 129
7. Joachims, T. (1999). Making large-scale SVM learning practical. *Advances in Kernel Methods - Support Vector Learning*, MIT-Press. 132
8. Joachims, T. (2000). *The maximum margin approach to learning text classifiers: Methods, theory, and algorithms*. Doctoral dissertation, Universität Dortmund, Informatik, LS VIII. 126, 130, 131
9. Kanth, K. V. R., Agrawal, D., Singh, A. (1998). Dimensionality Reduction for Similarity Searching in Dynamic Databases. *ACM SIGMOD*. 129
10. Mannila, H., Toivonen, H., Verkamo, A. I. (1995). Discovering frequent episodes in sequences. *International Conference on Knowledge Discovery and Data Mining*. 127
11. Mitra, P., Murthy, C. A., Pal, S. K. (2000). Data Condensation in Large Databases by Incremental Learning with Support Vector Machines. *International Conference on Pattern Recognition*. 130
12. Platt, J. C. (1999). Fast Training of Support Vector Machines using Sequential Minimal Optimization. *Advances in Kernel Methods*, MIT Press, 185-208. 130
13. Syed, N. A., Liu, H., Sung, K. K. (1999). Incremental Learning with Support Vector Machines. *International Joint Conference on Artificial Intelligence*. 130

14. Vilalta, R., Ma, S., Hellerstein, J.(2001). Rule induction of computer events. *IEEE International Workshop on Distributed Systems: Operations & Management*, Springer Verlag, Lecture Notes in Computer Science. 127
15. Weiss, G., Hirsh, H.(1998). Learning to predict rare events in event sequences. *International Conference on Knowledge Discovery and Data Mining*. 127
16. Zaki, M. J.(2001). Sequence mining in categorical domains. *Sequence Learning: Paradigms, Algorithms, and Applications*, pp. 162-187, Springer Verlag, Lecture Notes in Computer Science. 127
17. Zipf, G. K.(1949). *Human behavior and the principle of least effort: An introduction to human ecology*. Addison-Wesley Press. 130

Privacy-Oriented Data Mining by Proof Checking

Amy Felty¹ and Stan Matwin^{2,*}

¹ SITE, University of Ottawa
Ottawa, Ontario K1N 6N5, Canada
afelty@site.uottawa.ca

² LRI – Bôt 490, Université Paris-Sud
91405 ORSAY CEDEX, France
stan@site.uottawa.ca

Abstract. This paper shows a new method which promotes ownership of data by people about whom the data was collected. The data owner may preclude the data from being used for some purposes, and allow it to be used for other purposes. We show an approach, based on checking the proofs of program properties, which implements this idea and provides a tool for a verifiable implementation of the Use Limitation Principle. The paper discusses in detail a scheme which implements data privacy following the proposed approach, presents the technical components of the solution, and shows a detailed example. We also discuss a mechanism by which the proposed method could be introduced in industrial practice.

1 Introduction

Privacy protection, generally understood as “...the right of individuals to control the collection, use and dissemination of their personal information that is held by others” [5], is one of the main issues causing criticism and concern surrounding KDD and data mining. Cheap, ubiquitous and persistent database and KDD resources, techniques and tools provide companies, governments and individuals with means to collect, extract and analyze information about groups of people and individual persons. As such, these tools remove the use of person’s data from their control: a consumer has very little control over the uses of data about her shopping behavior, and even less control over operations that combine this data with data about her driving or banking habits, and perform KDD-type inferences on those combined datasets. In that sense, there is no data ownership by the person whom the data is about. This situation has been the topic of growing concern among people sensitive to societal effects of IT in general, and KDD in particular. Consequently, both macro-level and technical solutions have been proposed by the legal and IT community, respectively.

At the macro-level, one of the main concepts is based on the fact that in most of the existing settings the data collectors are free to collect and use data

* On leave from SITE, University of Ottawa, Canada

as long as these operations are not violating constraints explicitly stated by the individuals whose data are used. The onus of explicitly limiting the access to one's data is on the data owner: this approach is called "opting-out". It is widely felt (e.g. [8]) that a better approach would be opting-in, where data could only be collected with an explicit consent for the collection and specific usage from the data owner.

Another macro concept is the Use Limitation Principle (ULP), stating that the data should be used only for the explicit purpose for which it has been collected. It has been noted, however, that "...[ULP] is perhaps the most difficult to address in the context of data mining or, indeed, a host of other applications that benefit from the subsequent use of data in ways never contemplated or anticipated at the time of the initial collection." [7].

At the technical level, there has been several attempts to address privacy concerns related to data collection by websites, and subsequent mining of this data. The main such proposal is the Platform for Privacy Preferences (P3P) standard, developed by the WWW Consortium [11]. The main idea behind the P3P is a standard by which websites, collecting data from the users, will describe their policies and ULPs in XML. Users, or their browsers, will then decide whether the site's data usage is consistent with the user's constraints on the use of their data, which can also be specified as part of a P3P privacy specification. Although P3P seems to be a step in the right direction, it has some well-known shortcomings. Firstly, the core of a P3P definition for a given website is the description of a ULP, called the P3P policy file. This policy file describes, in unrestricted natural language, what data is collected on this site, how it is used, with whom it is shared, and so on. There are no provisions for enforcement of the P3P policies, and it seems that such provisions could not be incorporated into P3P: the policy description in natural language cannot be automatically verified. The second weakness, noted by [6], is the fact that while P3P provides tools for opting-out, it does not provide tools for opting-in.

The data mining community has devoted relatively little effort to address the privacy concerns at the technical level. A notable exception is the work of R. Agrawal and R. Srikant [2]. In that paper the authors propose a procedure in which some or all numerical attributes are perturbed by a randomized value distortion, so that both the original values and their distributions are changed. The proposed procedure then performs a reconstruction of the original distribution. This reconstruction does not reveal the original values of the data, and yet allows the learning of decision trees which are comparable in their performance to the trees built on the original, "open" data. A subsequent paper [1] shows a reconstruction method which, for large data sets, does not entail information loss with respect to the original distribution. Although this proposal, currently limited to real-valued attributes (so not covering personal data such as SSN, phone numbers etc.) goes a long way towards protecting private data of an individual, the onus of perturbing the data and guaranteeing that the original data is not used rests with the organization performing the data mining. There is no mechanism ensuring that privacy is indeed followed by them.

In this paper we propose a different approach to enforce data ownership, understood as full control of the use of the data by the person whom the data describes. The proposed mechanism can support both the opt-out and the opt-in approach to data collection. It uses a symbolic representation of policies, which makes policies enforceable. Consequently, the proposed approach is a step into the direction of verifiable ULPs.

2 Overall Idea

The main idea of the proposed approach is the following process:

1. Individuals make symbolic statements describing what can and/or cannot be done with specific data about them. These permissions are attached to their data.
2. Data mining and database software checks and respects these statements.

In order to obtain a guarantee that 2) holds regardless of who performs the data mining/database operations, we propose the following additional steps:

- a. Data mining software developers provide, with their software, tools and building blocks with which users of the software can build theorems in a formal language (with proofs), stating that the software respects the user's permissions.
- b. An independent organization is, on request, allowed (remote) access to the environment that includes the data mining software and the theorems with the proofs, and by running a proof checker in this environment it can verify that the permissions are indeed respected by the software.

We can express this idea more formally as a high-level pseudo-code, to which we will refer in the remainder of the paper. We assume that in our privacy-oriented data mining scenario there are the following players and objects:

1. C , an individual (viewed at the same time as a driver, a patient, a student, a consumer, etc.)
2. a set of databases D containing records on different aspects of C 's life, e.g. driving, health, education, shopping, video rentals, etc.
3. a set of algorithms and data mining procedures A , involving records and databases from D , e.g. join two database tables, induce a classification tree from examples, etc.
4. a set (language) of permissions P for using data. P is a set of rules (statements) about elements of D and A . C develops her own set of permissions by choosing and/or combining elements of P and obtains a P_C . P_C enforces C 's ownership of the data. e.g. "my banking record shall not be cross-referenced (joined) with my video rental record" or "I agree to be in a decision tree leaf only with at least 100 other records". Here, we view P_C as a statement which is (or can be translated to) a predicate on programs expressed in a formal logic.

5. *Org* is an organization, e.g. a data mining consultancy, performing operations $a \in A$ on a large dataset containing data about many Cs.
6. S is the source code of $a \in A$, belonging to the data mining tool developer, and B is the executable of S . S may reside somewhere else than at *Org*'s, while B resides with *Org*.
7. A certifiable link $L(B, S)$ exists between B and S , i.e. *Org* and *Veri* (see below) may verify that indeed S = source code of B .
8. $P_C(S)$ is then a theorem stating that S is a program satisfying constraints and/or permissions on the use of data about C .
9. H is a proof checker capable of checking proofs of theorems $P_C(S)$.
10. *Veri*, a Verifier, is a generally trusted organization whose mandate here is to check that *Org* does not breach C 's permission.

The following behavior of the players C , *Org* and *Veri* in a typical data mining exercise is then provably respectful of the permissions of C with respect to their data:

1. *Org* wants to mine some data from D (such that C 's records are involved) with B . This data is referred to as $data_C$.
2. $data_C$ comes packaged with a set of C 's permissions: $data_C \parallel P_C$.
3. *Org* was given by the data mining tool developer (or *Org* itself has built) a proof $R(S, P_C)$ that S respects P_C whenever C 's data is processed. Consequently, due to 7) above, B also respects P_C .
4. *Org* makes $R(S, P_C)$ visible to *Veri*.
5. *Veri* uses P_C and S to obtain $P_C(S)$, and then *Veri* uses H to check that $R(S, P_C)$ is the proof of $P_C(S)$, which means that S guarantees P_C .

We can observe that, by following this scheme, *Veri* can verify that any permissions stated by C are respected by *Org* (more exactly, are respected by the executable software run by *Org*). Consequently, we have a method in which any ULP, expressed in terms of a P_C , becomes enforceable by *Veri*. The P_C permissions can be both “negative”, implementing an opt-out approach, and “positive”, implementing an opt-in approach. The latter could be done for some consideration (e.g. a micropayment) of C by *Org*.

It is important to note that in the proposed scheme there is no need to consider the owner of the data D : in fact, D on its own is useless because it can only be used in the context of P_C s of the different C s who are described by D . We can say that C s represented in D effectively own the data about themselves. Another important comment emphasizes the fact that there are two theorem proving activities involved in the proposed approach: proof construction is done by *Org* or the data mining developer, and proof checking is done by *Veri*. Both have been the topics of active research for more than four decades, and for both, automatic procedures exist. In our approach, we rely on an off-the-shelf solution [10] described in the next section. Between the two, the relatively hard proof construction is left as a one-time exercise for *Org* or for the tool developer where it can be assisted by human personnel, while much easier and faster automatic proof checking procedure is performed by the proposed system.

Let us observe that *Veri* needs the $data_C \parallel P_C, R(S, P_C)$, and S . S will need to be obtained from the data mining tool developer, and P_C can be obtained from C . Only $R(S, P_C)$ is needed from *Org* (access to B will also be needed for the purpose of checking $L(B, S)$). In general, *Veri*'s access to *Org* needs to be minimal for the scheme to be acceptable to *Org*. In that respect, we can observe that *Veri* runs proof checking H on a control basis, i.e. not with every execution of B by *Org*, but only occasionally, perhaps at random time intervals, and even then only using a randomly sampled C . A brief comment seems in order to discuss the performance of the system with a large number of users, i.e. when A works on a D which contains data about many C s. The overhead associated with the processing many C s is linear in their number. In fact, for each $C \in D$ this overhead can be conceptually split into two parts: 1. the proof checking part (i.e. checking the proof of $P_C(S)$), and 2. the execution part (i.e. extra checks resulting from C 's permissions are executed in the code B). The first overhead, which is the expensive one, needs to be performed only once for each C involved in the database. This could be handled in a preprocessing run.

At the implementation level, H could behave like an applet that *Org* downloads from *Veri*.

3 Implementation

Our current prototype implementation uses the Coq Proof Assistant [10]. Coq implements the Calculus of Inductive Constructions (CIC), which is a highly expressive logic. It contains within it a functional programming language that we use to express the source code, S , of the data mining program examples discussed in this paper. Permissions, P_C , are expressed as logical properties about these programs, also using CIC. The Coq system is interactive and provides step-by-step assistance in building proofs. We discuss its use below in building and checking a proof $R(S, P_C)$ of a property P_C of an example program S .

Our main criterion in choosing a proof assistant was that it had to implement a logic that was expressive enough to include a programming language in which we could write our data mining programs, and also include enough reasoning power to reason about such programs. In addition, the programming language should be similar to well-known commonly used programming languages. Among the several that met these criteria, we chose the one we were most familiar with.

3.1 Proof Checking

Proof checking is our enforcement mechanism, insuring that programs meet the permissions specified by individuals. In Coq, proofs are built by entering commands, one at a time, that each contribute to the construction of a CIC term representing the proof. The proof term is built in the background as the proving process proceeds. Once a proof is completed, the term representing the complete proof can be displayed and stored. Coq provides commands to replay and compile completed proofs. Both of these commands include a complete check of the

proof. *Veri* can use them to check whether a given proof term is indeed a proof of a specified property. Coq is a large system, and the code for checking proofs is only a small part of it. All of the code used for building proofs need not be trusted since the proof it builds can be checked after completion. The code for checking proofs is the only part of our enforcement mechanism that needs to be trusted. As stated, *Veri* is a generally trusted organization, so to ensure this trust *Veri* must certify to all others that it trusts the proof checker it is using, perhaps by implementing it itself.

3.2 Verifiable Link between the Source Code and the Object Code

As pointed out in Sect. 2, the scheme proposed here relies on a verifiable link $L(B, S)$ between the source code and the object code of the data mining program. Since theorems and proofs refer to the source programs, while the operations are performed by the object program, and the source S and object B reside with different players of the proposed scheme, we must have a guarantee that all the properties obtained for the source program are true for the code that is actually executed. This is not a data mining problem, but a literature search and personal queries did not reveal an existing solution, so we propose one here.

In a simplistic way, since *Veri* has access to S , *Veri* could compile S with the same compiler that was used to obtain B and compare the result with what *Org* is running. But compilation would be an extremely costly operation to be performed with each verification of $L(B, S)$. We propose a more efficient scheme, based on the concept of digital watermarking. S , which, in practice, is a rich library structure, containing libraries, makefiles etc., is first `tar`'ed. Then the resulting sequential file `tar(S)` is hashed by means of one of the standard hash functions used in the Secure Sockets Layer standard SSL, implemented in all the current Internet browsers. The Message Digest function MD5 [9] is an example of such a file fingerprinting function. The resulting, 128-bit long fingerprint of S is then embedded in random locations within B in the form of `DO NOTHING` instructions whose address part is filled with the consecutive bits forming the result of MD5.

This encoding inside B will originally be produced by a compiler, engineered for this purpose. Locations containing the fingerprint—a short sequence of integer numbers—are part of the definition of $L(B, S)$ and are known to *Veri*. *Veri* needs to produce $\text{MD5}(\text{tar}(S))$ and check these locations within B accordingly. The whole process of checking of $L(B, S)$ can be performed by a specialized applet, ensuring that B is not modified or copied.

3.3 Permissions Language

The logic implemented by Coq is currently used as our language of permissions. More specifically, any predicate expressible in Coq which takes a program as an argument is currently allowed. Each such predicate comes with type restrictions on the program. It specifies what the types of the input arguments to the program must be, as well as the type of the result. An example is given in the next section.

3.4 Issues

The permissions language is currently very general. We plan to design a language that is easy for users to understand and use, and can be translated to statements of theorems in Coq (or some other theorem prover).

As mentioned, a proof in Coq is built interactively with the user supplying every step. Having a smaller permissions language targeted to the data mining application will allow us to clearly identify the class of theorems we want to be able to prove. We will examine this restricted class and develop techniques for automating proof search for it, thus relieving much of the burden of finding proofs currently placed on either the data mining tool developer or *Org*. These automated search procedures would become part of the tools and building blocks provided by data mining software developers.

In our Coq solution described so far, and illustrated by example in the next section, we implement source code S using the programming language in Coq. We actually began with a Java program, and translated it by hand to Coq so that we could carry out the proof. In practice, proofs done directly on actual code supplied by data mining software developers would be much more difficult, but it is important to keep a connection between the two. We would like to more precisely define our translation algorithm from Java to Coq, and automate as much of it as possible. For now, we propose that the data mining tool developers perform the translation manually, and include a description of it as part of the documentation provided with their tools.

In the domain of Java and security, Coq has also been used to reason about the JavaCard programming language for multiple application smartcards [3], and to prove correctness properties of a Java byte-code verifier [4].

4 Example

We present an example program which performs a database join operation. This program accommodates users who have requested that their data not be used in a join operation by ignoring the data for all such users; none of their data will be present in the data output by the program. We present the program and discuss the proof in Coq. We first present the syntax of the terms of CIC used here. Let x and y represent variables and M , N represent terms of CIC. The class of CIC terms are defined using the following grammar.

$$\begin{aligned}
 & Prop \mid Set \mid \\
 M = N \mid & M \wedge N \mid M \vee N \mid M \rightarrow N \mid \neg M \mid \forall x : M. N \mid \exists x : M. N \\
 x \mid & MN \mid [x : M]N \mid x \{y_1 : M_1; \dots; y_n : M_n\} \mid \\
 & Case\ x : M\ of\ M_1 \Rightarrow N_1, \dots, M_n \Rightarrow N_n
 \end{aligned}$$

This set of terms includes both logical formulas and terms of the functional programming language. *Prop* is the type of logical propositions, whereas *Set* is the type of data types. For instance, two data types that we use in our example are

the primitive type for natural numbers and user-defined records. In Coq these types are considered to be members of *Set*. All the usual logical connectives for well-formed formulas are found on the second line. Note that in the quantified formulas, the type of the bound variable, namely M is given explicitly. N is the rest of the formula which may contain occurrences of the bound variable. CIC is a higher-order logic, which means for instance, that quantification over predicates and functions is allowed. On the third line, MN represents application, for example of a function or predicate M to its argument N . We write $MN_1 \dots N_n$ to represent $((MN_1) \dots)N_n$. The syntax $[x : M]N$ represents a parameterized term. For instance, in our example, N often represents a function that takes an argument x of type M .

The term $x \{y_1 : M_1; \dots; y_n : M_n\}$ allows us to define record types, where $y_1, \dots, y_n$ are the field names, $M_1, \dots, M_n$ are their types, and x is the name of the constant used to build records. For example, a new record is formed by writing $xN_1, \dots, N_n$, where for $i = 1, \dots, n$, the term N_i has type M_i and is the value for field y_i . For our example program, we will use three records. One of these records, for example is the following used to store payroll information.

```
Record Payroll : Set :=
  mkPay {PID : nat; JoinInd : bool; Position : string; Salary : nat}.
```

The **Record** keyword introduces a new record in Coq. In this case its name is *Payroll*. The types *nat* and *bool* are primitive types in Coq, and *string* is a type we have defined. The *JoinInd* field is the one which indicates whether or not (value *true* or *false*, respectively) the person who owns this data has given permission to use it in a join operation. The *mkPay* constant is used to build individual records. For example, if n, b, s , and m are values of types *nat*, *bool*, *string*, and *nat*, respectively, then the term $(mkPay \ n \ b \ s \ m)$ is a *Payroll* record whose *PID* value is n , *JoinInd* value is b , etc.

A partial definition of the other two records we use is below.

```
Record Employee : Set :=
  mkEmp {Name : string; EID : nat; ...}.
Record Combined : Set :=
  mkComb {CID : nat; CName : string; CSalary : nat; ...}.
```

The *Employee* record is the one that will be joined with *Payroll*. The *PID* and *EID* fields must have the same value and *JoinInd* must have value *true* in order to perform the join. The *Combined* record is the result of the join. The *CID* field represents the common value of *PID* and *EID*. All other fields come from either one or the other record.

In general, how do the different players know the names of the fields in different *Ds*? Firstly, names of the sensitive fields could be standardized, which in a way is already happening with XML. Alternatively, in a few databases generally relied on, e.g. government health records or driving records, these names would be disclosed to *Veri*. In this example, for simplicity we specify exactly what fields are in each record. We could alternatively express it so that the user's privacy could be ensured independently of the exact form of these records (as long as they both have an *ID* field, and at least one of them has a *JoinInd* field).

The **Definition** keyword introduces a definition in Coq. The following defines a function which takes an *Employee* and *Payroll* record and returns the *Combined* record resulting from their join.

```
Definition mk_Combined : Employee → Payroll → Combined :=
  [E : Employee][P : Payroll]
  (mkComb (EID E) (Name E) (Salary P) ...).
```

The term $(EID\ E)$ evaluates to the value of the *EID* field in record *E*. The *CID* field of the new record is obtained by taking *EID* from *E*, *CName* is obtained by taking *Name* from *E*, *CSalary* is obtained by taking *Salary* from *P*, etc.

The main function implementing the join operation is defined in Coq as:

```
Fixpoint Join [Ps : list Payroll] : (list Employee) → (list Combined) :=
  [Es : list Employee]
  Cases Ps of
    nil          ⇒ (nil Combined)
  | (cons p ps) ⇒ (app (check_JoinInd_and_find_employee_record p Es)
                      (Join ps Es))
end.
```

FixPoint indicates a recursive definition. We represent the set of payroll records in the database using the built in datatype for lists in Coq, and similarly for the other sets. *Join* takes lists *Ps* of payroll records and *Es* of employee records as arguments, and is defined by case on the structure of *Ps* using the *Case* syntax presented above. In general, to evaluate the expression

$$Case\ x : M\ of\ M_1 \Rightarrow N_1, \dots, M_n \Rightarrow N_n$$

the argument *x* of type *M* is matched against the patterns $M_1, \dots, M_n$. If the first one that matches is M_i , then the value N_i is returned. In this example, *Ps* is either the empty list (*nil*) or the list $(cons\ p\ ps)$ with head *p* and rest of the list *ps*. In the first case, an empty list of combined records is returned. In the second case, the function *check_JoinInd_and_find_employee_record* (not shown here) is called. Note that it takes a single *Payroll* record *p* and the entire list of *Employee* records *Es* as arguments. It is defined by recursion on *Es*. If a record in *Es* is found (1) whose *EID* matches the *PID* of *p*, and (2) whose *JoinInd* field has value *true*, then *mk_Combined* is called to join the two records. A list of length 1 containing this record is returned. Otherwise, an empty list of *Combined* records is returned. Function *app* is Coq's append function used to combine the results of this function call with the recursive call to *Join*.

As stated in the previous section, player *C* states permissions as a predicate P_C that must hold of programs *S*. In this example, *Join* is the program *S*. P_C can be expressed as the following definition where *S* is the formal parameter:

```
Definition Pc :=
  [S : ((list Payroll) → (list Employee) → (list Combined)) → Prop]
  ∀Ps : list Payroll. ∀Es : list Employee. (UniqueJoinInd Ps) →
  ∀P : Payroll. (In P Ps) → ((JoinInd P) = false) →
  ¬∃C : Combined. ((In C (S Ps Es)) ∧ ((CID C) = (PID P))).
```

This predicate states that for any payroll record P with a *JoinInd* field with value *false*, there will be no combined record C in the output of the code S such that the *CID* field of C has a value the same as the *PID* field of P .

The theorem that is written $P_C(S)$ in the previous section is obtained in this case by applying the Coq term Pc to *Join* (written $(Pc\ Join)$ in Coq). By replacing the formal parameter S by the actual parameter *Join* and expanding the definition of Pc , we obtain the theorem that we have proved in Coq. A request to Coq's proof checking operation to check this proof is thus a request to verify that the preferences of the user are enforced by the *Join* program.

In the theorem, the constant *In* represents list membership in Coq. The *UniqueJoinInd* predicate is a condition which will be satisfied by any well-formed database with only one payroll record for each *PID*. We omit its definition.

The proof of $(Pc\ Join)$ proceeds by structural induction on the list Ps . It makes use of seven lemmas, and the whole proof development is roughly 300 lines of Coq script. Compiling this proof script (which includes fully checking it) takes 1 second on a 600MHz Pentium III running linux.

5 Acceptance

In a design of a system which would be used by many different players, close attention needs to be paid to their concerns and interests, lest the system will not be accepted. Firstly, individuals C need to be given an easy tool in which to express their positive and negative permissions. In the design of the permissions language, we are taking into account the kind of data being mined (different Ds), and the schema of processing (joins, different classifiers, etc). Initially, a closed set of permissions could be given to them, from which they would choose their preferences. Such permissions could be encoded either on a person's smart card, or in C 's entry in the Public Key Authority directory. More advanced users could use a symbolic language in which to design their permissions. Such a language needs to be designed, containing the typical database and data mining/machine learning operations.

Secondly, who could be the *Veri* organization? It would need to be a generally trusted body with strong enough IT resources and expertise to use a special-purpose proof checker and perform the verifications on which the scheme proposed here is based. One could see a large consumer's association playing this role. Alternatively, it could be a company which makes its mandate fighting privacy abuses, e.g. Junkbusters.

Thirdly, if the scheme gains wider acceptance, developers of data mining tools can be expected to provide theorems (with proofs) that their software S respects the standard permissions that Cs specify and *Veri* supports. These theorems and their proofs will be developed in a standard language known by both the developers and *Veri*; we use Coq as the first conceptual prototype of such a language.

Fourthly, what can be done to make organizations involved in data mining (*Org* in this proposal), and tools providers, accept the proposed scheme? We

believe that it would be enough to recruit one large *Org* and one recognized tool provider to follow the scheme. The fact that, e.g., a large insurance company follows this approach would need to be well publicized in the media. In addition, *Veri* would grant a special logo, e.g. “Green Data Miner”, to any *Org* certified to follow the scheme. The existence of one large *Org* that adheres to this proposal would create a subtle but strong social pressure on others to join. Otherwise, the public would be led to believe that *Orgs* that do not join in fact do not respect privacy of the people whose data they collect and use. This kind of snowball model exists in other domains; it is, e.g., followed by Transparency International.

6 Discussion and Future Work

The paper introduces a new method which implements a mechanism enforcing data ownership by the individuals to whom the data belongs. This is a preliminary description of the proposed idea which, as far as we know, is the first technical solution guaranteeing privacy of data owners understood as their full control over the use of the data, providing verifiable Use Limitation Principle, and supplying a mechanism for opt-in data collection.

The method is based on encoding permissions on the use of the data as theorems about programs that process and mine the data. Theorem proving techniques are then used to express the fact that these programs actually respect the permissions. This initial proposal outlines the method, describes its components, and shows the detailed example of the encoding. We rely on some of the existing tools and techniques for representing the permissions and for checking the theorems about the code that claims to respect them. We also discuss some of the auxiliary techniques needed for the verification.

We are currently working on a prototype of the system described in this paper. This prototype uses some of the Weka’s data mining functions as *A*. We translate the permission-implementing modification of the Weka code into CIC’s functional language and build the proof that the CIC code respects the permission stated above. Coq proof checking then automatically checks that the theorem about the modified code is true, which guarantees that the user’s constraint is respected by the modified Weka code. Furthermore, we are considering how the experience with Weka could be extended to one of the commercial data mining systems.

A lot of work is left to implement the proposed method in a robust and efficient manner, allowing its wide adoption by data mining tool developers and organizations that perform data mining, as well as by the general public. A permission language acceptable for an average user must be designed and tested. A number of tools assisting and/or automating the activities of different players need to be developed. Firstly, a compiler of the permissions language into the formal (here, CIC) statements is needed. Another tool assisting the translation of live code (e.g. Java) into the formal representation (CIC) must also be developed. Our vision is that with the acceptance of the proposed method such for-

mal representation will become part of the standard documentation of the data mining software. Finally, a tool assisting construction of proofs that programs respect the permissions, and eventually building these proofs automatically, is also needed.

An organization sympathetic to the proposed approach and willing to implement and deploy it on a prototype basis needs to be found. This *Org* will not only protect the owners of the data, but can also act as a for-profit data provider. The latter aspect is possible as the proposed method supports an opt-in approach to data collection, based on the user's explicit consent. A commercial mechanism rewarding the opting-in individuals could be worked out by this organization and tested in practice.

Acknowledgements. The authors acknowledge the support of the Natural Sciences and Engineering Research Council of Canada, Computing and Information Technologies Ontario, and the Centre National de la Recherche Scientifique (France). Rob Holte, Francesco Bergadano, Doug Howe, Wladimir Sachs, and Nathalie Japkowicz are thanked for discussing some aspects of the work with us.

References

1. D. Agrawal and C. C. Aggarwal. On the design and quantification of privacy preserving data mining algorithms. In *Twentieth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 247–255. ACM, May 2001. 139
2. R. Agrawal and R. Srikant. Privacy-preserving data mining. In W. Chen, J. F. Naughton, and P. A. Bernstein, editors, *2000 ACM SIGMOD International Conference on Management of Data*, pages 439–450. ACM, May 2000. 139
3. G. Barthe, G. Dufay, L. Jakubiec, B. Serpette, and S. Sousa. A formal executable semantics of the JavaCard platform. In *European Symposium on Programming*, pages 302–319. Springer-Verlag, 2001. 144
4. Y. Bertot. Formalizing a JVMIL verifier for initialization in a theorem prover. In *Computer-Aided Verification*, pages 14–24. Springer-Verlag, 2001. 144
5. Electronic Privacy Information Center and Junkbusters. Pretty poor privacy: An assessment of P3P and internet privacy. <http://www.epic.org/reports/pretypoorprivacy.html>, June 2000. 138
6. K. Coyle. P3P:pretty poor privacy?: A social analysis of the platform for privacy preferences (P3P). <http://www.kcoyle.net/p3p.html>, June 1999. 139
7. Information and Privacy Commissioner/Ontario. Data mining: Staking a claim on your privacy. <http://www.ipc.on.ca/english/pubpres/papers/datamine.htm#Examples>, January 1998. 139
8. D. G. Ries. Protecting consumer online privacy — an overview. http://www.pbi.org/Goodies/privacy/privacy_ries.htm, May 2001. 139
9. R. L. Rivest. RFC 1321: The MD5 message-digest algorithm. Internet Activities Board, 1992. 143
10. The Coq Development Team. The Coq Proof Assistant reference manual: Version 7.2. Technical report, INRIA, 2002. 141, 142
11. W3C. Platform for privacy preferences. <http://www.w3.org/P3P/introduction.html>, 1999. 139

Choose Your Words Carefully: An Empirical Study of Feature Selection Metrics for Text Classification

George Forman

Hewlett-Packard Laboratories
1501 Page Mill Rd. MS 1143
Palo Alto, CA, USA 94304
gforman@hpl.hp.com

Abstract. Good feature selection is essential for text classification to make it tractable for machine learning, and to improve classification performance. This study benchmarks the performance of twelve feature selection metrics across 229 text classification problems drawn from Reuters, OHSUMED, TREC, etc. using Support Vector Machines. The results are analyzed for various objectives. For best accuracy, F-measure or recall, the findings reveal an outstanding new feature selection metric, "Bi-Normal Separation" (BNS). For precision alone, however, Information Gain (IG) was superior. A new evaluation methodology is offered that focuses on the needs of the data mining practitioner who seeks to choose one or two metrics to try that are mostly likely to have the best performance for the single dataset at hand. This analysis determined, for example, that IG and Chi-Squared have correlated failures for precision, and that IG paired with BNS is a better choice.

1 Introduction

As online resources continue to grow exponentially, so too will the need to improve the efficiency and accuracy of machine learning methods: to categorize, route, filter and search for relevant text information. Good feature selection can (1) improve classification accuracy or equivalently, reduce the amount of training data needed to obtain a desired level of performance and (2) conserve computation, storage and network resources needed for training and all future use of the classifier. Conversely, poor feature selection limits performance no degree of clever induction can make up for a lack of predictive signal in the input features.

This paper presents the highlights of an empirical study of twelve feature selection metrics on 229 text classification problem instances drawn from 19 datasets that originated from Reuters, OHSUMED, TREC, etc. [3]. (For more details of the study than space permits here, see [1].) We analyze the results from various perspectives,

including accuracy, precision, recall and F-measure, since each is appropriate in different situations. Further, we introduce a novel analysis that is focused on a subtly different goal: to give guidance to the data mining practitioner about which feature selection metric or combination is *most likely* to obtain the best performance for the *single given* dataset they are faced with, supposing their text classification problem is drawn from a distribution of problems similar to that studied here.

Our primary focus is on obtaining the best overall classification performance regardless of the number of features needed to obtain that performance. We also analyze which metrics excel for small sets of features, which is important for situations where machine resources are severely limited, low latency classification is needed, or large scalability is demanded.

The results on these benchmark datasets showed that the well-known Information Gain metric was not best for the goals of F-measure, Recall or Accuracy, but instead an outstanding new feature selection metric, χ^2 -Bi-Normal Separation. For the goal of Precision alone, however, Information Gain was superior.

In large text classification problems, there is typically a substantial skew in the class distribution. For example, in selecting news articles that best match one's personalization profile, the positive class of interest contains many fewer articles than the negative background class. For multi-class problems, the skew increases with the number of classes. The skew of the classification problems used in this study is 1:31 on average, and ~4% exceed 1:100. High class skew presents a particular challenge to induction algorithms, which are hard pressed to beat the high accuracy achieved by simply classifying everything as the negative majority class. For this reason, accuracy scores can under-represent the value of good classification. Precision and recall are often preferable measures for these situations, or their harmonic average, F-measure.

High class skew makes it that much more important to supply the induction algorithm with well chosen features. In this study, we consider each binary class decision as a separate problem instance and select features for it alone. This is the natural setting for 2-class problems, e.g. in identifying spam vs. valuable email. This is also an important subcomponent for good multi-class feature selection [2], i.e. determining a fixed set of features for multiple 2-class problems (aka n -of- m , *topic or keyword identification*), or for k -of- m multi-class problems, e.g. determining where to file a new item for sale in the large Ebay.com classified ad categories.

The choice of the induction algorithm is not the object of study here. Previous studies have shown Support Vector Machines (SVM) to be a consistent top performer [e.g. 6], and a pilot study comparing the use of the popular Naïve Bayes algorithm, logistic regression, and C4.5 decision trees confirmed the superiority of SVM. (When only a small number of features are selected, however, we found Naïve Bayes to be the best second choice, compared to the others.)

Related Work: For context, we mention that a large number of studies on feature selection have focused on non-text domains. These studies typically deal with much lower dimensionality, and often find that wrapper methods perform best. Wrapper methods, such as sequential forward selection or genetic search, perform a search over the space of all possible subsets of features, repeatedly calling the induction algorithm as a subroutine to evaluate various subsets of features. For high-dimensional problems, however, this approach is intractable, and instead feature scoring metrics

are used independently on each feature. This paper is only concerned with feature scoring metrics; nevertheless, we note that advances in scoring methods should be welcome to wrapper techniques for use as heuristics to guide their search more effectively.

Previous feature selection studies for *text* domain problems have not considered as many datasets, tested as many metrics, nor considered support vector machines. For example, the valuable study by Yang and Pedersen [7] considered five feature selection metrics on the standard Reuters dataset and OHSUMED. It did not consider SVM, which they later found to be superior to the algorithms they had studied, LLSF and kNN [6]. The question remains then: do their findings generalize to SVM?

Such studies typically consider the problem of selecting one set of features for 1-of-m or n-of-m multi-class problems. This fails to explore the best possible accuracy obtainable for any single class, which is especially important for high class skew. Also, as pointed out in [2], all feature scoring metrics can suffer a blind spot for multi-class problems when there are many good predictive features available for one or a few easy classes that overshadow the useful features for difficult classes.

This study also recommends feature selection strategies for varied situations, e.g. different tradeoffs between precision and recall, and for when resources are tight.

2 Feature Selection Methods

The overall feature selection procedure is to score each potential word/feature according to a particular feature selection metric, and then take the best k features. Scoring a feature involves counting its occurrences in training examples for the positive and the negative classes separately, and then computing a function of these.

In addition, there are some other filters that are commonly applied. First, rare words may be eliminated, on the grounds that they are unlikely to be present to aid any given classification. For example, on a dataset with thousands of words, those occurring two or fewer times may be removed. Word frequencies typically follow a Zipf distribution ($\sim 1/\text{rank}^p$). Easily half the total number of unique words may occur only a single time, so eliminating words under a given low rate of occurrence yields great savings. The particular choice of threshold can have an effect on accuracy, and we consider this further in our evaluation. If we eliminate rare words based on a count from the whole dataset *before* we split off a training set, we have leaked some information about the test set to the training phase. Without expending a great deal more resources for cross-validation studies, this research practice is unavoidable, and is considered acceptable in that it does not use the class labels of the test set.

Additionally, overly common words, such as `a` and `of`, may also be removed on the grounds that they occur so frequently as to not be discriminating for any particular class. Common words can be identified either by a threshold on the number of documents the word occurs in, e.g. if it occurs in over half of all documents, or by supplying a *stopword* list. Stopwords are language-specific and often domain-specific. Depending on the classification task, they may run the risk of removing words that are essential predictors, e.g. the word `can` is discriminating between `aluminum` and `glass` recycling.

It is also to be mentioned that the common practice of *stemming* or *lemmatizing*—merging various word forms such as plurals and verb conjugations into one distinct term—also reduces the number of features to be considered. It is properly considered, however, a *feature engineering* option.

An ancillary feature engineering choice is the representation of the feature value. Often a Boolean indicator of whether the word occurred in the document is sufficient. Other possibilities include the count of the number of times the word occurred in the document, the frequency of its occurrence normalized by the length of the document, the count normalized by the inverse document frequency of the word. In situations where the document length varies widely, it may be important to normalize the counts. For the datasets included in this study, most documents are short, and so normalization is not called for. Further, in short documents words are unlikely to repeat, making Boolean word indicators nearly as informative as counts. This yields a great savings in training resources and in the search space of the induction algorithm. It may otherwise try to discretize each feature optimally, searching over the number of bins and each bin's threshold. For this study, we selected Boolean indicators for each feature. This choice also widens the choice of feature selection metrics that may be considered, e.g. Odds Ratio deals with Boolean features, and was reported by Mladenic and Grobelnik to perform well [5].

A final choice in the feature selection policy is whether to rule out all negatively correlated features. Some argue that classifiers built from positive features only may be more transferable to new situations where the background class varies and retraining is not an option, but this benefit has not been validated. Additionally, some classifiers work primarily with positive features, e.g. the Multinomial Naïve Bayes model, which has been shown to be both better than the traditional Naïve Bayes model, and considerably inferior to other induction methods for text classification [e.g. 6]. Negative features are numerous, given the large class skew, and quite valuable in practical experience. For example, when scanning a list of Web search results for the author's home page, a great number of hits on George Foreman the boxer show up and can be ruled out strongly via the words "boxer" and "champion," of which the author is neither. The importance of negative features is empirically confirmed in the evaluation.

2.1 Metrics Considered

Here we enumerate the feature selection metrics we evaluated. In the interest of brevity, we omit the equations and mathematical justifications for the metrics that are widely known (see [1,5,7]). Afterwards, we show a novel graphical analysis that reveals the widely different decision curves they induce. Paired with an actual sample of words, this yields intuition about their empirical behavior.

Notation: $P(+)$ and $P(-)$ represent the probability distribution of the positive and negative classes; pos is the number of documents in the positive class. The variables tp and fp represent the raw word occurrence counts in the positive and negative classes, and tpr and fpr indicate the sample true-positive-rate, $P(\text{word}|+)$, and false-positive-rate, $P(\text{word}|-)$. These summary statistics are appropriate for Boolean features. Note that any metric that does not have symmetric values for negatively correlated features

is made to value negative features equally well by inverting the value of the feature, i.e. $tpr' = 1 - tpr$ and $fpr' = 1 - fpr$, without reversing the classes.

Commonly Used Metrics:

Chi: Chi-Squared measures the divergence from the expected distribution assuming the feature is actually independent of the class value.

IG: Information Gain measures the decrease in entropy when given the feature. Yang and Pederson reported IG and Chi performed very well [7].

Odds: Odds Ratio reflects the probability ratio of the (positive) class given the feature. In the study by Mladenic and Grobelnik [5] it yielded the best F-measure for Multinomial Naïve Bayes, which works primarily from positive features.

DFreq: Document Frequency simply measures in how many documents the word appears, and can be computed without class labels. It performed much better than Mutual Information in the study by Yang and Pedersen, but was consistently dominated by IG and Chi.

Additional Metrics:

Rand: Random ranks all features randomly and is used as a baseline for comparison. Interestingly, it scored highest for precision in the study [5], although this was not considered valuable because its recall was near zero, yielding the lowest F-measure scores.

Acc: Accuracy estimates the expected accuracy of a simple classifier built from the single feature, i.e. $P(1 \text{ for } + \text{ class and } 0 \text{ for } - \text{ class}) = P(1|+)P(+) + P(0|-)P(-) = tpr P(+) + (1-fpr) P(-)$, which simplifies to the simple decision surface $tp \geq fp$. Note that it takes the class skew into account. Since $P(-)$ is large, fpr has a strong influence. When the classes are highly skewed, however, better accuracy can sometimes be achieved simply by always categorizing into the negative class.

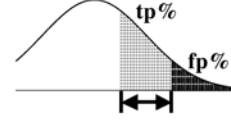
Acc2: Accuracy2 is similar, but supposes the two classes were balanced in the equation above, yielding the decision surface $tpr - fpr$. This removes the strong preference for low fpr .

F1: F₁-measure is the harmonic mean of the precision and recall: $2 \text{ recall precision} / (\text{recall} + \text{precision})$, which simplifies to $2 tp / (pos + tp + fp)$. This metric is motivated because in many studies the F-measure is the ultimate measure of performance of the classifier. Note that it focuses on the positive class, and that negative features, even if inverted, are devalued compared to positive features. This is ultimately its downfall as a feature selection metric.

OddN: Odds Numerator is the numerator of Odds Ratio, i.e. $tpr * (1-fpr)$.

PR: Probability Ratio is the probability of the word given the positive class divided by the probability of the word given the negative class, i.e. tpr/fpr . It induces the same decision surface as $\log(tpr/fpr)$, which was studied in [5]. Since it is not defined at $fpr=0$, we explicitly establish a preference for features with higher tp counts along the axis by substituting $fpr'=1e-8$.

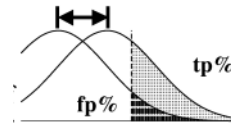
BNS: Bi-Normal Separation is a new feature selection metric we defined as $F^{-1}(tpr) - F^{-1}(fpr)$, where F^{-1} is the standard Normal distribution's inverse cumulative probability function. For intuition, suppose the occurrence of a given



feature in each document is modeled by the event of a random Normal variable exceeding a hypothetical threshold. The prevalence rate of the feature corresponds to the area under the curve past the threshold. If the feature is more prevalent in the positive class, then its threshold is further from the tail of the distribution than that of the negative class. The BNS metric measures the separation between these thresholds.

An alternate view is motivated by ROC threshold analysis:

The metric measures the horizontal separation between two standard Normal curves where their relative position is uniquely prescribed by tpr and fpr , the area under the tail of each curve (cf. a traditional hypothesis test where tpr and fpr



estimate the center of each curve). The BNS distance metric is therefore proportional to the area under the ROC curve generated by the two overlapping Normal curves, which is a robust method that has been used in the medical testing field for fitting ROC curves to data in order to determine the efficacy of a treatment. Its justifications in the medical literature are many and diverse, both theoretical and empirical [4].

Pow: Pow is $(1-fpr)^k \square (1-tpr)^k$, where k is a parameter. It is theoretically unmotivated, but is considered because it prefers frequent terms [7], aggressively avoids common fp words, and can generate a variety of decision surfaces given parameter k , with higher values corresponding with a stronger preference for positive words. This leaves the problem of optimizing k . We chose $k=5$ after a pilot study.

2.2 Graphical Analysis

In order to gain a more intuitive grasp for the selection biases of these metrics, we present in Figure 1 the actual decision curves they induce in ROC space—true positives vs. false positives—when selecting exactly 100 words for distinguishing abstracts of general computer science papers vs. those on probabilistic machine learning techniques. The horizontal axis represents far more negative documents (1750) than the vertical axis (50), for a skew of 1:35. The triangle below the diagonal represents negatively correlated words, and the symmetrically inverted decision curves are shown for each metric. We see that Odds Ratio and BNS treat the origin and upper right corner equivalently, while IG and Chi progressively cut off the top right— and symmetrically the bottom left, eliminating many negative features.

The dots represent the specific word features available in this problem instance—note that there are many words sharing the same tp and fp counts near the origin, but the black and white visualization does not indicate the many collisions. Very few words have high frequency and they also tend to be non-predictive, i.e. they stay close to the diagonal as they approach the upper right corner. This partly supports the practice of eliminating the most frequent words (the bold dotted line depicts a cut-off threshold that eliminates words present in $> \square$ of all documents), but note that it saves only 28 words out of 12,500.

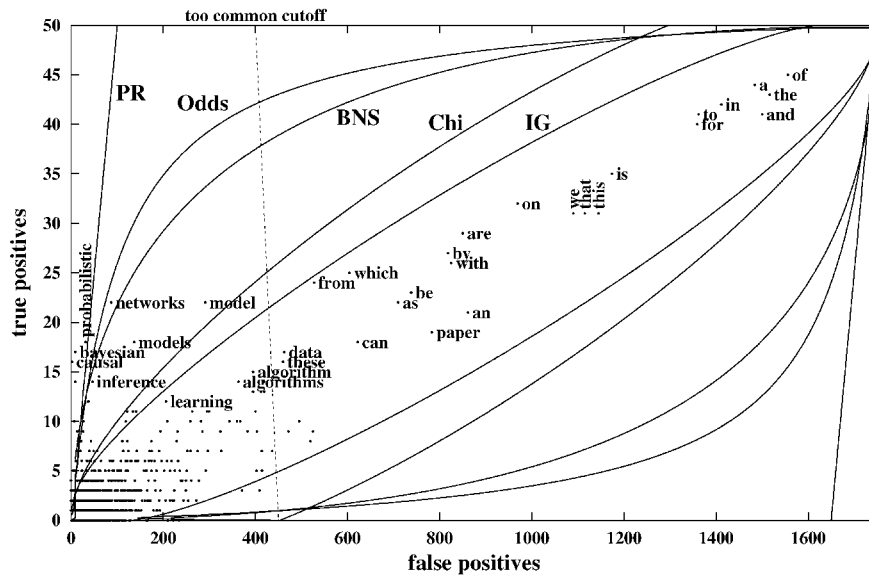


Fig. 1. Decision boundary curves for the feature selection metrics Probability Ratio, Odds Ratio, Bi-Normal Separation, Chi-Squared, and Information Gain. Each curve selects the “best” 100 words, each according to its view, for discriminating abstracts of data mining papers from others. Dots represent actual words, and many of the 12K words overlap near the origin

Since word frequencies tend toward a Zipf distribution, most of the potential word features appear near the origin. This implies that feature selection is most sensitive to the shape of the decision curve in these dense regions. Figure 2 shows a zoomed-in view where most of the words occur. The bold diagonal line near the origin shows a rare word cutoff of <3 occurrences, which eliminates 7333 words for this dataset. This represents substantial resource savings ($\sim 60\%$) and the elimination of fairly uncertain words that are unlikely to re-occur at a rate that would be useful for classification.

3 Experimental Method

Performance Measures: While several studies have sought solely to maximize the F-measure, there are common situations where precision is to be strongly preferred over recall, e.g. when the cost of false positives is high, such as mis-filtering a legitimate email as spam. Precision should also be the focus when delivering Web search results, where the user is likely to look at only the first page or two of results. Finally, there are situations where accuracy is the most appropriate measure, even when there is high class skew, e.g. equal misclassification costs. For these reasons, we analyze the performance for each of the four performance goals.

There are two methods for averaging the F-measure over a collection of 2-class classification problems. One is the *macro-averaged F-measure*, which is the traditional arithmetic mean of the F-measure computed for each problem. Another is the *micro-averaged F-measure*, which is an average weighted by the class

distribution. The former gives equal weight to each problem, and the latter gives equal weight to each document classification (which is equivalent to overall accuracy for a 1-of-m problem). Since highly skewed, small classes tend to be more difficult, the macro-averaged F-measure tends to be lower. We focus on macro-averaging because we are interested in average performance across different problems, without regard to the problem size of each. (To measure performance for a given problem instance, we use 4-fold stratified cross-validation, and take the average of 5 runs.)

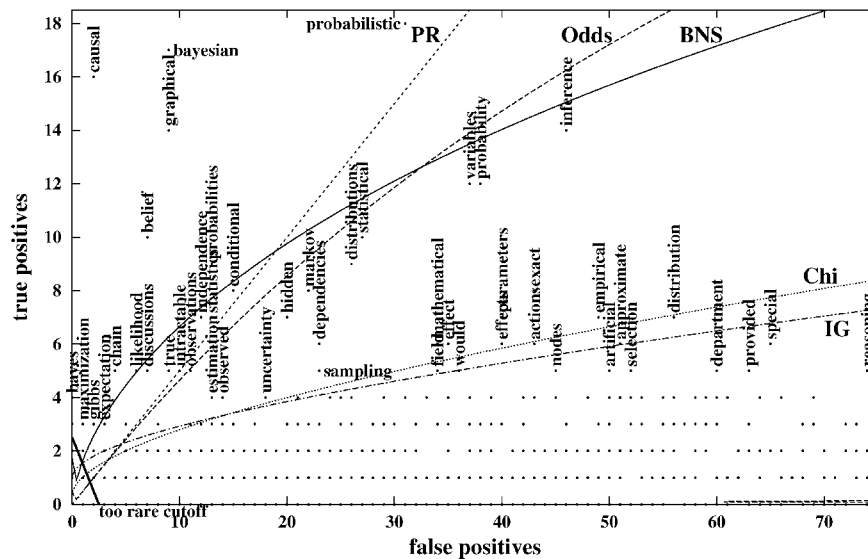


Fig. 2. Zoomed-in version of Figure 1, detailing where most words occur

A data mining practitioner has a different goal in mind—to choose a feature selection technique that maximizes their *chances* of having the best metric *for their single dataset of interest*. Supposing the classification problems in this study are representative of problems encountered in practice, we compute for each metric, the percentage of problem instances for which it was optimal, or within a given error tolerance of the best method observed for that instance.

Induction Algorithm: We performed a brief pilot study using a variety of classifiers, including Naïve Bayes, C4.5, logistic regression and SVM with a linear kernel (each using the WEKA open-source implementation with default parameters). The results confirmed previous findings that SVM is an outstanding method [6], and so the remainder of our presentation uses it alone. It is an interesting target for feature selection because no comparative text feature selection studies have yet considered it, and its use of features is entirely along the decision boundary between the positive and negative classes, unlike many traditional induction methods that model the density. We note that the traditional Naïve Bayes model fared better than C4.5 for these text problems, and that it was fairly sensitive to feature selection, having its performance peak at a much lower number of features selected.

Datasets: We were fortunate to obtain a large number of text classification problems in preprocessed form made available by Han and Karypis, the details of which are laid out in [3] and in the full version of this study [1]. These text classification problems are drawn from the well known Reuters, OHSUMED and TREC datasets. In addition, we included a dataset of abstracts of computer science papers gathered from Cora.whizbang.com that were categorized into 36 classes, each containing 50 training examples. Taken altogether, these represent 229 two-class text classification problem instances, with a positive class size of 149 on average, and class skews averaging 1:31 (median 1:17, 5th percentile 1:3, 95th 1:97, max 1:462).

Feature Engineering and Selection: Each feature represents the Boolean occurrence of a forced-lowercase word. Han [3] reports having applied a stopword list and Porter's suffix-stripping algorithm. From an inspection of word counts in the data, it appears they also removed rare words that occurred <3 times in most datasets. Stemming and stopwords were not applied to the Cora dataset, and we used the same rare word threshold. We explicitly give equal importance for negatively correlated word features by inverting *tpr* and *fpr* before computing the feature selection metric. We varied the number of selected features in our experiments from 10 to 2000. Yang and Pedersen evaluated up to 16,000 words, but the F-measure had already peaked below 2000 for Chi-Squared and IG [7]. If the features are selected well, most of the information should be contained in the initial features selected.

4 Empirical Results

Figure 3 shows the macro-averaged F-measure for each of the feature selection metrics as we vary the number of features to select. The absolute values are not of interest here, but rather the overall trends and the separation of the top performing curves. We see that to maximize the F-measure on average, BNS performed best by a wide margin, using 500 to 1000 features. This is a significant result in that BNS has not been used for feature selection before, and the significance level, even in the barely visible gap between BNS and IG at 100 features, is greater than 99.9% confidence in a paired t-test of the 229*5 runs. Like the results of Yang and Pedersen [7], performance begins to decline around 2000 features.

If for scalability reasons one is limited to 20-50 features, a better metric to use is IG (or Acc2, which is simpler to program. Surprisingly, Acc2, which ignores class skew, performs much better than Acc, which accounts for skew.). IG dominates the performance of Chi at every size of feature set.

Accuracy: The results for accuracy are much the same, and their graphs must be omitted for space (but see [1]). BNS again performed the best by a smaller, but still $>99.9\%$ confident, margin. At 100 features and below, however, IG performed best, with Acc2 being statistically indistinguishable at 20 features.

Precision-Recall Tradeoffs: As discussed, one's goal in some situations may be solely precision or recall, rather than F-measure. Figure 4 shows this tradeoff for each metric, macro-averaged across all sample problems and evaluated at 1000 features selected. We see that the success of BNS with regard to its high F-measure is because

it obtains on average much higher recall than any other method. If, on the other hand, precision is the sole goal, IG is the best at any number of features (and Chi is statistically indistinguishable over 1000 features).

4.1 Best Chances of Obtaining Maximum Performance

The problem of choosing a feature selection metric is somewhat different when viewed from the perspective of a data mining practitioner whose task is to get the best performance on a *given* set of data, rather than averaging over a large number of datasets. Practitioners would like guidance as to which metric is most *likely* to yield the best performance for their single dataset at hand. Supposing the problem instance is drawn from a distribution similar to that in this study, we offer the following analysis: For each feature selection metric, we determine the percentage of the 229 problem instances for which it matched the best performance found within a small tolerance (taking the maximum over any number of features). We repeat this separately for F-measure, precision, recall and accuracy.

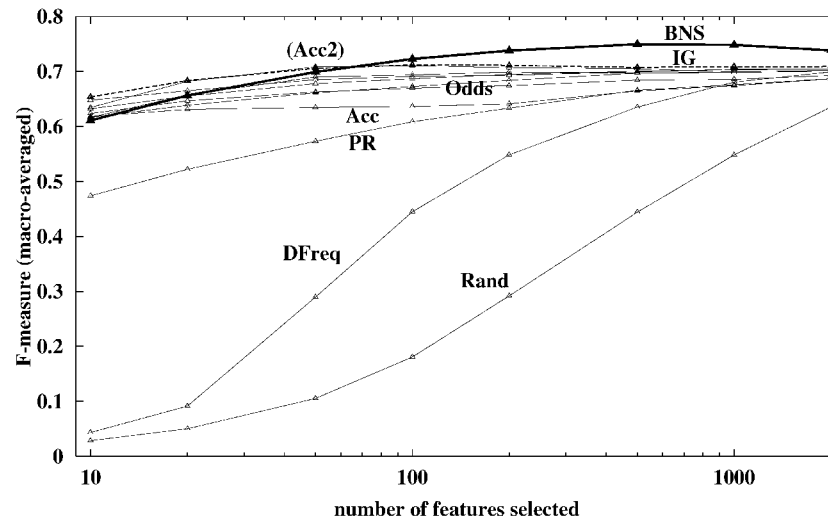


Fig. 3. F-measure averaged over 229 problems for each metric & number of features

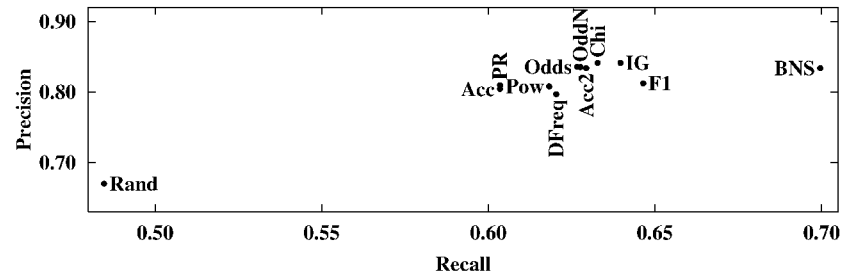


Fig. 4. Precision-Recall tradeoffs at 1000 features from Fig. 3

Figure 5a shows these results for the goal of maximum F-measure as we vary the acceptable tolerance from 1% to 10%. As it increases, each metric stands a greater chance of obtaining close to the maximum, thus the trend. We see that BNS attained within 1% of best performance for 65% of the 229 problems, beating IG at just 40%. Figure 5b shows similar results for Accuracy, F-measure, Precision and Recall (but using 0.1% tolerance for accuracy, since large class skew compresses the range). Note that for precision, several metrics beat BNS, notably IG. This is seen more clearly in Figure 6a, which shows these results for varying tolerances. IG consistently dominates at higher tolerances, though the margin is less striking than Figure 5a.

Residual Analysis: If one were willing to invest the extra effort to try two different metrics for one's dataset and select the one with better precision via cross-validation, the two leading metrics, IG and Chi, would seem a logical choice. However, it may be that wherever IG fails to attain the maximum, Chi also fails. To evaluate this, we repeated the procedure considering the maximum performance of pairs of feature selection metrics. Figure 6b shows these results for each metric paired with IG. Observe that, surprisingly, IG+Chi performed the worst of the pairs, validating the hypothesis that it has correlated failures. BNS, on the other hand, has uncorrelated failures and so, paired with IG, gives the best precision, and by a significant margin.

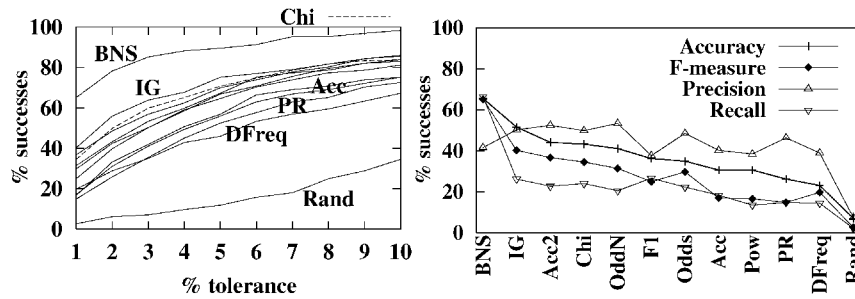


Fig. 5. (a) Percentage of problems on which each metric scored within $x\%$ tolerance of the best F-measure of any metric. (b) Same, for F-measure, recall, and precision @1%, accuracy @0.1%

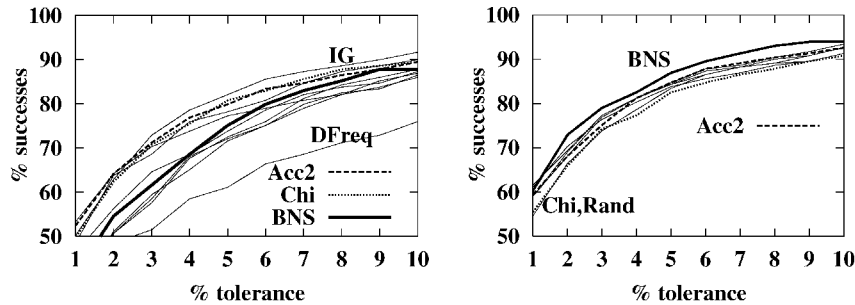


Fig. 6. (a) As Figure 5a, but for precision. (b) Same, but each metric is combined with IG

This paired analysis was repeated for F-measure, Recall and Accuracy, and consistently revealed that BNS paired with IG sustained the best performance.

Due to space limitations, refer to [1] for the complete results, as well as related experiments we performed. Below we briefly mention two of the ancillary findings:

Lesion Study on Negative Features: In the experiments above, we inverted negative features so that they would be treated identically to positive features, creating the symmetrical decision surfaces seen in Figure 1. In a related suite of experiments, we suppressed negative features altogether to determine their importance. When deprived of negative features, no feature selection metric was competitive with the previous results for BNS, IG or Acc2. We conclude that negative features are essential to high quality classification.

Sensitivity to the Rare Word Cutoff: In the existing datasets, words were removed that occurred fewer than 3 times in the (training & testing) corpus. Some text preparation practices use a much higher threshold to reduce the size of the data. We performed a suite of experiments on the Cora dataset, varying this threshold up to 25, which eliminates the majority of the potential features from which to select. The macro-averaged F-measure, precision and accuracy (for BNS at 1000 features selected) each decline steadily as the threshold increases; recall rises slightly up to a threshold of 10, and then falls off. We conclude that to reduce space, one should set the rare word cutoff low, and then perform aggressive feature selection using a metric.

5 Conclusion

This paper presented an extensive comparative study of feature selection metrics for the text domain, focusing on support vector machines and 2-class problems, typically with high class skew. It revealed an outstanding new feature selection metric, Bi-Normal Separation, which is mathematically equivalent to the bi-normal assumption that has been used in the medical field for fitting ROC curves [4]. Another contribution of this paper is a novel evaluation methodology that considers the common problem of trying to select one or two metrics that have the best chances of obtaining the best performance for a *given* dataset. Somewhat surprisingly, selecting the two best performing metrics is sub-optimal, because when the best metric fails, the other may have correlated failures, as occurs for IG and Chi. Pairing IG with BNS is consistently a better choice. The reader is referred to [1] for additional results.

Acknowledgements

We wish to thank the WEKA project for their software, the ID Laboratory/INRIA for use of their hardware, and Han & Karypis and Tom Fawcett for the prepared datasets.

References

1. Forman, G.: An Extensive Empirical Study of Feature Selection Metrics for Text Classification. Tech Report HPL-2002-147, Hewlett-Packard Laboratories. Submitted to Special Issue on Variable and Feature Selection, J. of Machine Learning Research. (2002)
2. Forman, G.: Avoiding the Siren Song: Undistracted Feature Selection for Multi-Class Text Classification. TR HPL-2002-75, Hewlett-Packard Laboratories. Submitted as above. (2002)
3. Han, E. S., Karypis, G.: Centroid-Based Document Classification: Analysis & Experimental Results. In: Principles of Data Mining and Knowledge Discovery (PKDD). (2000) 424-431
4. Hanley, J. A.: The Robustness of the "Binormal" Assumptions Used in Fitting ROC Curves. Medical Decision Making, 8(3). (1988) 197-203
5. Mladenic, D., Grobelnik, M.: Feature Selection for Unbalanced Class Distribution and Naïve Bayes. In: 16th International Conference on Machine Learning (ICML). (1999)
6. Yang, Y., Liu, X.: A Re-examination of Text Categorization Methods. In: ACM SIGIR Conference on Research and Development in Information Retrieval. (1999) 42-49
7. Yang, Y., Pedersen, J. O.: A Comparative Study on Feature Selection in Text Categorization. In: International Conference on Machine Learning (ICML). (1997) 412-420

Generating Actionable Knowledge by Expert-Guided Subgroup Discovery

Dragan Gamberger¹ and Nada Lavrač²

¹ Rudjer Bošković Institute
Bijenička 54, 10000 Zagreb, Croatia
`dragan.gamberger@irb.hr`

² Jožef Stefan Institute
Jamova 39, 1000 Ljubljana, Slovenia
`nada.lavrac@ijs.si`

Abstract. This paper discusses actionable knowledge generation. Actionable knowledge is explicit symbolic knowledge, typically presented in the form of rules, that allows the decision maker to recognize some important relations and to perform an action, such as targeting a direct marketing campaign, or planning a population screening campaign aimed at targeting individuals with high disease risk. The disadvantages of using standard classification rule learning for this task are discussed, and a subgroup discovery approach proposed. This approach uses a novel definition of rule quality which is extensively discussed.

1 Introduction

In KDD one can distinguish between *predictive* and *descriptive* induction tasks. Classification rule learning [2,10] is a form of *predictive induction*. The distinguishing feature of predictive induction is the input data formed of labeled training examples (with class assigned to each training instance), and the output aimed at solving classification and prediction tasks. This paper provides arguments for actionable knowledge generation through recently developed *descriptive induction* approaches. These involve mining of association rules (e.g., APRIORI [1]), subgroup discovery (e.g., MIDOS [16]), and other approaches to non-classificatory induction. In this work we are particularly interested in subgroup discovery, where a subgroup discovery task can be defined as follows: given a population of individuals and a property of those individuals we are interested in, find population subgroups that are statistically ‘most interesting’, e.g., are as large as possible and have the most unusual statistical (distributional) characteristics with respect to the property of interest.

Actionable knowledge is explicit symbolic knowledge that allows the decision maker to perform an action, such as, for instance, select customers for a direct marketing campaign, or select individuals for population screening concerning high disease risk. The term *actionability* [14] denotes a *subjective* measure of

interestingness of a discovered pattern: “a pattern is interesting if the user can do something with it to his or her advantage” [13,14].¹

This paper presents some shortcomings of actionable knowledge generation through predictive induction and proposes an approach to expert-guided knowledge discovery, where the induction task is to detect different, potentially important subgroups among which the expert will be able to select the patterns which are actionable. The paper is organized as follows. Section 2 discusses the types of induced knowledge and shortcomings of standard classification rule learning for actionable knowledge generation. Section 3 presents the advantages of subgroup discovery approaches for the formation of actionable knowledge, and proposes an approach to subgroup discovery, developed by adapting an existing confirmation rule learning algorithm. We conclude with some experimental evaluation results in Section 4 and lessons learned in Section 5.

2 Shortcomings of Classification Rule Learning for Actionable Knowledge Generation

In symbolic predictive induction, two most common approaches are rule learning and decision tree learning. The goal of rule learning is to generate separate models, one for each class, inducing class characteristics in terms of class properties occurring in the descriptions of training examples. Classification rule learning results in *characteristic descriptions*, usually generated separately for each class by repeatedly applying the covering algorithm. In decision tree learning, on the other hand, the rules which can be formed of paths leading from the root node to class labels in the leaves represent *discriminating descriptions*, formed of properties that best discriminate between the classes. Hence, classification rules serve two different purposes: characterization and discrimination.

An open question, discussed in this paper, is whether the knowledge induced by rule learning and decision tree learning is actionable in medical and marketing applications, outlined in this paper, whose goal is to uncover the properties of subgroups of the population which can guide a decision maker in directing some targeted campaign. The motivation for this work comes from two applications

- A medical problem of population screening aimed at spotting the individuals in a town or region with high risk for Coronary Heart Disease (CHD) [5]. In this application, the hard problem is to find suspect CHD cases with slightly abnormal values of risk parameters and in cases when combinations of different risk factors occur. The risk group models should help general practitioners to recognize CHD and/or to detect the illness even before the first symptoms actually occur.
- A marketing problem of direct mailing aimed at spotting potential customers of a certain product [3]. In this application, the problem is to select subgroups of potential customers that can be targeted by an advertising campaign. The

¹ The other subjective measure introduced in [14] is unexpectedness: “a pattern is interesting to the user if it is surprising to the user”.

specific task is to find significant characteristics of customer subgroups who do not know a brand, relative to the characteristics of the population that recognizes the brand.

We argue that for such and similar tasks the models induced through classification rule learning and decision tree learning are not actionable. Besides subjective reasons [14] that can be due to the inappropriate choice of parameters used in induced descriptions, some objective reasons for the non-actionability of induced patterns that are due to the method used are listed below:

- Classification rules and decision trees could be used to classify all individuals of a selected population, but this is unpractical and virtually impossible.
- Rules formed of decision tree paths are discriminant descriptions, hence they are not actionable for the above tasks.
- Classification rules forming characteristic descriptions are intuitively expected to be actionable. However, the fact that they have been generated by a covering algorithm (used in AQ [10], CN2 [2], and most other rule learners) hinders their actionability. Only first few rules induced by a covering algorithm may be of interest as subgroup descriptions with sufficient coverage. Subsequent rules are induced from smaller and strongly biased example subsets, e.g., subsets including only positive examples not covered by previously induced rules. This bias prevents a covering algorithm to induce descriptions uncovering significant subgroup properties of the entire population.

A deeper analysis of the reasons for the non-actionability of patterns induced by decision tree and classification rule induction can be found in [9]. Our approach to dealing with the above deficiencies is described in this paper, proposing an approach to actionable knowledge generation where the goal is to uncover properties of individuals for actions like population screening or targeting a marketing campaign. For such tasks, actionable rules are characterized by high coverage (support), as well as high sensitivity and specificity², even if this can be achieved only at a price of lower classification accuracy, which is a quality to be optimized in classification/prediction tasks.

3 Actionable Knowledge Generation through Subgroup Discovery

Subgroup discovery has the potential for inducing actionable knowledge to be used by a decision maker. The approach described in this paper is an approach to

² *Sensitivity* measures the fraction of positive cases that are classified as positive, whereas *specificity* measures the fraction of negative cases classified as negative. If TP denotes true positives, TN true negatives, FP false positives, FN false negatives, Pos all positives, and Neg all negatives, then $Sensitivity = TPr = \frac{TP}{TP+FN} = \frac{TP}{Pos}$, and $Specificity = \frac{TN}{TN+FP} = \frac{TN}{Neg}$, and $FalseAlarm = FPr = 1 - Specificity = \frac{FP}{TN+FP} = \frac{FP}{Neg}$. Quality measures in association rule learning are *support* and *confidence*: $Support = \frac{TP}{Pos+Neg}$ and $Confidence = \frac{TP}{TP+FP}$.

descriptive induction, but the underlying methodology uses elements and techniques from *predictive induction*. By basing the induction on labeled training instances, the induction process can be targeted to uncovering properties forming actionable knowledge. On the other hand, the standard assumptions like “induced rules should be as distinct as possible, covering different parts of the population” (which is the case in decision tree learning, as well as in rule learning using the covering algorithm) need to be relaxed; this enables the discovery of intersecting subgroups with high coverage/support, describing some population segments in a multiplicity of ways. This knowledge is redundant, if viewed purely from a classifier perspective, but extremely valuable in terms of its descriptive power, uncovering genuine properties of subpopulations from different viewpoints.

3.1 Algorithm *SD* for Subgroup Discovery

Algorithm SD is outlined in Figure 1. The algorithm is used in the Data Mining Server available on-line at <http://dms.irb.hr> and the reader can test it there. The algorithm assumes that the user selects one class as a *target class*, and learns subgroup descriptions of the form $TargetClass \leftarrow Cond$, where $Cond$ is a conjunction of features. The result is a set of best rules, induced using a heuristic beam search algorithm that allows for the induction of relatively general rules which may cover also some non-target class examples.

The aim of this heuristic rule learning algorithm is the search for rules with a maximal q value, where q is computed using the user-defined $TP/FP-tradeoff$ function. This function defines a tradeoff between true positives TP and false positives FP (see also Footnote 2). By searching for rules with high quality q , this algorithm tries to find rules that cover many examples of the target class and a low number of non-target examples. By changing a parameter of the tradeoff function the user can obtain rules of variable generality.

Typically, *Algorithm SD* can generate many rules of high quality q satisfying the requested condition of a minimal number of covered target class examples, defined by the *min_support* parameter. Accepting all these rules as actionable knowledge is generally not desired. A solution to this problem is to select a relatively small number of rules which are as diverse as possible. The algorithm implemented in the confirmation rule set concept [4] accepts as diverse those rules that cover diverse sets of target class examples. The approach cannot guarantee statistical independence of the selected rules, but ensures the diversity of generated models. Application of this algorithm is suggested for postprocessing of detected subgroups.

3.2 Rule Quality Measures for Subgroup Discovery

Various rule evaluation measures and heuristics have been studied for subgroup discovery [7,16], aimed at balancing the size of a group (referred to as factor g in [7]) with its distributional unusualness (referred to as factor p). The properties of functions that combine these two factors have been extensively studied

Algorithm SD: Subgroup Discovery

Input: $E = P \cup N$ (E training set, P positive (target class) examples,
 N negative (non-target class) examples)
 L set of all defined features (attribute values), $l \in L$
 $rule\_quality$ (user-defined $TP/FP - tradeoff$ function)
 $min\_support$ (minimal support for rule acceptance)
 $beam\_width$ (number of rules in the beam)

Output: $S = \{TargetClass \leftarrow Cond\}$
(set of rules formed of $beam\_width$ best conditions $Cond$)

- (1) **for** all rules in the beam ($i = 1$ to $beam\_width$) **do**
initialize condition part of the rule to be empty, $Cond(i) \leftarrow \{\}$
initialize rule quality, $q(i) \leftarrow 0$
- (2) **while** there are improvements in the beam **do**
- (3) **for** all rules in the beam ($i = 1$ to $beam\_width$) **do**
- (4) **for** all $l \in L$ **do**
- (5) form a new rule by forming a new condition as a conjunction of the
condition from the beam and feature l , $Cond(i) \leftarrow Cond(i) \wedge l$
- (6) compute rule quality q defined by the $TP/FP - tradeoff$ function
- (7) **if** $TP \geq min\_support$ **and** q is larger than any $q(i)$ in the beam **do**
- (8) replace the worst rule in the beam with the new rule and
reorder the rules with respect to their quality
- (9) **end for** features
- (10) **end for** rules from the beam
- (11) **end while**

Fig. 1. Heuristic beam search rule construction algorithm for subgroup discovery

(the “ p - g -space”, [7]). Similarly, the weighted relative accuracy heuristic, used in [15], trades off generality of the rule ($p(Cond)$, i.e., rule coverage) and relative accuracy ($p(Class|Cond) - p(Cond)$).

In contrast with the above measures, in which the generality of a rule is used in the generality/unusualness or generality/relative-accuracy tradeoff, the measure used in *Algorithm SD* is aimed to enable expert guided subgroup discovery in the TP/FP space, in which FP (plotted on the X -axis) needs to be minimized, and TP (plotted on the Y -axis) needs to be maximized. The TP/FP space is similar to the ROC (Receiver Operating Characteristic) space [11] in which a point in the ROC space shows classifier performance in terms of false alarm or *false positive rate* $FPr = \frac{FP}{TN+FP}$ (plotted on the X -axis) that needs to be minimized, and sensitivity or *true positive rate* $TPr = \frac{TP}{TP+FN}$ (plotted on the Y -axis) that needs to be maximized. In the ROC space, an appropriate tradeoff, determined by the expert, can be achieved by applying different algorithms, as well as by different parameter settings of a selected mining algorithm. The ROC space and the TP/FP space are equivalent if a single problem is being analysed: in the ROC space the results are evaluated based on the TPr/FPr

tradeoff, and in the TP/FP space based on the TP/FP tradeoff - the "rate" is just a normalising factor enabling us intra-domain comparisons.

It is well known from the ROC analysis, that in order to achieve the best results, the discovered rules should be as close as possible to the top-left corner of the ROC space. This means that in the TP_r/FP_r tradeoff, TP_r should be as large as possible, and FP_r as small as possible. Similarly, in the TP/FP space, TP should be as large as possible, and FP as small as possible.

For marketing problems, for instance, we have learned that intuitions like "how expensive is every FP prediction in terms of additional TP 's that should be covered by the rule" are useful for understanding the problem and directing the search. Suppose that some cost parameter c is defined that says: "For every additional FP , the rule should cover more than c additional TP examples in order to be better." Based on this reasoning, we can define a quality measure q_c , using the following TP/FP tradeoff: $q_c = TP - c * FP$. Quality measure q_c is easy to use because of the intuitive interpretation of parameter c . It also has a nice property for subgroup discovery: by changing the c value we can move in the TP/FP space and select the optimal point based on parameter c .

Consider a different quality measure q_g , using another TP/FP tradeoff: $q_g = TP/(FP + g)$. This quality measure is actually used in *Algorithm SD* for the evaluation of different rules in the TP/FP space, as well as for heuristic construction of interesting rules. Below we explain why this quality measure has been selected, and not some other more intuitive quality measure like the q_c measure defined above.

3.3 Analysis of the q_g Quality Measure

The selected quality measure q_g and generalization parameter g used in it, enable that by changing parameter g , different optimal points (rules) in the TP/FP space can be selected as the final solution. Although large g means that more general solutions can be expected, sometimes we would like to know in advance

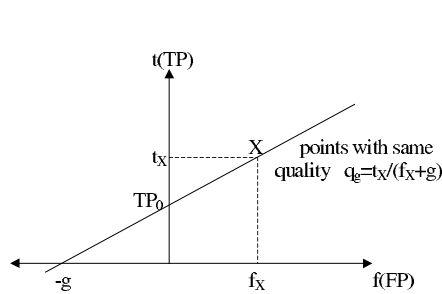


Fig. 2. Properties of quality q_g

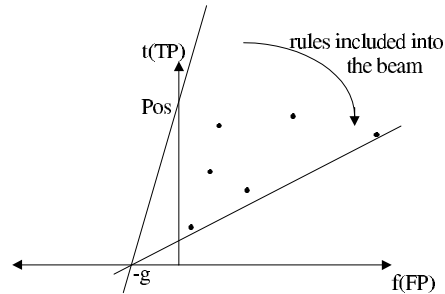


Fig. 3. Rules with highest quality included into the beam for $q_g = TP/(FP + g)$

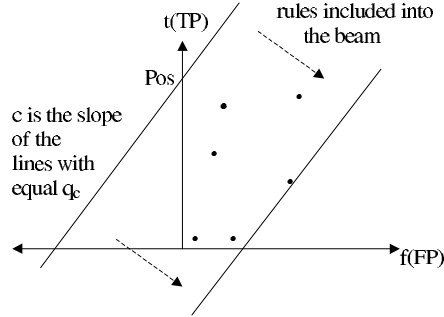


Fig. 4. Rules with highest quality included in the beam for $q_c = TP - c * FP$

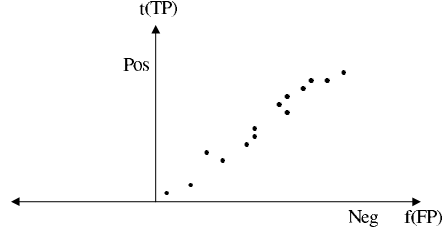


Fig. 5. Placement of interesting features in the TP/FP space after the first iteration

what properties of the selected rule can be expected for the selected g value, or, stated alternatively, by determining the desired properties of the rule under construction, what parameter value g should we select.

In *Algorithm SD*, increased generality (increasing g means moving to the right in the TP/FP space) results in more general subgroups discovered, covering more instances. If the value of g is low (1 or less) then covering of any non-target instance is made relatively very expensive and the final result are rules that cover only few target cases but also nearly no non-target class cases. This results in rules with high specificity (high confidence or low false alarm rate). If the value of parameter g is high (10 or higher) then covering of few non-target examples is not so expensive and more general rules can be generated. This approach is very appropriate for domains in which false positive predictions are not very expensive, like risk group detection in medical problems or detection of interesting customer groups in marketing, in which ‘pure’ rules would have too low coverage, making them unactionable.

If the algorithm employs exhaustive search (or if all points in the TP/FP space are known in advance) then there is no difference between the two measures q_g and q_c . Any of the two could be used for selecting the optimal point, only the values that must be selected for parameters g and c would be different. In this case, q_c might be even better because its interpretation is more intuitive.

However, since *Algorithm SD* is a heuristic beam search algorithm, the situation is different. Subgroup discovery is an iterative process, performing one or more iterations (typically 2–5) until good rules are constructed by forming conjunctions of features in the rule body. In this process, a rule quality measure is used for rule selection (for which the two measures q_g and q_c are equivalent) as well as for the selection of features and their conjunctions that have high potential for the construction of high quality rules in subsequent iterations; for this use, rule quality measure q_g is better than q_c . Let us explain why.

Suppose that we have a point (a rule) x in the TP/FP space, where t_x is its TP value and f_x its FP value, respectively. For a selected g value, q_g can be

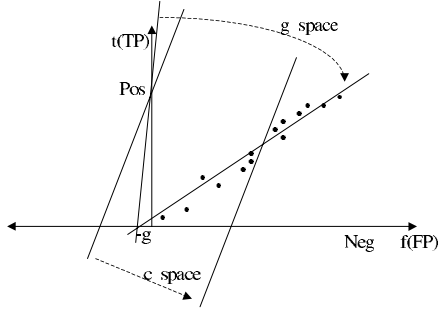


Fig. 6. The quality q_c employing the c parameter tends to select points with small TP values, while quality q_g employing the g parameter will include also many points with large TP values (from the right part of the TP/FP space) that have a chance to take part in building conjunctions of high quality rules

determined for this point x . It can be shown that all points that have the same quality q_g as the point (rule) x lie on a line defined by the following function:

$$t = \frac{t_x * f}{f_x + g} + \frac{t_x * g}{f_x + g} = \frac{t_x * (f + g)}{f_x + g}.$$

In this function, t represents the TP value of the rule with quality q_g which covers exactly $f = FP$ negative examples. By selecting different f 's, corresponding t 's can be determined by this function.

The line, determined by this function, crosses the $t(TP)$ line at point $t = t_x * g / (f_x + g)$ and the $f(FP)$ line at point $f = -g$. This is shown in Figure 2. The slope of this line is equal to the quality of point X , which equals $t_x / (f_x + g)$.

In the TP/FP space, points with higher quality than q_g are above this line, in the direction of the upper left corner. Notice that in the TP/FP space the top-left is the preferred part of the space: points in that part represent rules with the best TP/FP tradeoff. This reasoning indicates that points that will be included in the beam must all lie above the line of equal weights q_{beam} which is defined by the last point (rule) in the beam.

If represented graphically, first *beam_width* number of rules, found in the TP/FP space when rotating the line from point $(0, Pos)$ in the clockwise direction, will be included in the beam. The center of rotation is point $(-g, 0)$. This is illustrated in Figure 3. On the other hand, for the q_c quality measure defined by $q_c = TP - c * FP$ the situation is similar but not identical. Again points with same quality lie on a line, but its slope is constant and equal to c . Points with higher quality lie above the line in the direction of the left upper corner. The points that will be included into the beam are the first *beam_width* points in the TP/FP space found by a *parallel* movement of the line with slope c , starting from point $(0, Pos)$ in the direction towards the lower right corner. This is illustrated in Figure 4.

Let us now assume that we are looking for an optimal rule which is very specific. In this case, parameter c will have a high value while parameter g will have a very small value. The intention is to find the same optimal rule in the TP/FP space. At the first level of rule construction only single features are considered and most probably their quality as the final solution is rather poor.

See Figure 5 for a typical placement of potentially interesting features in the TP/FP space.

The primary function of these features is to be good building blocks so that by conjunctively adding other features, high quality rules can be constructed. By adding conjunctions, solutions generally move in the direction of the left lower corner. The reason is that conjunctions can reduce the number of FP predictions, but they reduce the number of TP 's as well. Consequently, by conjunctively adding features to rules that are already in the left lower corner, the algorithm will not be able to find their specializations nearer to the left upper corner. Only the rules that have high TP value, and are in the right part of the TP/FP space, have a chance to take part in the construction of interesting new rules.

Figure 6 illustrates the main difference between quality measures q_g and q_c : the former tends to select more general features from the right upper part of the TP/FP space (points in the so-called ' g space'), while the latter 'prefers' specific features from the left lower corner (points in the so-called ' c space'). In cases when c is very large and g is very small, the effect can be so important that it may prevent the algorithm from finding the optimal solution even with a large beam width. Notice, however, that *Algorithm SD* is heuristic in its nature and no statements are true for all cases. This means that in some, but very rare cases, the quality based on parameter c may result in a better final solution.

4 Experimental Evaluation

We have verified the claimed properties of the proposed rule quality measure in the medical Coronary Heart Disease (CHD) problem [5]. The task is the detection of subgroups which can be used as risk group models. The domain includes three levels of descriptors (basic level A with 10, level B with 16, and level C with 21 descriptors) and the results of subgroup discovery are five models (A1 and A2 for level A, B1 and B2 for level B, and C1 for level C), presented in [5]. Algorithm SD with the q_g measure was used for subgroup detection, with the goal of detecting different, potentially relevant subgroups. The algorithm was used iteratively many times with different g values. In each iteration few best solutions from the beam were shown to the domain expert. The selection of subgroups which will be used as model descriptions was based on the expert knowledge. The position of the expert selected subgroups in the TP/FP space is presented in Figures 7–9. It can be noticed that the selected subgroups do not lie on the ROC curves: this means that expert-selected actionability properties of subgroups were more important than the optimization of their TP/FP tradeoff.

For the purpose of comparing the q_g and q_c measures we have constructed one ROC curve for each of the two measures. The procedure was repeated for all levels A–C. The ROC curve for the q_g measure was constructed so that for g values between 1 and 100 the best subgroups lying on the convex hull in the TP/FP space were selected: this results in the thick lines in Figures 7–9. The thin lines represent ROC curves obtained for subgroups induced by the q_c measure for c values between 0.1 and 50.

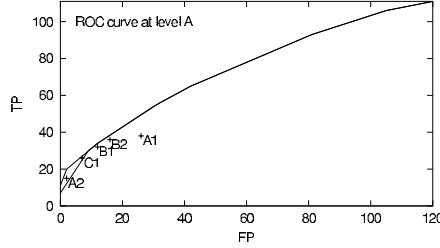


Fig. 7. TP/FP space presenting the ROC curves of subgroups induced using quality measures $q_g = TP/(FP + g)$ (thick line) and $q_c = TP - c * FP$ (thin line) at data level A. Labels A1–C1 denote positions of subgroups selected by the medical expert as interesting risk group descriptions [8,5]

Figure 7 for level A demonstrates that both curves agree in the largest part of the TP/FP space, but that for small FP values the q_g measure is able to find subgroups covering more positive examples. According to the analysis in the previous section, this was the expected result. In order to make the difference more obvious, for levels B and C, only the left part of the TP/FP space is shown in Figures 8 and 9. Similar curve properties can be noticed for different data sets.

The differences between the ROC curves for q_g and q_c measures may seem small and insignificant, but in reality it is not so. The majority of interesting subgroups (this claim is supported also by models A1–C1 selected by the domain expert) are subgroups with a small false positive rate which lie in the range in which q_g works better. In addition, for subgroups with $FP = 0$ the true positive rate in our examples was about two times larger for subgroups induced with q_g than with q_c . Furthermore, note that for levels A and B there are two out of five subgroups (A2 and C1) which lie in the gap between the ROC curves. If the q_c measure instead of q_g measure were used in the experiments described in [5], at least subgroup A2 could not have been detected.

5 Conclusions and Lessons Learned

This work describes actionable knowledge generation in the descriptive induction framework, pointing out the importance of effective expert-guided subgroup discovery in the TP/FP space. Its main advantages are the possibility to induce knowledge with different generalization levels (achieved by tuning the g parameter of the subgroup discovery algorithm) and the measure that ensures high quality rules also in the heuristic environment. In addition, the paper argues that expert’s involvement in the induction process is substantial for successful actionable knowledge generation.

The presented methodology has been applied to different medical and marketing domains. In the medical problem of detecting and describing of Coronary Heart Disease risk groups we have learned a few important lessons. The main is that in this type of problem, there are no predefined specificity or sensitivity levels to be satisfied. The actionability of induced models, based on the detected subgroups, largely depends on the applied subgroup discovery method, but also on (a) whether the attributes used in the induced model can be easily and reliably measured, and (b) how interesting/unexpected are the subgroup descriptions in

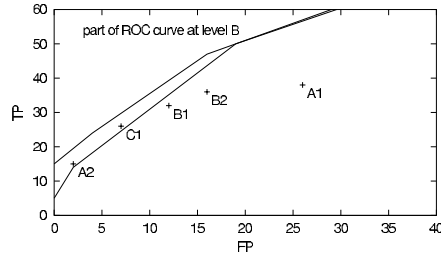


Fig. 8. The left part of the ROC curves representing subgroups induced at data level B

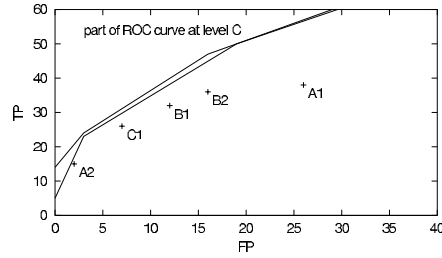


Fig. 9. The left part of the ROC curves representing subgroups induced at data level C

the given population. Evaluation of such properties is completely based on expert knowledge and the success of the search depends on expert involvement. The aim of machine learning based subgroup detection described in this work is thus to enable the domain expert to effectively search the hypothesis space, ranging from very specific to very general models.

In the marketing problems where the task is to find significant characteristics of customer subgroups who do not know a brand compared to the characteristics of the population that recognizes the brand, the main lesson learned is that the ROC space is very appropriate for the comparison of induced models. Only subgroups lying on the convex hull may be optimal solutions and all other subgroups can be immediately discarded. When concrete parameters of the mailing campaign are known, like marginal cost per mailing and the size of the population, they define the slope of the lines with equal profit in the ROC space. Movements in the ROC space along these lines will not change the amount of total profit while movements upward or downward will increase or decrease the profit, respectively. The optimal subgroup in a concrete marketing situation is the point on the convex hull which has an equal profit line as its tangent. In terms of actionability, however, the appropriate parameters for subgroup discovery need to be determined in data preprocessing.

Acknowledgements

This work was supported by the the Croatian Ministry of Science and Technology, Slovenian Ministry of Education, Science and Sport, and the EU funded project Data Mining and Decision Support for Business Competitiveness: A European Virtual Enterprise (IST-1999-11495). We are grateful to Miro Kline, Bojan Cestnik, and Peter Flach for their collaboration in marketing domains, and to Goran Krstajić for his collaboration in the experiments in coronary heart disease risk group detection.

References

1. Agrawal, R., Mannila, H., Srikant, R., Toivonen, H., & Verkamo, A. I. (1996) Fast discovery of association rules. In U. M. Fayyad, G. Piatetski-Shapiro, P. Smyth and R. Uthurusamy (Eds.) *Advances in Knowledge Discovery and Data Mining*, pp. 307–328. AAAI Press. 163
2. Clark, P. & Niblett, T. (1989). The CN2 induction algorithm. *Machine Learning*, 3(4):261–283. 163, 165
3. Flach, P. & Gamberger, D. (2001) Subgroup evaluation and decision support for direct mailing marketing problem. *Integrating Aspects of Data Mining, Decision Support and Meta-Learning Workshop* at ECML/PKDD 2001 Conference. 164
4. Gamberger, D. & Lavrač, N. (2000) Confirmation rule sets. In *Proc. of 4th European Conference on Principles of Data Mining and Knowledge Discovery (PKDD2000)*, pp.34–43, Springer. 166
5. Gamberger, D. & Lavrač, N. (2002) Descriptive induction through subgroup discovery: a case study in a medical domain. In *Proc. of 19th International Conference on Machine Learning (ICML2002)*, Morgan Kaufmann, in press. 164, 171, 172
6. Kagan, T. & Ghosh, J. (1996) Error correlation and error reduction in ensemble classifiers. *Connection Science*, 8, 385–404.
7. Klösgen, W. (1996) Explora: A multipattern and multistrategy discovery assistant. In U. M. Fayyad, G. Piatetski-Shapiro, P. Smyth and R. Uthurusamy (Eds.) *Advances in Knowledge Discovery and Data Mining*, pp. 249–271. MIT Press. 166, 167
8. Krstačić, G., Gamberger, D., & Šmuc, T. (2001) Coronary heart disease patient models based on inductive machine learning. In *Proc. of 8th Conference on Artificial Intelligence in Medicine in Europe (AIME 2001)*, pp.113–116. 172
9. Lavrač, N., Gamberger, D., & Flach, P. (2002) Subgroup discovery for actionable knowledge generation: Defences of classification rule learning and lessons learned. *Data Mining Lessons Learned Workshop* at ICML 2002 Conference, to be printed. 165
10. Michalski, R. S., Mozetič, I., Hong, J., & Lavrač, N. (1986) The multi-purpose incremental learning system AQ15 and its testing application on three medical domains. In *Proc. Fifth National Conference on Artificial Intelligence*, pp. 1041–1045, Morgan Kaufmann. 163, 165
11. Provost, F. & Fawcett, T. (2001) Robust classification for imprecise environments. *Machine Learning*, 42(3), 203–231. 167
12. Rivest, R. L. & Sloan, R. (1988) Learning complicated concepts reliably and usefully. In *Proc. Workshop on Computational Learning Theory*, 69–79, Morgan Kaufman.
13. Piatetsky-Shapiro, G. & Matheus, C. J. (1994) The interestingness of deviation. In *Proc. of the AAAI-94 Workshop on Knowledge Discovery in Databases*, pp. 25–36. 164
14. Silberschatz, A. & Tuzhilin, A. (1995) On Subjective Measure of Interestingness in Knowledge Discovery. In *Proc. First International Conference on Knowledge Discovery and Data Mining (KDD)*, 275–281. 163, 164, 165
15. Todorovski, L., Flach, P., & Lavrač, N. (2000) Predictive Performance of Weighted Relative Accuracy. In Zighed, D. A., Komorowski, J. and Zytkow, J., editors, *Proc. of the 4th European Conference on Principles of Data Mining and Knowledge Discovery (PKDD2000)*, Springer-Verlag, 255–264. 167

16. Wrobel, S. (1997) An algorithm for multi-relational discovery of subgroups. In *Proc. First European Symposium on Principles of Data Mining and Knowledge Discovery*, pp.78–87, Springer. 163, 166

Clustering Transactional Data

Fosca Giannotti¹, Cristian Gozzi¹, and Giuseppe Manco²

¹ CNUCE-CNR, Pisa Research Area
Via Alfieri 1, 56010 Ghezzano (PI), Italy
F.Giannotti@cnuce.cnr.it
C.Gozzi@guest.cnuce.cnr.it

² ISI-CNR
Via Bucci 41c, 87036 Rende (CS), Italy
manco@isi.cs.cnr.it

Abstract. In this paper we present a partitioning method capable to manage transactions, namely tuples of variable size of categorical data. We adapt the standard definition of mathematical distance used in the K -Means algorithm to represent dissimilarity among transactions, and re-define the notion of cluster centroid. The cluster centroid is used as the representative of the common properties of cluster elements. We show that using our concept of cluster centroid together with Jaccard distance we obtain results that are comparable in quality with the most used transactional clustering approaches, but substantially improve their efficiency.

1 Introduction

The need to develop algorithms for clustering transactional data, i.e., tuples of variable size of categorical data, comes from many relevant applications, such as web data analysis. Log data typically contain collections of single accesses of web users to web resources, and the set of one user's accesses can be further collected into *sessions*. In such context there are some relevant difficulties that make the traditional approaches unsuitable, such as the enormous amount of transactions or distinct elements in a single transaction, that can be huge, and the inherent difficulty of classical algorithms to “semantically” cluster discrete-valued data. As an example, an important semantic feature in this context is the notion of cluster representative: namely, a transaction subsuming the characteristics of the transactions belonging to the cluster.

Transactional clustering is related to two classes of problems: clustering of categorical attributes and clustering of variable-length sets. An example of algorithm that deals with such problems in a unified way was introduced by Guha et al. [4]. Here, the ROCK algorithm is presented, an agglomerative hierarchical method for clustering sets of categorical values. Hierarchical methods are often presented in the literature as the best quality clustering approaches, but they are limited because of their quadratic complexity over the number of object under consideration. Moreover, it is difficult to generate a representative providing

an easy interpretation of the clusters population. An alternative to Hierarchical methods is that of exploiting partitioning methods such as, e.g., K -Means and its variants, which have a linear scalability on the size of the dataset. However, these methods do not supply an adequate formalism for tuples of variable size containing categorical data.

The main idea of our approach is that of defining a new notion of cluster centroid, that represents the common properties of cluster elements. Similarity inside a cluster is hence measured by using the cluster representative, that becomes also a natural tool for finding an explanation of the cluster population. Our definition of cluster centroid is based on a data representation model which simplifies the one used in document clustering. In fact, we use compact representation of boolean vectors that states only presence and absence of items, while document clustering requires to store the frequencies of items (words).

In this paper we show that using our concept of cluster centroid associated with jaccard distance we obtain results having a quality comparable with other approaches used in this task, but we have better performances in term of execution time. Moreover, cluster representatives provide an immediate explanation of clusters features. A particular remark is due to the performance of our approach, which is relevant in case of real-time applications, e.g., clustering of search engine query results or web access sessions for personalization. In these cases, response time is a fundamental requirement of clustering algorithms, and performances of traditional methods is unacceptable.

The plan of the paper is as follows. Section 2 provides the formal foundations of the clustering algorithm that is shown in section 3. Finally, in subsection 3.1 we show formal and empirical results to prove that the algorithm is both efficient and effective on transactional data and in subsection 3.2 we briefly describe a real application on web log data.

2 Problem Statement

The most well-known partitioning approaches to clustering data are the centroid-based methods, such as the K -Means algorithm [2]. In such approaches, each object x_i is assigned to a cluster j according to its distance $d(x_i, m_j)$ from a value m_j representing the cluster itself. m_j is called the *centroid* (or *representative*) of the cluster.

Definition 1. *Given a set of objects $D = \{x_1, \dots, x_n\}$, a centroid-based clustering problem is to find a partition $\mathcal{C} = \{C_1 \dots C_k\}$, of D such that:*

1. *each C_i is associated to a centroid m_i*
2. *$x_i \in C_j$ if $d(x_i, m_j) \leq d(x_i, m_l)$ for $1 \leq l \leq k, j \neq l$*
3. *The partition $\mathcal{C}$ minimizes $\sum_{i=1}^k \sum_{x_j \in C_i} d^2(x_j, m_i)$*

□

The K -Means algorithm works as follows. First of all, K objects are randomly selected from D . Such objects correspond to some initial cluster centroids, and

each remaining object in D is assigned to the cluster satisfying condition 2. Next, the algorithm iteratively recomputes the centroid of each cluster and re-assigns each object to the cluster of the nearest centroid. The algorithm terminates when the centroids do not change anymore. In that case, in fact, condition 3 holds.

The general schema of K -Means is parametric to the functions d and rep , that formalize respectively the concepts of distance measure and cluster centroid. Such concepts in turn are parametric to the domain of D .

Definition 2. *Given a domain $\mathcal{U}$ equipped with a distance function $d : \mathcal{U} \times \mathcal{U} \mapsto \mathbb{R}$ and a set $\mathcal{S} = \{x_1, \dots, x_m\} \subseteq \mathcal{U}$, the centroid of $\mathcal{S}$ is the element that minimizes the sum of the squared distances:*

$$rep(\mathcal{S}) = \min_{v \in \mathcal{U}} \sum_{i=1}^m d^2(x_i, v)$$

□

Transactional data, in this paper, are referred to as vectors of variable size, containing categorical values.

Definition 3. *Given a set $\mathcal{I} = \{a_1 \dots a_m\}$, where a_i is a categorical value (henceforth called item), The domain of transactional data is defined as $\mathcal{U} = \text{Powerset}(\mathcal{I})$. A subset $I \subseteq \mathcal{I}$ is called an itemset.* □

Example 1. Let us suppose that a_i represents a web URL, and that $\mathcal{I}$ models all the pages contained within a web server. An itemset can then represent a typical web user session within the web server, i.e., the set of pages a user has accessed within the given web server. ◁

2.1 Dissimilarity Measure

As we already mentioned, the standard approach used to deal with transactional data in clustering algorithms is that of representing such data by means of fixed-length boolean attributes.

Example 2. Let us suppose $\mathcal{I} = \{a_1, \dots, a_{10}\}$. We can represent the itemsets $I_1 = \{a_1, a_2\}$, $I_2 = \{a_1, a_2, a_3, a_5, a_6, a_7\}$ and $I_3 = \{a_4\}$ by means of the following vectors:

$$\begin{array}{ll} I_1 & [1, 1, 0, 0, 0, 0, 0, 0, 0, 0] \\ I_2 & [1, 1, 1, 0, 1, 1, 1, 0, 0, 0] \\ I_3 & [0, 0, 0, 1, 0, 0, 0, 0, 0, 0] \end{array}$$

In this representation, position i of the boolean vector represents object a_i . A value 1 corresponds to the presence of a_i to the transaction, while a value 0 corresponds to its absence. ◁

In principle, the above representation allows to directly apply commonly used distance definitions, such as Minkowsky or Mismatch count, and related optimal cluster centroids. Such approaches, however, do not capture our intuitive idea of transaction similarity. In the above example, I_2 is more similar to I_1 than to I_3 , since there is a partial match between I_1 and I_2 and, by the converse, I_2 and I_3 are disjoint. Now, the above dissimilarity measures consider both the presence and the absence of an item within a transaction. As a consequence, sparse transactions (i.e., transactions containing a very small subset of $\mathcal{I}$) are very likely to be similar, even if the items they contain are quite different.

This problem has many common aspects with the problem of clustering documents. There, a document is coded as a term-vector, which contains the frequency of each significant term in the document. In this context, ad-hoc measures for these kind of data are used: *Jaccard Coefficient* [10] (henceforth called s_J), that computes the number of elements in common of two documents, and *Cosine similarity* [10] (henceforth called s_C), that measures the degree of orthogonality of two document vectors.

A distance measure can be straightforwardly defined from these measures. For example, we can define $d(x, y) = 1 - s(x, y)$.

In the simplified hypothesis that term-vectors do not contain frequencies, but behave simply as boolean vectors (like in the case of web user sessions), a more intuitive but equivalent way of defining the Jaccard distance function can be provided. Given two itemsets I and J , we can represent $d(I, J)$ as the (normalized) difference between the cardinality of their union and the cardinality of their intersection:

$$d_J(I, J) = 1 - \frac{|I \cap J|}{|I \cup J|}$$

this measure capture our idea of similarity between objects, that is directly proportional to the number of common values, and inversely proportional to the number of different values for the same attribute.

2.2 Cluster Representative

In definition 2 we have specified the criteria for obtaining the cluster representative once the distance function is fixed. Intuitively, a cluster representative for transactional data should model the content of a cluster, in terms, e.g., of the elements that are most likely to appear in a transaction belonging to the cluster.

A problem with the traditional distance measures is that the computation of a cluster representative is computationally expensive. As a consequence, most approaches [1,9] approximate the cluster representative with the euclidean representative. However, such approaches suffer of some drawbacks:

- Huge cluster representatives cause poor performances, mainly because as soon as the clusters are populated, the cluster representatives are likely to become extremely huge.

- In transactional domains, the cluster representative does not represent a transaction.

On the other side, the use of the Jaccard distance for boolean vectors makes computing a cluster representative problematic as well. In fact, the problem of minimizing the expression $\sum_i d_J^2(x_i, c)$ given a set $\{x_1, \dots, x_m\}$ cannot be solved in polynomial time if c is required to be a boolean vector [3]. In addition, an optimal cluster centroid is not necessarily unique.

In order to overcome such problems, we can compute an approximation that resembles the cluster representatives associated to the euclidean and mismatch-count distances. Union and intersection seem good candidates to start with.

Lemma 1. *Given a set $\mathcal{S} = \{x_1, \dots, x_m\}$, $rep(\mathcal{S}) \subseteq \bigcup_i x_i$* □

The above lemma states that the elements we need to consider in order to compute the representative are those contained in the union of the elements of the cluster. However, the cluster centroid can be different from the union. The idea of approximating the cluster centroid with the union has some drawbacks that make such a choice unpractical. First of all, the resulting representative can have a huge cardinality because of the heterogeneity of objects. The second problem is represented by the fact that the union contains all the values in the objects without considering their frequencies. In this case, in fact, the resulting intra-cluster similarity can be misleading, since the cluster may contain elements with little elements in common.

To overcome the above problems, one can think to use the intersection of the transactions as an approximation of the representative. And, indeed, the intersection is actually contained in a cluster representative.

Lemma 2. *Given a set $\mathcal{S} = \{x_1, \dots, x_m\}$, $\bigcap_i x_i \subseteq rep(\mathcal{S})$* □

Also in this case, the intersection does not necessarily correspond to the cluster representative. Again, it can be unpractical to approximate the cluster representative with the intersection. With densely populated clusters, it is very likely to obtain an empty intersection. On the other side, the cluster representative cannot be empty, as the following result shows.

Lemma 3. *Given a set $\mathcal{S} = \{x_1 \dots x_m\}$, $rep(\mathcal{S})$ is not empty.* □

A property of cluster representatives is that frequent items are very likely to belong to them. Such a property is not true in general: in cases where the most frequent items are contained in huge, non-homogeneous transactions, and there are also small transactions, the elements contained in small transactions are more likely to appear in the representative than the most frequent items. However, such a property is unlikely to hold in homogeneous groups of transactions, like in the case of transactions grouped according to Jaccard similarity. Hence, we can provide a greedy heuristic that, starting from the intersection, iteratively refines the approximation of the cluster representative by iteratively adding the most frequent items until the sum of the distances can be minimized. Figure 1

Algorithm $rep_H(\mathcal{S})$;

Input : A set of transactions $\mathcal{S} = \{\mathbf{x}_1, \dots, \mathbf{x}_m\}$.

Output : A transaction $\mathbf{m}$ that minimizes $f(\mathbf{y}) = \sum_{\mathbf{x} \in \mathcal{S}} d^2(\mathbf{x}, \mathbf{y})$
Method :

- sort $\bigcup \mathcal{S}$ by increasing frequency, obtaining the list $a_1, \dots, a_m$ such that $freq(a_i, \mathcal{S}) > freq(a_{i+1}, \mathcal{S})$;
 - Let initially $\mathbf{m} = \bigcap_{\mathbf{x} \in \mathcal{S}} \mathbf{x}$;
 - while $f(\mathbf{m})$ decreases
 - add a_h to $\mathbf{m}$, for increasing values of h ;
-

Fig. 1. Greedy representative computation

shows the main schema of greedy procedure for representative computation. In principle, the procedure can produce a local minimum that does not necessarily correspond to the actual minimum. However, experimental results have shown that in most cases the heuristic computes the actual representative, and in each case it provides a suitable, compact approximation of the cluster representative.

Computing such an approximation can still be expensive, since we need to sort the items on the basis of their frequencies. We can define a further approximation which avoids such computation, by introducing a user-defined threshold value γ over the frequency of the items appearing in any transaction of the cluster. Such a parameter should intuitively represent the degree of intra-cluster similarity desired by the user, and corresponds to the minimum percentage of occurrences an item must have to be inserted into the approximation of the cluster representative.

Definition 4. Let $\mathcal{S} = \{x_1, \dots, x_m\}$ be a set of transactions, and $\gamma \in [0, 1]$. The representative of $\mathcal{S}$ is defined as

$$rep_\gamma(\mathcal{S}) = \left\{ v \in \bigcup_i x_i \mid freq(v, \mathcal{S})/m \geq \gamma \right\}$$

where $freq(v, \mathcal{S}) = |\{x_i \mid v \in x_i\}|$. □

The approaches $rep_H(\mathcal{S})$ and $rep_\gamma(\mathcal{S})$ represent two viable alternatives: rep_γ represents an approximation that is, as we shall see, extremely efficient to compute. However, it is influenced by the value of γ . Greater γ values correspond to a stronger intra-cluster similarity, less populated clusters and low-cardinality representatives. By the converse, lower γ values correspond to a weaker intra-cluster similarity, huge clusters and high-cardinality representatives. On the other side, rep_H allows us to avoid specifying the threshold, at the cost of a less efficient computation.

2.3 Unclustered Objects

From the definition of d_J , the elements with empty intersection have distance 1. Using our distance this means that objects having an empty intersection with each cluster representative are not inserted in any cluster. We refer to these unclustered objects as *trash*. This problem is mainly due to the fact that, within the domain $\mathcal{U}$ equipped with d_J several objects can be equally distant from the other clusters. Consequently, any assignment of such objects to a cluster is not significant. Using the mean vector as cluster centroid and jaccard or cosine distances there are not unclustered objects, because it is impossible to find objects having a null vector dot product with all cluster means. This is not always good, because unclustered objects can have a distance from the centroid very close to 1 and for this reason they must be considered outliers. Experimental results show that the cardinality of the trash is related to the clustering parameters (K and γ) and to the structure of dataset (Number of input objects, average cardinality of sets, number of distinct values). To overcome the problem of large trashes, various solutions can be adopted that act over these parameters.

- The initial cluster centroids can be carefully chosen, in order to limit the size of the trash.
- We can iteratively reduce the value of the γ value, or augment the number of clusters.
- We can further apply the clustering technique to the trash, by choosing a random set of cluster representatives among the trash, and grouping the remaining elements re-iterating the algorithm. In such a way, we can avoid the effects of the high dissimilarity between the trash and the clusters previously generated.

3 Transactional K -Means

The main schema of the algorithm is shown in fig. 2. The algorithm has two main phases. In the first phase, it computes $k + 1$ clusters. The tuples are assigned to the first k clusters according to the distance measure d_J . $(k + 1)$ -th cluster (the trash cluster) is created to contain objects that are not assigned to any of the first k clusters.

The second phase has the main objective to manage the trash cluster. The main idea in this phase is to try to recursively split the trash cluster into l further clusters. Of course, the final result may contain clusters with a single element: elements substantially different can remain in the trash cluster until they are not chosen as cluster centroids.

3.1 Experimental Results

The objective of the following experiments is to study the behavior of the algorithm. To this purpose,

Algorithm TrK-Means(D, k, γ);

Input : a dataset $D = \{x_1, \dots, x_N\}$ of transactions, the desired number k of clusters.

A cluster representative threshold value γ .

Output : a partition $\mathcal{C} = \{C_1, \dots, C_{k+l}\}$ of D in $k + l$ clusters, where $l \geq 0$.

Method :

- Randomly choose $x_{i_1}, \dots, x_{i_k}$ and set $m_j = x_{i_j}$ for $1 \leq j \leq k$.
 - Repeat
 - for each j , set $C_j = \{x_i | d_J(x_i, m_j) < d_J(x_i, m_l), 1 \leq l \leq k\}$;
 - set $C_{k+1} = \{x_i | \text{for each } j \ d_J(x_i, m_j) = 1\}$;
 - set $m_j = \text{rep}(C_j)$ for $1 \leq j \leq k$;
 until m_j do not change;
 - recursively apply the algorithm to C_{k+1} , producing a partition of C_{k+1} in l clusters.
-

Fig. 2. The Transactional K-Means Algorithm

- We first compare the performance and scalability of the two version of our algorithm (the one which uses the greedy procedure and the one which uses the γ threshold) with the performance of the traditional approaches (i.e., the approaches that adopt the mean vector as the cluster centroid and jaccard and cosine distance measures).
- Next, we compare the quality of results of the different approaches with real and synthetic datasets using some predefined quality measures. In this context, we study how the γ threshold and the number k of desired clusters influence the trash population.

In our experiments, we use both real and synthetic data. Synthetic data generation is tuned, among the others, according to [8, section 2.4.3] the number of transactions ($|\mathcal{D}|$), the average size of the transactions ($|T|$), and the number of different items (N).

We compared the performance of the approaches shown in this paper with the ROCK algorithm and two versions of the K -Means algorithm that use the mean vector as a representative, and respectively the Jaccard and Cosine similarity measures. Figure 3 contains two graphics that show respectively the comparison of total execution times of the various approaches, and the average iteration time of γ -based K -Means algorithm w.r.t. the number of transactions. Figure 4 shows the scalability of K -Means based approaches w.r.t. the number of distinct items and the number of desired clusters. As expected, the overhead due to the mean vector centroid is significantly large. For datasets of little and medium size, ROCK is more efficient than K -Means with mean vector representative, but its execution time increases rapidly due to its quadratic complexity. Despite its high formal complexity, the greedy-based algorithm has better performances than the mean vector algorithms because of the lower cardinality of representatives, and than ROCK. It is worth noticing from fig. 3 and fig. 4 that K -Means based

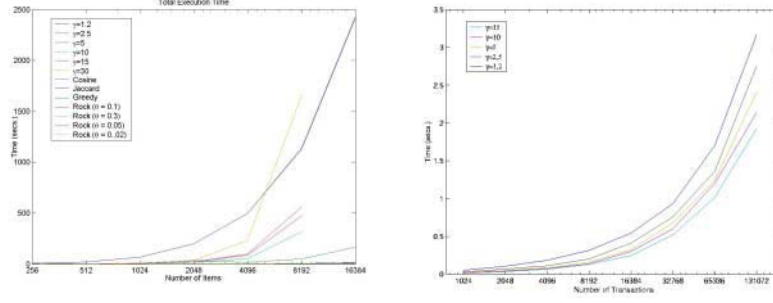


Fig. 3. Transactional clustering scalability varying N , $T = 30$, $D = 5K$

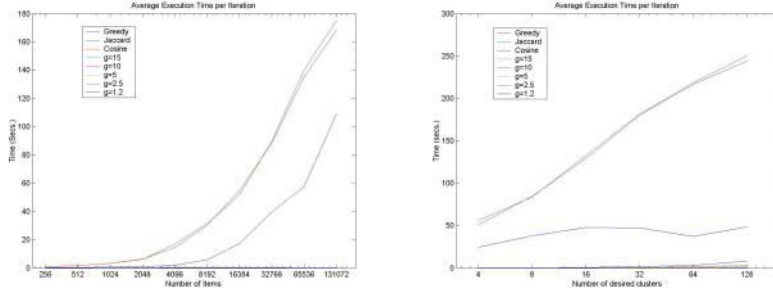


Fig. 4. Transactional clustering scalability varying D and K

algorithms scale, as expected, in linear time w.r.t. K , $|D|$ and $|N|$. However, the execution time of γ -based approach is inversely proportional to the value of the γ threshold: lower γ values correspond to larger representatives and consequently to an increasing cost for computing the cluster assignment.

To analyze quality of results we adopt two main approaches. The first approach is based on the consideration that the aim of a clustering algorithm is that of minimizing the intra-cluster dissimilarity [7]. The second approach, only suitable for synthetized data, adopts a further quality measure, namely the $\mathcal{F}$ -measure [5].

According to the definition of clustering given in this paper, a natural quality measure associated to an instance of the k -Means approach that produces a partition $P = \{C_1, \dots, C_{k+1}\}$ of the dataset D , where C_{k+1} is an additional cluster containing unclustered objects, is the following:

$$\mathcal{Q}_d = Avg_{i=\{1\dots k\}} \frac{Avg_{\mathbf{x}, \mathbf{y} \in C_i} d(\mathbf{x}, \mathbf{y})}{Avg_{\mathbf{x}, \mathbf{y} \in D} d(\mathbf{x}, \mathbf{y})}$$

This quality measure computes the average of the average percentages of intra cluster distances respect to total distance between transactions. One drawback of the above measure is that large quantities of unclustered objects cause a better quality values: high γ values can cause a large trash quantity, but they cause also an higher intra-cluster similarity. Thereby, in order to evaluate the quality of $\mathcal{Q}_d$ measure one should consider also the number of unclustered objects.

When using synthesized data, we can adopt as a further quality measure the $\mathcal{F}$ -measure [5]. We provide a class label for each transaction (where the number of different class labels, C , is a further parameter to be specified in the data generation process). A transaction $t = \{a_1, \dots, a_n\}$ is assigned to a class by, first, randomly assigning a class frequency to each item belonging to the transaction, and, next, choosing the class c ($1 \leq c \leq C$) that maximizes the formula

$$L(c) = \prod_{i=1}^n (1 + freq(a_i|c))$$

The $\mathcal{F}$ -measure of a cluster i w.r.t. a class j is given by evaluating the *relevance* of the cluster:

$$\mathcal{F}(i, j) = \frac{2 \times p(i, j) \times r(i, j)}{p(i, j) + r(i, j)}$$

where $p(i, j)$ and $r(i, j)$ represent respectively *precision* and *recall* of i w.r.t. j . The total $\mathcal{F}$ -measure is given by the weighted average of all values of the maximal $\mathcal{F}$ -measure for every class. Higher values of the $\mathcal{F}$ measure represent higher quality clusters.

We compare the results quality of greedy-based and threshold-based algorithm with the ROCK algorithm and the other K -Means methods that adopt the mean vector with Jaccard and Cosine distance. We use three datasets as a testbench: the first two are synthetic datasets, and differ mainly in the number of distinct items $|D|$ and in the average transaction length $|T|$. As soon as $|T|$ and $|D|$ increase, transactions are less likely to be similar, and consequently the trash size is likely to increase as well. This phenomenon is even more evident in the third dataset, representing web user sessions, that tend to be extremely heterogeneous. Table 1 resumes the quality results of the approaches. As we can see, the approaches show comparable results, even though threshold-based K -Means algorithm and ROCK are more sensitive to threshold values.

3.2 An Application: Web Sessions Clustering

In this section we describe a sample application of the algorithm: clustering of web sessions. The main objective of the experiment is to study the behavior of a typical (anonymous) web user coming from a given community. To this purpose, the idea of using the web logs of a proxy server of a given community gives us sufficient information about the browsing patterns of the the users belonging to that community.

We made our experiments with the data available from the web logs coming from the proxy server of the University of Pisa. A web log is a file in which each

Table 1. Quality measures: Synth1=T100.D10k.N1k.C8, Synth2=T30.D1k.N1k.C8

Synth 1	<i>Greedy</i>	<i>Jaccard</i>	<i>Cosine</i>	$\gamma=10$	$\gamma=5$	$\gamma=2.5$	<i>rock0.05</i>	<i>rock0.03</i>	<i>rock0.01</i>
$\mathcal{F}$	0.62439	0.70300	0.70561	0.72288	0.70402	0.69156	0.52927	0.83265	0.7384
Q_d	0.92195	0.93612	0.93612	0.93472	0.93715	0.94389	0.9283	0.93961	0.94036
Trash	143	0	0	0	0	0	0	0	0

Synth 2	<i>Greedy</i>	<i>Jaccard</i>	<i>Cosine</i>	$\gamma=10$	$\gamma=5$	$\gamma=2.5$	<i>rock0.05</i>	<i>rock0.03</i>	<i>rock0.01</i>
$\mathcal{F}$	0.68695	0.69645	0.64357	0.70808	0.66988	0.56565	0.68926	0.68232	0.45594
Q_d	0.91893	0.91724	0.91726	0.91565	0.92732	0.93996	0.90508	0.90366	0.93002
Trash	87	0	0	0	0	0	0	0	0

Web data	<i>Greedy</i>	<i>Jaccard</i>	<i>Cosine</i>	$\gamma=10$	$\gamma=5$	$\gamma=2.5$	<i>rock0.05</i>	<i>rock0.03</i>	<i>rock0.01</i>
Q_d	0.78195	0.9489	0.9446	0.90361	0.9072	0.93411	0.9151	0.93012	0.95471
Trash	3717	0	0	2350	1981	1667	0	0	0

line contains a description of the access to a given web resource from a given user. A typical log row contains information about the IP of the user requesting the resource, the address of the resource requested, the size and status of the request and the time the request was made. We considered logs covering two weeks of browsing activity of the users of the University of Pisa.

In the preprocessing phase, we grouped the user accesses by client. Each web session corresponds to the sequence of pages that are requested by a given user in a reasonably low time interval [6]. In our experiment, the dataset contained 5961 sessions with average cardinality of 26 web accesses. For this dataset we obtained good quality clusters with acceptable trash quantity. Figure 5 shows an example of the results obtained using the algorithm at site level with $\gamma = 0.05$ and $K = 32$. For this experiment, the initial cardinality of the trash cluster is 1335, and can be reduced to smaller values by activating the second phase of the algorithm. The resulting cluster representatives contain significant sites concerning related arguments. This peculiarity makes the cluster interpretation extremely simple. For instance, we can observe that cluster 1 groups sessions concerning linux resources, cluster 2 groups sessions concerning libraries and online document retrieval services, and cluster 3 groups sessions concerning on-line news. On a total of 32 clusters, 20 cluster are directly understandable with a simple manual scan of the representative.

4 Conclusion and Future Work

The great advantage of the K -Means algorithm in data mining applications is its linear scalability. This makes it particularly suitable to deal with large datasets. However the approaches proposed to extend the use of K -Means to categorical attributes are suitable only for attributes with small domains, because of the large data structure needed to represent the inputs. The algorithm we have proposed overcomes such limitations, using a similarity measure capable to deal with sets of categorical objects and using efficiently computable (frequency-based) concepts of cluster representatives.

www.linux-mandrake.com	www.citeseer.com	www.mondadori.com
wxstudio.linuxbox.com	www.informatik.uni-trier.de	www.repubblica.it
Sunsite.cnlab-switch.ch	search.yahoo.com	www.gazzetta.it
casema.linux.tucows.com	computer.org	www.espressoedit.kataweb.it
dada.linuxberg.com	www.google.com	www.kataweb.it
www.kdevelop.org	citeseer.nj.nec.com	
www.newplanetsoftware.com	liinwww.ira.uka.de	
www.linuxberg.com	www.acm.org	
linuxpress.4mg.com	www.yahoo.it	
www.pluto.linux.it		
linuxberg.concepts.nl		
www.kde.org		
www.its.caltech.edu		

Fig. 5. Examples of Cluster representatives

These extensions make the algorithm proposed in this paper particularly suitable to on-line applications, i.e., applications that need an extremely fast computation of cluster assignments. Examples of such applications are clustering and reorganization of web search engines and analysis of web log accesses for on-line personalization. Formal and experimental results show that the algorithm maintains a linear scalability, and contemporarily the overall cost of the algorithm is not affected by the number of distinct values in the attribute domain.

The effectiveness of the approach is also proved by the quality and easy interpretability of the results, made viable by the definition of the cluster representative. The main drawback of the approach, i.e., the presence of unclustered objects can be overcome by suitably adapting the technique as shown in section 2.3. However, in the perspective of using the algorithms for on-line applications, the problem of unclustered objects can be profitably ignored in favour of efficiency, provided that the quality of the clusters actually computed is acceptable.

References

1. I. Dhillon and D. Modha. Concept Decomposition for Large Sparse Data Using Clustering. *Machine Learning*, 42:143–175, 2001. 178
2. D. Fasulo. An analysis of Recent Work on Clustering Algorithms. Technical report, University of Washington, April 1999. Available at <http://www.cs.washington.edu/homes/dfasulo>. 176
3. M. Grötschel and Y. Wakabayashi. A cutting plane algorithm for a clustering problem. *Mathematical Programming*, 45:59–96, 1989. 179
4. S. Guha, R. Rastogi, and K. Shim. ROCK: A robust clustering algorithm for categorical attributes. In *15th International Conference on Data Engineering (ICDE '99)*, pages 512–521, Washington - Brussels - Tokyo, March 1999. IEEE. 175
5. J. Han and M. Kamber. *Data Mining Techniques*. Morgan Kaufman, 2001. 183, 184
6. R. Cooley B. Mobasher and J. Srivastava. Grouping Web Page References into Transactions for Mining World Wide Web Browsing Patterns. In *Proceedings of the IEEE Knowledge and Data Engineering Exchange Workshop (KDEX-97)*, 1997. 185

7. L. Schulman. Clustering for Edge-Cost Minimization. In *Proceedings of the thirty-second annual acm symposium on Theory of computing*, pages 547–555, Portland, USA, May 2000. 183
8. R. Srikant. *Fast Algorithms for Mining Association Rules and Sequential Patterns*. PhD thesis, University of Wisconsin-Madison, 1996. 182
9. M. Steinbach, G. Karypis, and V. Kumar. A Comparison of Document Clustering Techniques. In *ACM-SIGKDD Workshop on Text Mining*, 2000. 178
10. A. Strehl, J. Ghosh, and R. Mooney. Impact of Similarity Measures on Web-page Clustering. In K. Bollacker, editor, *Proceedings of AAAI workshop on AI for Web Search*, pages 58–64. AAAI Press, July 2000. 178

Multiscale Comparison of Temporal Patterns in Time-Series Medical Databases

Shoji Hirano and Shusaku Tsumoto

Department of Medical Informatics, Shimane Medical University, School of Medicine
89-1 Enya-cho, Izumo, Shimane 693-8501, Japan
hirano@ieee.org

Abstract. This paper presents a method for analyzing time-series data on laboratory examinations based on phase-constraint multiscale matching and rough clustering. Multiscale matching compares two subsequences throughout various scales of view. It has an advantage of preserving connectivity of subsequences even if the subsequences are represented at different scales. Rough clustering groups up objects according not to the topographic measures such as the center or deviance of objects in a cluster but to the relative similarity and indiscernibility of objects. We use multiscale matching to obtain similarity of sequences and rough clustering to cluster the sequences according to the obtained similarity. We slightly modified dissimilarity measure in multiscale matching so that it suppresses excessive shift of phase that may cause incorrect matching of the sequences. Experimental results on the hepatitis dataset show that the proposed method successfully clustered similar sequences into an independent cluster, and that correspondence of subsequences are also successfully captured.

Keywords: multiscale matching, rough clustering, rough sets, medical data mining, temporal knowledge discovery

1 Introduction

Since hospital information systems were first introduced in large hospitals in 1980's, huge amount of time-series laboratory examination data, for example blood and biochemical examination data, have been stored in the databases. Recently, analysis of such temporal examination databases has been attracting much interests because it might reveal underlying relationships between temporal course of examination and onset of diseases. Long-term laboratory examination databases might also enable us to validate a hypothesis about temporal course of chronic diseases that has not been evaluated yet on large samples. However, despite their importance, time-series medical databases have not widely been considered as the subject of analysis. This is primarily due to inhomogeneity of the data. Basically, the data were collected without considering further use in automated analysis. Therefore it involves the following problems. (1) Missing values: Examinations are not performed on every day when a patient comes to

the hospital. It depends on the needs for examination. (2) Irregular interval of data acquisition: A patient consults a doctor in different interval depending on his/her condition, hospital's vacancy, and other factors. The intervals can vary from a few days to several months. (3) Noise: The data can be distorted due to contingent change of patient's condition. These problems make it difficult to compare similarity of temporal patterns on different patients. Therefore, the data have been mainly used for visual comparison among small samples, where the scale-merits of large temporal databases have not been exploited.

In this paper, we present a hybrid approach to the analysis of such inhomogeneous time-series medical databases. The techniques employed here are phase-constraint multiscale structure matching [1] and rough-sets based clustering technique [2]. The first one, multiscale structure matching, is a method that effectively compares two objects by partially changing observation scales. We apply this method to the time-series data, and examine similarity of two sequences in both long-term and short-term points of view. It has an advantage that connectivity of segments is preserved in the matching results even when the partial segments are obtained from different scales. We slightly modified dissimilarity measure in multiscale matching so that it suppresses excessive shift of phase that causes incorrect matching results. The second technique, rough-sets based clustering, clusters sequences based on their indiscernibility defined in the context of rough set theory [3]. The method can produce interpretable clusters even under the condition that similarity of objects is defined only as a relative similarity. Our method attempts to cluster the temporal sequences according to their long- and short-term similarity by combining the two techniques. First, we apply multiscale structure matching to all pairs of sequences and obtain similarity for each of them. Next, we apply rough-sets based clustering technique to cluster the sequences based on the obtained similarity. After then, common patterns in the clustered sequences can be visualized to understand relations to the diagnostic classes.

The remaining part of this paper is organized as follows. In Section 2 we introduce some related work. In Section 3 we describe the procedure of our method including explanation of each process such as preprocessing of data, multiscale structure matching and rough sets-based clustering. Then we show some experimental results in Section 4 and finally conclude the technical results.

2 Related Work

Data mining in time-series data has received much interests in both theoretical and applicational areas. A widely used approach in time-series data mining is to cluster sequences based on the similarity of their primary coefficients. Agrawal et al. [4] utilize discrete Fourier transformation (DFT) coefficients to evaluate similarity of sequences. Chan et al. [5] obtain the similarity based on the frequency components derived by the discrete wavelet transformation (DWT). Korn et al. [6] use singular value decomposition (SVD) to reduce complexity of sequences and compare the sequences according to the similarity of their eigen-

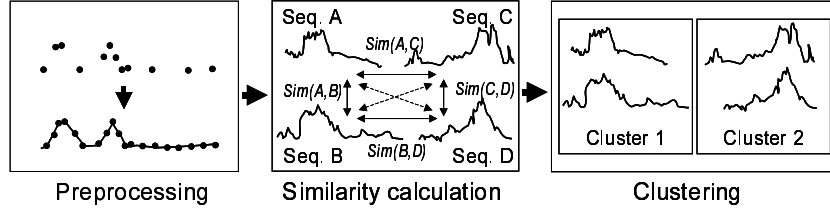


Fig. 1. Overview of the method

waves. Another approach includes comparison of sequences based on the similarity of forms of partial segments. Morinaka et al. [7] propose the L-index, which performs piecewise comparison of linearly approximated subsequences. Keogh et al. [8] propose a method called piecewise aggregate approximation (PAA), which performs fast comparison of subsequences by approximating each subsequence with simple box waves having constant length.

These methods can compare the sequences in various scales of view by choosing proper set of frequency components, or by simply changing size of the window that is used to translate a sequence into a set of simple waves or symbols. However, they are not designed to perform cross-scale comparison. In cross-scale comparison, connectivity of subsequences should be preserved across all levels of discrete scales. Such connectivity is not guaranteed in the existing methods because they do not trace hierarchical structure of partial segments. Therefore, similarity of subsequences obtained on different scales can not be directly merged into the resultant sequences. In other words, one can not capture similarity of sequences by partially changing scales of observation.

On the other hand, clustering has a rich history and a lot of methods have been proposed. They include, for example, k-means [9], fuzzy c-means [10], EM algorithm [11], CLIQUE [12], CURE [13] and BIRCH [14]. However, the similarity provided by multiscale matching is relative and not guaranteed to satisfy triangular inequality. Therefore, the methods based on the center, gravity or other types of topographic measures can not be applied to this task. Although classical agglomerative hierarchical clustering [15] can treat such relative similarity, in some case it has a problem that the clustering result depends on the order of handling objects.

3 Methods

3.1 Overview

Figure 1 shows an overview of the proposed method. First, we apply pre-processing to all the input sequences and obtain the interpolated sequences resampled in a regular interval. This procedure rearranges all data on the same time-scale and is required to compare long- and short-term difference using their length of trajectory. A simple linear interpolation of nearest neighbors is used to fill in a

missing value. Next, we apply multiscale structure matching to all possible combinations of two sequences and obtain their similarity as a matching score. We here restricted combinations of pairs so that they have the same attributes such as GPT-GPT, because our interest is not on the cross-attributes relationships. After obtaining similarity of the sequences, we cluster the sequences by using rough-set based clustering. Consequently, the similar sequences are clustered into the same clusters and their features are visualized.

3.2 Phase-Constraint Multiscale Structure Matching

Multiscale structure matching, proposed by Mokhtarian [17], is a method to describe and compare objects in various scales of view. Its matching criterion is similarity between partial contours. It seeks the best pair of partial contours throughout all scales, not only in the same scale. This enables matching of object not only from local similarity but also from global similarity. The method required much computation time because it should continuously change the scale, however, Ueda et al. [1] solved this problem by introducing a segment-based matching method which enabled the use of discrete scales. We use Ueda's method to perform matching of time sequences between patients. We associate a convex/concave structure in the time-sequence as a convex/concave structure of partial contour. Such a structure can be generated by increase/decrease of examination values. Then we can compare the sequences from different terms of observation.

Now let $x(t)$ denote a time sequence where t denotes time of examination. The sequence at scale σ , $X(t, \sigma)$, can be represented as a convolution of $x(t)$ and a Gauss function with scale factor σ , $g(t, \sigma)$, as follows:

$$\begin{aligned} X(t, \sigma) &= x(t) \otimes g(t, \sigma) \\ &= \int_{-\infty}^{+\infty} x(u) \frac{1}{\sigma\sqrt{2\pi}} e^{-(t-u)^2/2\sigma^2} du. \end{aligned}$$

Figure 2 shows an example of sequences in various scales. From Figure 2 and the function above, it is obvious that the sequence will be smoothed at higher scale and the number of inflection points is also reduced at higher scale. Curvature of the sequence can be calculated as

$$K(t, \sigma) = \frac{X''}{(1 + X'^2)^{3/2}},$$

where X' and X'' denotes the first- and second-order derivative of $X(t, \sigma)$, respectively. The m -th derivative of $X(t, \sigma)$, $X^{(m)}(t, \sigma)$, is derived as a convolution of $x(t)$ and the m -th order derivative of $g(t, \sigma)$, $g^{(m)}(t, \sigma)$, as

$$X^{(m)}(t, \sigma) = \frac{\partial^m X(t, \sigma)}{\partial t^m} = x(t) \otimes g^{(m)}(t, \sigma).$$

The next step is to find inflection points according to change of the sign of the curvature and to construct segments. A segment is a partial contour whose ends

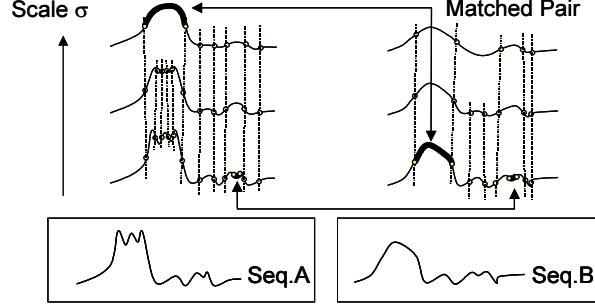


Fig. 2. Multiscale matching

correspond to the adjacent inflection points. Let $\mathbf{A}^{(k)}$ be a set of N segments that represents the sequence at scale $\sigma^{(k)}$ as

$$\mathbf{A}^{(k)} = \left\{ a_i^{(k)} \mid i = 1, 2, \dots, N^{(k)} \right\}.$$

Then, difference between segments $a_i^{(k)}$ and $b_j^{(h)}$, $d(a_i^{(k)}, b_j^{(h)})$ is defined as follows:

$$d(a_i^{(k)}, b_j^{(h)}) = \frac{|\theta_{a_i}^{(k)} - \theta_{b_j}^{(h)}|}{\theta_{a_i}^{(k)} + \theta_{b_j}^{(h)}} \left| \frac{l_{a_i}^{(k)}}{L_A^{(k)}} - \frac{l_{b_j}^{(h)}}{L_B^{(h)}} \right|,$$

where $\theta_{a_i}^{(k)}$ and $\theta_{b_j}^{(h)}$ denote rotation angles of tangent vectors along the contours, $l_{a_i}^{(k)}$ and $l_{b_j}^{(h)}$ denote length of the contours, $L_A^{(k)}$ and $L_B^{(h)}$ denote total segment length of the sequences A and B at scales $\sigma^{(k)}$ and $\sigma^{(h)}$. According to the above definition, large differences can be assigned when difference of rotation angle or relative length is large. Continuous $2n - 1$ segments can be integrated into one segment at higher scale. Difference between the replaced segments and another segment can be defined analogously, with additive replacement cost that suppresses excessive replacement.

The above similarity measure can absorb shift of time and difference of sampling duration. However, we should suppress excessive back-shift of sequences in order to correctly distinguish the early-phase events from late-phase events. Therefore, we extend the definition of similarity as follows.

$$d(a_i^{(k)}, b_j^{(h)}) = \frac{1}{3} \left(\left| \frac{d_{a_i}^{(k)}}{D_A^{(k)}} - \frac{d_{b_j}^{(h)}}{D_B^{(h)}} \right| + \frac{|\theta_{a_i}^{(k)} - \theta_{b_j}^{(h)}|}{\theta_{a_i}^{(k)} + \theta_{b_j}^{(h)}} + \left| \frac{l_{a_i}^{(k)}}{L_A^{(k)}} - \frac{l_{b_j}^{(h)}}{L_B^{(h)}} \right| \right),$$

where $d_{a_i}^{(k)}$ and $d_{b_j}^{(h)}$ denote dates from first examinations, $D_A^{(k)}$ and $D_B^{(h)}$ denote durations of examinations. By this extension, we can simultaneously evaluate the

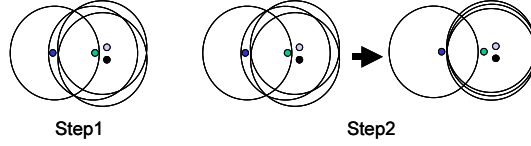


Fig. 3. Rough clustering

following three similarities: (1) dates of events (2) velocity of increase/decrease (3) duration of each event.

The remaining procedure of multiscale structure matching is to find the best pair of segments that minimizes the total difference. Figure 2 illustrates the process. For example, in the upper part of Figure 2, five contiguous segments at the lowest scale of Sequence A are integrated into one segment at the highest scale, and this segment is well matched to one segment in Sequence B at the lowest scale. While, another pair of segments is matched at the lowest scale. In this way, matching is performed throughout all scales. The matching process can be fasten by implementing dynamic programming scheme. For more details, see ref [1]. After matching process is completed, we calculate the remaining difference and use it as a measure of similarity between sequences.

3.3 Rough-Sets Based Clustering

Generally, if similarity of objects is represented only as a relative similarity, it is not an easy task to construct interpretable clusters because some of important measures such as inter- and intra-cluster variances are hard to be defined. The rough-set based clustering method is a clustering method that clusters objects according to the indiscernibility of objects. It represents denseness of objects according to the *indiscernibility degree*, and produces interpretable clusters even for the objects mentioned above. Since similarity of sequences obtained through multiscale structure matching is relative, we use this clustering method to classify the sequences.

The clustering method lies its basis on the *indiscernibility* of objects, which forms basic property of knowledge in rough sets. Let us first introduce some fundamental definitions of rough sets related to our work. Let $U \neq \phi$ be a universe of discourse and X be a subset of U . An equivalence relation, R , classifies U into a set of subsets $U/R = \{X_1, X_2, \dots, X_m\}$ in which following conditions are satisfied:

- (1) $X_i \subseteq U, X_i \neq \phi$ for any i ,
- (2) $X_i \cap X_j = \phi$ for any i, j ,
- (3) $\cup_{i=1,2,\dots,n} X_i = U$.

Any subset X_i , called a category, represents an equivalence class of R . A category in R containing an object $x \in U$ is denoted by $[x]_R$. For a family of equivalence relations $\mathbf{P} \subseteq \mathbf{R}$, an indiscernibility relation over $\mathbf{P}$ is denoted by $IND(\mathbf{P})$ and

defined as follows

$$IND(\mathbf{P}) = \{(x_i, x_j) \in U^2 \mid \forall Q \in \mathbf{P}, [x_i]_Q = [x_j]_Q\}.$$

The clustering method consists of two steps: (1) assignment of initial equivalence relations and (2) iterative refinement of initial equivalence relations. Figure 3 illustrates each step. In the first step, we assign an initial equivalence relation to every object. An initial equivalence relation classifies the objects into two sets: one is a set of objects similar to the corresponding objects and another is a set of dissimilar objects. Let $U = \{x_1, x_2, \dots, x_n\}$ be the entire set of n objects. An initial equivalence relation R_i for object x_i is defined as

$$R_i = \{\{P_i\}, \{U - P_i\}\},$$

$$P_i = \{x_j \mid s(x_i, x_j) \geq S_i\}, \quad \forall x_j \in U.$$

where P_i denotes a set of objects similar to x_i . Namely, P_i is a set of objects whose similarity to x_i , s , is larger than a threshold value S_i . Here, s corresponds to the inverse of the output of multiscale structure matching, and S_i is determined automatically at a place where s largely decreases. A set of indiscernible objects obtained using all sets of equivalence relations corresponds to a cluster. In other words, a cluster corresponds to a category X_i of $U/IND(\mathbf{R})$.

In the second step, we refine the initial equivalence relations according to their global relationships. First, we define an indiscernibility degree, γ , which represents how many equivalence relations commonly regards two objects as indiscernible objects, as follows:

$$\gamma(x_i, x_j) = \frac{1}{|U|} \sum_{k=1}^{|U|} \delta_k(x_i, x_j),$$

$$\delta_k(x_i, x_j) = \begin{cases} 1, & \text{if } [x_k]_{R_k} \cap ([x_i]_{R_k} \cap [x_j]_{R_k}) \neq \emptyset \\ 0, & \text{otherwise.} \end{cases}$$

Objects with high indiscernibility degree can be interpreted as similar objects. Therefore, they should be classified into the same cluster. Thus we modify an equivalence relation if it has ability to discern objects with high γ as follows:

$$R'_i = \{\{P'_i\}, \{U - P'_i\}\},$$

$$P'_i = \{x_j \mid \gamma(x_i, x_j) \geq T_h\}, \quad \forall x_j \in U.$$

This prevents generation of small clusters formed due to the too fine classification knowledge. T_h is a threshold value that determines indiscernibility of objects. Therefore, we associate T_h with roughness of knowledge and perform iterative refinement of equivalence relations by constantly decreasing T_h . Consequently, coarsely classified set of sequences are obtained as $U/IND(\mathbf{R}')$.

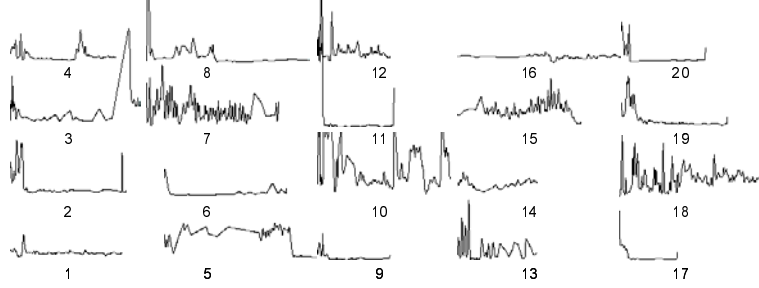


Fig. 4. Test patterns

Table 1. Similarity of the sequences

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	1.00	0.70	0.68	0.78	0.00	0.63	0.48	0.71	0.72	0.61	0.73	0.66	0.64	0.72	0.50	0.00	0.53	0.00	0.74	0.45
2		1.00	0.61	0.73	0.00	0.68	0.22	0.46	0.68	0.67	0.72	0.73	0.72	0.68	0.54	0.00	0.68	0.00	0.77	0.41
3			1.00	0.75	0.45	0.51	0.68	0.47	0.71	0.70	0.69	0.73	0.71	0.81	0.68	0.00	0.62	0.00	0.72	0.55
4				1.00	0.00	0.60	0.52	0.47	0.75	0.71	0.64	0.79	0.75	0.82	0.47	0.00	0.60	0.00	0.75	0.48
5					1.00	0.23	0.62	0.49	0.33	0.53	0.44	0.45	0.50	0.44	0.56	0.01	0.00	0.26	0.53	0.30
6						1.00	0.00	0.00	0.59	0.00	0.58	0.39	0.61	0.65	0.00	0.00	0.47	0.00	0.47	0.48
7							1.00	0.49	0.54	0.80	0.57	0.73	0.73	0.59	0.76	0.00	0.00	0.44	0.62	0.39
8								1.00	0.53	0.47	0.57	0.56	0.51	0.49	0.54	0.00	0.00	0.00	0.66	0.51
9									1.00	0.00	0.68	0.00	0.00	0.00	0.00	0.00	0.82	0.00	0.00	0.00
10										1.00	0.59	0.83	0.76	0.75	0.81	0.00	0.47	0.11	0.59	0.37
11											1.00	0.76	0.54	0.68	0.00	0.00	0.74	0.00	0.76	0.00
12												1.00	0.81	0.78	0.67	0.00	0.70	0.00	0.63	0.40
13													1.00	0.75	0.00	0.00	0.64	0.00	0.67	0.35
14														1.00	0.00	0.00	0.66	0.00	0.71	0.00
15															1.00	0.00	0.43	0.20	0.55	0.39
16																1.00	0.00	0.00	0.43	0.19
17																	1.00	0.00	0.00	0.00
18																		1.00	0.39	0.03
19																			1.00	0.00
20																				1.00

4 Experimental Results

We applied the proposed method to time-series GPT sequences in the hepatitis data set [18]. The dataset contained long time-series data on laboratory examinations, which were collected on a university hospital in Japan. The subjects were 771 patients of hepatitis B and C who took examinations between 1982 and 2001. Due to incompleteness in data acquisition, time-series GPT sequences were available only for 195 of 771 patients.

First, in order to evaluate applicability of multiscale matching to time-series data analysis, we applied the proposed method to a small subset of sequences which was constructed by randomly selecting 20 sequences from the data set. Figure 4 shows all the pre-processed sequences. Each sequence originally has different sampling intervals from one day to one year. From preliminary analysis we found that the most frequently appeared interval was one week; this means that most of the patients took examinations on a fixed day of a week. According to this observation, we determined resampling interval to seven days.

Table 1 shows normalized similarity of the sequences derived by multiscale matching. Since consistency of self-similarity ($s(A, B) = s(B, A)$) holds, the

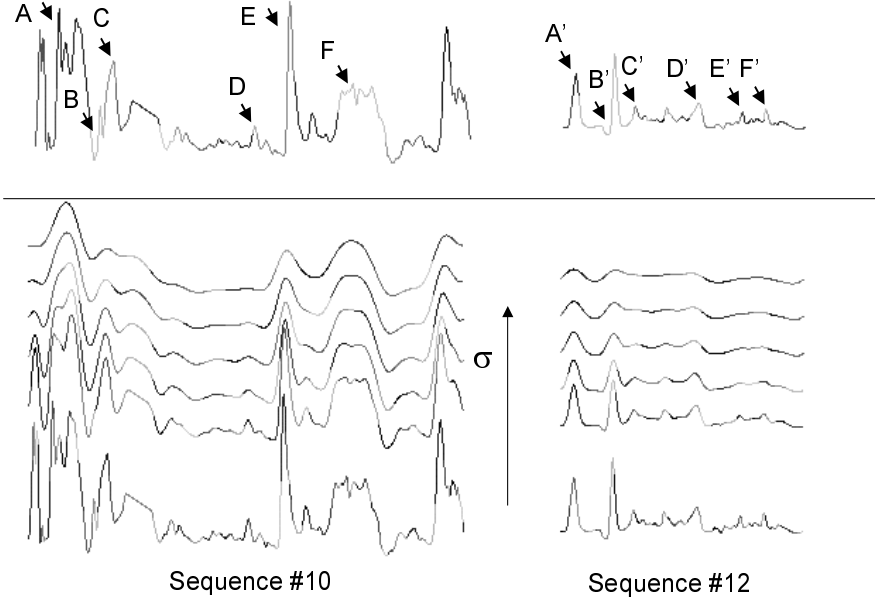


Fig. 5. Matching result of sequences #10 and #12

lower-left half of the matrix is omitted. We can observe that higher similarity was successfully assigned to intuitively similar pairs of sequences.

Based on this similarity, the rough clustering produced nine clusters: $U/IND(\mathbf{R}) = \{\{1,2,9,11,17,19\}, \{4,3,8\}, \{7,14,15\}, \{10,12,13\}, \{5\}, \{6\}, \{16\}, \{18\}, \{20\}\}$. A parameter Th for rough clustering was set to $Th = 0.6$. Refinement was performed up to five times with constantly decreasing Th toward $Th = 0.4$. It can be seen that similar sequences were clustered into the same cluster. Some sequences, for example #16, were clustered into independent clusters due to remarkably small similarity to other sequences. This is because multiscale matching could not find good pairs of subsequences.

Figure 5 shows the result of multiscale matching on sequences #10 and #12, that have high similarity. We changed σ from 1.0 to 13.5, with intervals of 2.5. At the bottom of the figure there are original two sequences at $\sigma = 1.0$. The next five sequences represent sequences at scales $\sigma = 3.5, 6.0, 8.5, 11.0$, and 13.5 , respectively. Each of the colored line corresponds to a segment. The matching result is shown at the top of the figure. Here the lines with same color represent the matched segments, for example, segment A matches segment A' and segment B matches segment B' . We can clearly observe that increase/decrease patterns of sequences are successfully captured; large increase (A and A'), small decrease with instant increase (B and B'), small increase (C and C') and so on. Segments $D - F$ and $D' - F'$ have similar patterns and the feature was

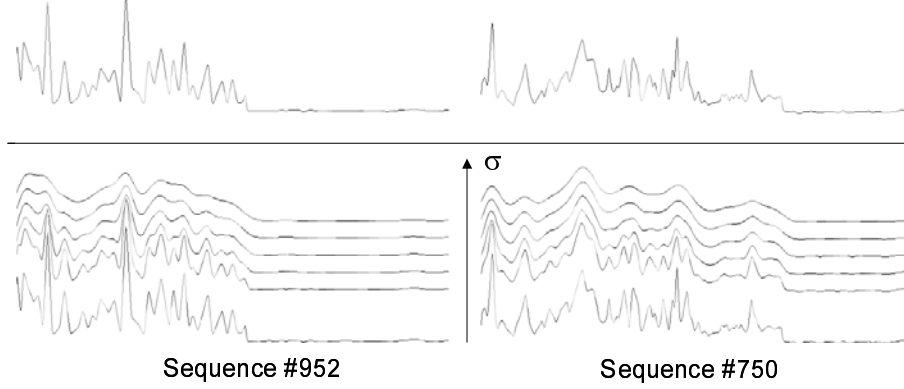


Fig. 6. Matching result of sequences #952 and #750

also correctly captured. It can also be seen that the well-matched segments were obtained in the sequences with large time difference.

Next, we applied the proposed method to the full data set containing 195 GPT sequences. For this data set, rough clustering produced 14 clusters:

$U/IND(\mathbf{R}) = \{\{2, 19, 36, 37, 49, \dots, 953, 955 \text{ (total 165 sequences)}\}, \{16, 35, 111\}, \{86, 104, 142, 171, 215, 273, 509, 523, 610, 663\}, \{149\}, \{703\}, \{706\}, \{737\}, \{740\}, \{743, 801, 894, 897, 942\}, \{750, 952\}, \{771\}, \{533, 594\}, \{689, 731\}\}$, where a sequence number corresponds to a masked ID of the patient. The first cluster seems to be uninteresting because it contains too many sequences. This cluster was generated as a result of improper assignment of Th , which caused excessive refinement of clusters. However instead, we could find very interesting patterns in other clusters. For example, the 10th cluster contained sequences 750 and 952, which had very similar patterns as shown in Figure 6. In both sequences, increase and decrease of GPT values were repeatedly observed in the early half period of data acquisition, and they became flat in the late period. A physician evaluated that this might be an interesting pattern that represents degree of damage of the liver.

5 Conclusions

In this paper, we have presented an analysis method of time-series medical databases based on the hybridization of phase-constraint multiscale structure matching and rough clustering. The method first obtained similarity of sequences by multiscale comparison of sequences in which connectivity of subsequences were preserved even if they were represented at different scales. Then rough clustering grouped up the sequences according to their relative similarity. This hybridization enabled us not only to cluster time-series sequence from both long- and short-term viewpoints but also to visualize correspondence of subsequences.

In the experiments on the hepatitis data set, we showed that the sequences were successfully clustered into intuitively correct clusters, and that some interesting patterns were discovered by visualizing the clustered sequences. It remains as a future work to evaluate usefulness of the method in other databases.

Acknowledgment

This work was supported in part by the Grant-in-Aid for Scientific Research on Priority Area (B)(No.759) “Implementation of Active Mining in the Era of Information Flood” by the Ministry of Education, Culture, Science and Technology of Japan.

References

1. N. Ueda and S. Suzuki (1990): A Matching Algorithm of Deformed Planar Curves Using Multiscale Convex/Concave Structures. *IEICE Transactions on Information and Systems*, J73-D-II(7): 992–1000. 189, 191, 193
2. S. Hirano and S. Tsumoto (2001): Indiscernibility Degrees of Objects for Evaluating Simplicity of Knowledge in the Clustering Procedure. *Proceedings of the 2001 IEEE International Conference on Data Mining*. 211–217. 189
3. Z. Pawlak (1991): *Rough Sets, Theoretical Aspects of Reasoning about Data*. Kluwer Academic Publishers, Dordrecht. 189
4. R. Agrawal, C. Faloutsos, and A. N. Swami (1993): Efficient Similarity Search in Sequence Databases. *Proceedings of the 4th International Conference on Foundations of Data Organization and Algorithms*: 69–84. 189
5. K. P. Chan and A. W. Fu (1999): Efficient Time Series Matching by Wavelets. *Proceedings of the 15th IEEE International Conference on Data Engineering*: 126–133. 189
6. F. Korn, H. V. Jagadish, and C. Faloutsos (1997): Efficiently Supporting Ad Hoc Queries in Large Datasets of Time Sequences. *Proceedings of ACM SIGMOD International Conference on Management of Data*: 289–300. 189
7. Y. Morinaka, M. Yoshikawa, T. Amagasa and S. Uemura (2001): The L-index: An Indexing Structure for Efficient Subsequence Matching in Time Sequence Databases. *Proceedings of International Workshop on Mining Spatial and Temporal Data, PAKDD-2001*: 51–60. 190
8. E. J. Keogh, K. Chakrabarti, M. J. Pazzani, and S. Mehrotra (2001): “Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases” *Knowledge and Information Systems* 3(3): 263–286. 190
9. S. Z. Selim and M. A. Ismail (1984): K-means-type Algorithms: A Generalized Convergence Theorem and Characterization of Local Optimality. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(1): 81–87. 190
10. J. C. Bezdek (1981): *Pattern Recognition with Fuzzy Objective Function Algorithm*. Plenum Press, New York. 190
11. A. P. Dempster, N. M. Laird, and D. B. Rubin (1977): Maximum likelihood from incomplete data via the EM algorithm. *J. of Royal Statistical Society Series B*, 39: 1–38. 190

12. R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan (1998): Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications. Proceedings of ACM SIGMOD International Conference on Management of Data: 94–105. 190
13. S. Guha, R. Rastogi, and K. Shim(1998): CURE: An Efficient Clustering Algorithm for Large Databases. Proceedings of ACM SIGMOD International Conference on Management of Data: 73–84. 190
14. T. Zhang, R. Ramakrishnan, and M. Livny (1996): BIRCH: An Efficient Data Clustering Method for Very Large Databases. Proceedings of ACM SIGMOD International Conference on Management of Data: 103–114. 190
15. M. R. Anderberg (1973): Cluster Analysis for Applications. Academic Press, New York. 190
16. R. H. Shumway and D. S. Stoffer (2000): Time Series Analysis and Its Applications. Springer-Verlag, New York.
17. F. Mokhtarian and A. K. Mackworth (1986): Scale-based Description and Recognition of planar Curves and Two Dimensional Shapes. IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-8(1): 24-43. 191
18. URL: <http://lisp.vse.cz/challenge/ecmlpkdd2002/> 195

Association Rules for Expressing Gradual Dependencies

Eyke Hüllermeier

Department of Mathematics and Computer Science
University of Marburg, Germany
`eyke@mathematik.uni-marburg.de`

Abstract. Data mining methods originally designed for binary attributes can generally be extended to quantitative attributes by partitioning the related numeric domains. This procedure, however, comes along with a loss of information and, hence, has several disadvantages. This paper shows that fuzzy partitions can overcome some of these disadvantages. Particularly, fuzzy partitions allow for the representation of association rules expressing a tendency, that is, a gradual dependence between attributes. This type of rule is introduced and investigated from a conceptual as well as a computational point of view. The evaluation and representation of a gradual association is based on linear regression analysis. Furthermore, a complementary type of association, expressing absolute deviations rather than tendencies, is discussed in this context.

1 Introduction

Data mining aims at extracting understandable pieces of knowledge from usually large sets of data stored in a database. It comes as no surprise that *rule-based* models play a prominent role in this field, as rules provide a simple and intelligible yet expressive means of knowledge representation. Among the related techniques that have been developed, so-called *association rules* (or associations for short) have gained considerable attraction [1]. An association rule is meant to represent dependencies between attributes in a databases. Typically, such a rule involves two sets A and B of binary attributes, also called features or items. The intended meaning of a rule symbolized as $A \rightarrow B$ is that a transaction (a data record stored in the database) that contains the set of items A is likely to contain the items B as well.

Generally, a database does not contain binary attributes only but also attributes with values ranging on (completely) ordered scales, e.g. cardinal or ordinal attributes. This has motivated a corresponding generalization of (binary) association rules [5]. The simplest approach, to be detailed in Section 2, is to partition the domain $\mathfrak{D}_X$ of a quantitative attribute X into intervals $A \subseteq \mathfrak{D}_X$ and to associate a new binary variable with each interval. This leads to interval-based association rules of the form $X \in A \rightarrow Y \in B$.

A slightly different type of association rule, particularly interesting in connection with quantitative attributes, has recently been considered in [2]. This

type of rule, which has a more statistical flavor, is of the following form:

$$X \in A \rightarrow \text{mean}(Y) = \bar{y}_A, \quad (1)$$

where X and Y are attributes and A is an interval. This rule says that the mean value of Y is $\bar{y}_A$ if the database is restricted to those transactions satisfying $X \in A$, an information which is clearly interesting if $\bar{y}_A$ deviates significantly from the overall (unconditional) mean $\bar{y}$. The basic idea underlying this approach can be summarized as follows: The (empirical) *distribution* of an attribute Y changes significantly when focusing on a certain subpopulation (a subset of the database). In this connection, a subpopulation is specified by the condition $X \in A$, and the change of the distribution is measured by the change of the mean. Clearly, the mean could be replaced by any other statistic of interest, for example the variance or the median, or even by the distribution of Y itself. See [3] for a closely related data mining method.

In this paper, we elaborate further on quantitative association rules. Especially, we propose a new type of rule which is able to express a kind of *tendency*, that is, a gradual dependence between attributes. In this connection, the idea of a “fuzzy” partition of a quantitative domain plays an important role. After having pointed out some difficulties caused by classical partitions (Section 2), this idea will be motivated in Section 3. In Section 4, two types of association rules will be introduced, namely the aforementioned rules expressing a tendency and a complementary type of rule expressing absolute deviations.

Notation. We proceed from a database $\mathcal{D}$, which is a collection of transactions (records) t . A transaction t assigns a value $t[A]$ to each attribute $A \in \mathcal{A}$, where $\mathcal{A}$ is an underlying set of attributes. We focus on cardinally scaled attributes X ; the domain of an attribute X is denoted $\mathfrak{D}_X$. When discussing simple rules involving two (fixed) attributes X and Y , we consider the database $\mathcal{D}$ as a collection of data points $(x, y) = (t[X], t[Y])$, i.e. as a projection of the original database. This notation is generalized to rules involving more variables in a canonical manner. An association rule is written in the form $A \rightarrow B$, or sometimes $\{A \rightarrow B\}$, where A and B can be single items or sets of items.

2 Problems with Binary Partitions

Most algorithms in data mining have been designed for binary variables, and methods capable of dealing with quantitative attributes are mostly extensions of these algorithms. A standard approach in this connection is to replace a quantitative attribute X with domain $\mathfrak{D}_X$ by a finite set of binary variables X_{A_i} with domain $\{0, 1\}$, where the $A_i \subseteq \mathfrak{D}_X$, $1 \leq i \leq k$, are intervals such that $\bigcup_{i=1}^k A_i = \mathfrak{D}_X$. An attribute X_{A_i} takes the value 1 if the related quantitative value x is covered by A_i and 0 otherwise: $X_{A_i} = 1 \Leftrightarrow x \in A_i$. Algorithms for binary attributes can then be applied to the new (transformed) data set.

Needless to say, this type of binarization comes along with a loss of information, since the precise value x cannot be recovered from the values of the

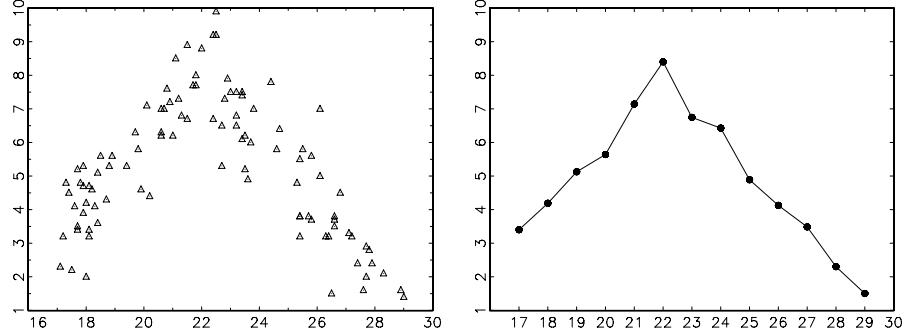


Fig. 1. Left: Observations (transactions) plotted as points in the instance space (x-axis: **weight**, y-axis: **perf**). Right: Mean performance for subclasses $D_w \doteq \{(x, y) \in D \mid w - 1/2 < x \leq w + 1/2\}$, $w = 17, 18, \dots, 29$

binary variables $X_{A_1}, \dots, X_{A_k}$. Likewise, a subset $A_i \times B_j$ of the *instance space* $\mathfrak{D}_X \times \mathfrak{D}_Y$ becomes a “black box” when considering two variables X and Y . Information about the *distribution* of points $(x, y) \in A_i \times B_j$ is hence completely lost, which makes it impossible to discover *local* interdependencies between X and Y . Consequently, interval-based rules might convey a misleading picture of the underlying data. These problems are further aggravated by the sharp boundaries between the intervals or, more generally, between the range of support and non-support of a binary feature.

To illustrate, consider an artificial data set comprised of 100 data points (x_i, y_i) . The related variables X and Y can be thought of as, say, the weight of a dog in kg (**weight**) and a certain physical performance (**perf**) measured on a scale ranging from 0 (bad) to 10 (excellent). The following table shows some of the data:

i	1	2	3	4	5	6	7	8	9	11 ...
x_i	22.4	25.8	22.4	27.7	27.7	25.7	20.2	18.1	18.1	17.4 ...
y_i	6.7	5.6	9.2	2.0	2.9	3.8	4.4	3.4	4.7	4.5 ...

Fig. 1 (left) shows the complete data as points in the instance space. On average, **perf** seems to increase with **weight** up to a value of about 22, and to decrease afterwards. This impression is confirmed by the right picture in the same figure, showing the mean values of **perf** separately for the subpopulations $D_w \doteq \{(x, y) \in D \mid w - 1/2 < x \leq w + 1/2\}$, $w = 17, 18, \dots, 29$.

Now, a rule of the following kind would nicely characterize the above data: *The more normal the weight, the better the performance*. Note that this rule expresses a *tendency*, which cannot be accomplished by a (single) classical association rule. Moreover, the above rule involves a cognitive concept, namely “normal weight”, which is here understood as “a weight close to 22 kg”. Such concepts do often preexist in the head of a data miner, but are seldom adequately represented by an interval created in the course of a rule mining procedure.

In our example, a reasonable candidate for an interval-based association would be a rule of the form

$$\text{weight} \in [22 - \gamma, 22 + \gamma] \rightarrow \text{perf} \in [7, 10], \quad (2)$$

suggesting that a dog whose weight is close to 22 kg is likely to perform rather well. But how should γ be chosen? Note that (2), in conjunction with the rule

$$\text{weight} \notin [22 - \gamma, 22 + \gamma] \rightarrow \text{perf} \in [0, 7[,$$

induces a classification into low-performance and high-performance dogs. The borderline between these two groups is clearly arbitrary to some extent, and the classification will hardly reflect the true nature of the data.

The same problem occurs in the approach in [2], where a rule would be specified as

$$\text{weight} \in [22 - \gamma, 22 + \gamma] \rightarrow \text{mean}(\text{perf}) = f(\gamma). \quad (3)$$

Here, the (conditional) mean would be a decreasing function of γ , and the rule (3) would be considered as interesting only if $f(\gamma)$ is –in a statistical sense– *significantly* larger than the overall mean of **perf**. Again, there is no natural choice of the length of the interval. In [2], γ is basically determined by the confidence level of a t-test used for testing significance, which only removes but does not solve the problem.

Finally, let us mention two further problems of the interval-based approach. Firstly, sharp boundaries between intervals may lead to undesirable threshold effects, in much the same way as do *histograms* in statistics: A slight variation of the boundary points of the intervals can have a considerable effect on the histogram induced by a number of observations and may even lead to qualitative changes, that is changes of the shape of the histogram. Likewise, the variation of an interval can strongly influence the evaluation of a related association rule. Secondly, the interval-based approach becomes involved if the class of allowed intervals is not restricted in a proper way (for example in the form of fixed underlying partitions for the attributes). On the one hand, a rich class of intervals guarantees flexibility and representational power. On the other hand, one has to keep track of possible interactions between apparently interesting rules. For example, the antecedent and/or the consequent parts of two rules can overlap, which may cause problems of redundancy.

In summary, this section has pointed out the following difficulties of interval-based associations: Firstly, such rules are not able to express gradual dependencies between attributes. Secondly, some problems are caused by sharp boundaries: Their specification is often arbitrary, the evaluation of rules is sensitive toward the variation of boundary points, and rules are not very user-friendly due to a lack of readability and “cognitive relevance”. Thirdly, additional complications and computational costs occur if interactions between interval-based rules are not excluded in advance by restricting the class of allowed intervals.

3 Fuzzy Partitions

The use of fuzzy sets in connection with association rules – as with data mining in general [8] – has recently been motivated by several authors (e.g. [6]). Among other aspects, many of the aforementioned problems can be avoided – or at least alleviated – by the use of *fuzzy* instead of crisp (non-fuzzy) partitions. A fuzzy subset of a set (domain) $\mathfrak{D}$ is identified by a so-called membership function, which is a generalization of the characteristic function $\mathbb{I}_A(\cdot)$ of an ordinary set $A \subseteq \mathfrak{D}$ [10]. For each element $x \in \mathfrak{D}$, this function specifies the degree of membership of x in the fuzzy set. Usually, membership degrees are taken from the unit interval $[0, 1]$, i.e. a membership function is a mapping $\mathfrak{D} \rightarrow [0, 1]$. We shall use the same notation for ordinary sets and fuzzy sets. Moreover, we shall not distinguish between a fuzzy set and its membership function, that is, $A(x)$ denotes the degree of membership of the element x in the fuzzy set A . Note that an ordinary set A can be considered as a “degenerate” fuzzy set with membership degrees $A(x) = \mathbb{I}_A(x) \in \{0, 1\}$.

Fuzzy sets formalize the idea of *graded membership*, which allows an element to belong “more or less” to a set. A fuzzy set can have “non-sharp” boundaries. Consider the above mentioned concept of “normal weights” as an example. Is it reasonable to say that 23.4 kg is a normal weight (for a dog in our example) but 23.5 kg is not? In fact, any sharp boundary will appear rather arbitrary. Modeling the concept “normal weight” as a fuzzy set A , it becomes possible to express, for example, that a weight of 22 kg is completely in accordance with this concept ($A(22) = 1$), 24 kg is a “more or less” normal weight ($A(24) = 0.5$, say), and 26 kg is clearly not normal ($A(26) = 0$).

As can be seen, fuzzy sets can provide a reasonable representation of linguistic expressions and cognitive concepts. This way, they act as an interface between a quantitative, numerical level and a qualitative level, where knowledge is expressed in terms of natural language. In data mining, fuzzy sets thus allow for expressing patterns found at the quantitative (database) level in a user-friendly way.

Concerning the class of fuzzy concepts underlying the rule mining process, we advocate a fixed partition for each attribute. Even though the assumption of a fixed partition is often regarded as critical, it appears particularly reasonable in the fuzzy case. Apart from a simplification of the rule mining procedure, a fixed partition specified by the user or data miner himself guarantees the interpretability of the rules. In fact, the user will generally have a concrete idea of terms such as “normal weight”. Since it is the user who interprets the association rules, these rules should exactly reflect the meaning he has in mind and, hence, the user himself should characterize each linguistic expression in terms of a fuzzy set.

In this connection, it is worth mentioning that a given class of fuzzy concepts can be extended through the use of so-called (linguistic) modifiers. For example, applying the linguistic hedge (modifier) “almost” to the fuzzy concept “normal weight” – modeled by a fuzzy set A – yields the new concept “almost normal weight”. Formally, this concept is represented by means of a suitable transfor-

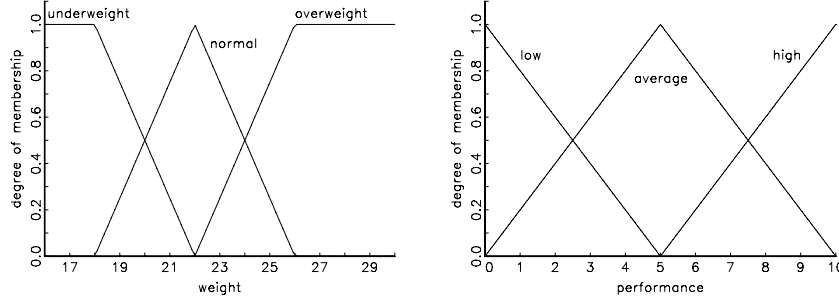


Fig. 2. Exemplary fuzzy partitions of the domains for weight and performance

mation of A . This way, a large number of interpretable fuzzy concepts can be built from a basic repertoire of fuzzy sets and modifier functions. However, we shall not elaborate any further on this aspect. Rather, we assume a fixed fuzzy partition for each attribute X . Formally, such a partition is defined as a class $\{A_1, \dots, A_k\}$ of fuzzy sets $A_i : \mathfrak{D}_X \rightarrow [0, 1]$ such that $\max_{1 \leq i \leq k} A_i(x) > 0$ for all $x \in \mathfrak{D}_X$. Fig. 2 shows fuzzy partitions of the domain of weights (using three fuzzy sets: underweight, normal, overweight) and the domain of performance (again with three fuzzy sets: low, average, high).

The discussion so far has shown that a (fixed) fuzzy partition can avoid some of the drawbacks related to classical partitions. Concerning the idea of association rules capable of expressing gradual dependencies between attributes, the following section will show that fuzzy partitions can also be beneficial in that respect.

4 Tendency and Deviation Rules

The basic quality measures for binary association rules $A \rightarrow B$ can be derived from the following contingency table:

	$B(y) = 0$	$B(y) = 1$	
$A(x) = 0$	n_{00}	n_{01}	$n_{0\bullet}$
$A(x) = 1$	n_{10}	n_{11}	$n_{1\bullet}$
	$n_{\bullet 0}$	$n_{\bullet 1}$	n

(4)

For example, the well-known support and confidence of a rule are given, respectively, by $\text{supp}(A \rightarrow B) = n_{11}/n$ and $\text{conf}(A \rightarrow B) = n_{11}/n_{1\bullet}$, where $n_{i,j}$ ($i, j \in \{0, 1\}$) is the number of tuples $(x, y) \in \mathcal{D}$ such that $A(x) = i$ and $B(y) = j$.

In the fuzzy case, $A(x)$ and $B(y)$ can take any value in the unit interval. This suggests extending the above contingency table to a *contingency diagram* as shown in Fig. 3. A record $(x, y) \in \mathcal{D}$ gives rise to a point with coordinates (u, v) in this diagram, where $u = A(x)$ is the degree of membership of x in A (the abscissa) and $v = B(y)$ is the degree of membership of y in B (the ordinate). As

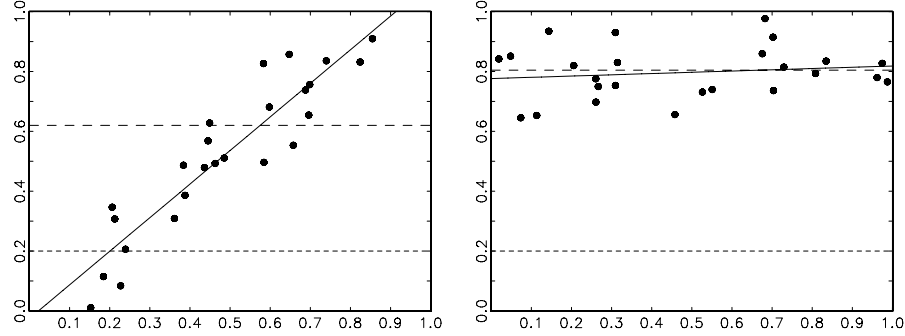


Fig. 3. Exemplary contingency diagrams for an underlying association $A \rightarrow B$. Each point is associated with a sample $(x, y) \in \mathcal{D}_A$: The abscissa corresponds to the membership of x in the fuzzy set A , the ordinate is the membership degree $B(y)$. The lines drawn by short and long dashes mark, respectively, the overall and conditional (given x is in A) mean value of $v = B(y)$. The third line is the regression line

will be seen, the contingency diagram provides a useful point of departure for specifying an association between A and B . Note that (4) can indeed be seen as a special case of a contingency diagram: In the non-fuzzy (binary) case, all points are located in the four “corners” of this diagram.

4.1 Contingency Diagrams

In order to illustrate the concept of a contingency diagram, consider the exemplary diagrams shown in Fig. 3. The following information is provided: Each point in a diagram corresponds to a tuple $(A(x_i), B(y_i))$, where $(x_i, y_i) \in \mathcal{D}_A \doteq \{(x, y) \in \mathcal{D} \mid A(x) > 0\}$; the objects (x, y) with $A(x) = 0$ are ignored. The solid line is the regression line derived for the points $\mathcal{D}_A$, i.e. the linear approximation $u \mapsto \alpha u + \beta$ minimizing the sum of squared errors

$$\sum_{i=1}^{|\mathcal{D}_A|} (\alpha A(x_i) + \beta - B(y_i))^2. \quad (5)$$

The line drawn by short dashes marks the overall mean value of $v = B(y)$, that is the value $\bar{v} \doteq |\mathcal{D}|^{-1} \sum_{(x,y) \in \mathcal{D}} B(y)$. This is the average degree to which the objects in $\mathcal{D}$ have the property B . Finally, the line drawn by long dashes shows the conditional mean of $v = B(y)$, given that x is in A . Since A is a fuzzy set, this value is calculated as a weighted average:

$$\bar{v}_A \doteq \left(\sum_{(x,y) \in \mathcal{D}} A(x) \cdot B(y) \right) \cdot \left(\sum_{(x,y) \in \mathcal{D}} A(x) \right)^{-1}.$$

Now, consider the first diagram in more detail. As can be seen, there is a strong correlation between $A(x)$ and $B(y)$. In fact, the positive slope of the regression line suggests the following tendency: *The more x is in A , the more y is in B .* Moreover, the conditional mean $\bar{v}_A$ appears to be significantly larger than the overall mean $\bar{v}$. The basic idea of our approach, to be detailed below, is to derive a suitable (linguistic) representation of an association $A \rightarrow B$ on the basis of this information.

The regression line in the second diagram has a slope close to 0. Still, the conditional mean $\bar{v}_A$ is much larger than the overall mean $\bar{v}$, suggesting a rule of the following kind: *If x is in A , then y is more in B than usual.*

4.2 From Contingency Diagrams to Association Rules

Clearly, the information provided by the contingency diagram can be regarded as reliable only if the diagram contains enough points. First of all, we therefore apply the common support criterion: A rule $A \rightarrow B$ is taken into consideration only if $\text{supp}(A \rightarrow B)$ exceeds a given threshold σ , where $\text{supp}(A \rightarrow B) \doteq \sum_{(x,y) \in \mathcal{D}} A(x)$. Note that this definition of support differs from the usual definition of fuzzy support, which is $\text{supp}(A \rightarrow B) \doteq \sum_{(x,y) \in \mathcal{D}} \min\{A(x), B(y)\}$.¹ In fact, it is modeled on the two types of association rules that will be introduced below: It corresponds to the (fuzzy) number of points considered when evaluating a rule of the first type and defines a lower bound to the number of involved points in second case.

Information from a Contingency Diagram. We proceed from the following information taken from the contingency diagram:

- The mean values $\bar{v}$, $\bar{v}_A$ (and the number of points $n_A \doteq |\mathcal{D}_A|$).
- The coefficients α, β of the regression line.
- A measure Q indicating the quality of the regression.

Here, we take Q as the usual R^2 coefficient, defined as

$$R^2 \doteq 1 - \frac{\sum_{i=1}^{n_A} e_i^2}{\sum_{i=1}^{n_A} (v_i - \bar{v}_A)^2},$$

where $e_i = v_i - (\alpha u_i + \beta)$, $(u_i, v_i) = (A(x_i), B(y_i))$. Of course, R^2 can be replaced or complemented by other measures. In this connection, it should also be mentioned that $A(x_i)$ and $B(y_i)$ might be related in a monotone though *nonlinear* way.² In such a case, a linear regression might lead to poor quality measures. Even though we restrict ourselves to the linear case in this paper, the method could clearly be extended in the direction of more general regression functions. For example, a straightforward (and easy to implement) generalization is to fit a polynomial of degree 2 to the data.

¹ The minimum operator is sometimes replaced by other combination operators.

² The Durbin-Watson test statistic is a useful indicator in this respect.

Note that simple formulae exist for the coefficients α, β in (5), e.g.

$$\beta = \frac{n_A \sum_{i=1}^{n_A} u_i v_i - \sum_{i=1}^{n_A} u_i \sum_{i=1}^{n_A} v_i}{n_A \sum_{i=1}^{n_A} u_i^2 - (\sum_{i=1}^{n_A} u_i)^2}, \quad (6)$$

$$\alpha = \bar{v}_A - \beta \bar{u}_A. \quad (7)$$

Let us anticipate a possible criticism of this derivation of regression coefficients: If the *marginal points* of the form $(A(x), B(y)) = (u, 0)$ are regarded as *censored* observations,³ simple linear regression techniques are actually not applicable and must be replaced by more sophisticated methods, such as Tobit regression models. Anyway, since our focus is on association rules rather than regression analysis, we shall not deepen this aspect further and rather proceed from (6–7) which yields at least good approximate results.

Generation of Rules. On the basis of the above information, two types of rules will be generated. The first type of rule, called *deviation rule* and denoted $A \rightarrow^d B$, expresses a (significant) deviation of the conditional mean. Suppose the points $(x, y) \in \mathcal{D}$ to be divided into two (fuzzy) samples: one for which $x \in A$ and one for which $x \notin A$. A point (x, y) belongs to the first sample, S_1 , with degree $A(x)$ and to the second sample, S_2 , with degree $1 - A(x)$. Let $\bar{v}_1 = \bar{v}_A$ and

$$\bar{v}_2 = \frac{\sum_{(x,y) \in \mathcal{D}} (1 - A(x)) B(y)}{\sum_{(x,y) \in \mathcal{D}} 1 - A(x)}$$

denote, respectively, the average of the membership degrees $B(y)$ in S_1 and S_2 . These averages can be considered as estimations of underlying parameters (expected values) ν_1 and ν_2 . Moreover, $\delta = \bar{v}_1 - \bar{v}_2$ is a simple point estimation of the deviation $\nu_1 - \nu_2$. Now, suppose $A \rightarrow^d B$ to be considered as interesting if $\nu_1 - \nu_2 > \Delta$, where Δ is a user-defined threshold. How can the “interestingness” of $A \rightarrow^d B$ on the basis of $\delta = \bar{v}_1 - \bar{v}_2$ be decided? Statistically speaking, the question is whether $\delta = \bar{v}_1 - \bar{v}_2$ is *significantly* larger than Δ . An appropriate decision principle is provided by the t-test adapted to the fuzzy case [7]:⁴

$$T = \frac{\bar{v}_1 - \bar{v}_2 - \Delta}{\sqrt{s_1^2/n_1 + s_2^2/n_2}}, \quad (8)$$

where $n_1 = |S_1| = \sum_{(x,y) \in \mathcal{D}} A(x)$, $n_2 = |S_2| = |\mathcal{D}| - n_1$, and s_1^2, s_2^2 denote, respectively, the variance of $B(y)$ for the two (fuzzy) samples. The deviation is considered to be significant at the .05 confidence level if $T > 1.645$.

Note that the denominator in (8) will generally be small (s_1^2, s_2^2 are upper-bounded by 1). In fact, it is not difficult to prove that $T > 1.645$ as soon as

$$\delta > \Delta^* \doteq \Delta + \frac{1.645}{\sqrt{|\mathcal{D}| \sigma(1 - \sigma)}}, \quad (9)$$

³ The membership of y in B cannot be negative.

⁴ This test compares the difference between the mean values of two *fuzzy* populations.

where σ is the support threshold. The right-hand side in (9) can be seen as a modified threshold that includes a “confidence offset”.

Once a deviation has been found to be significant, an adequate deviation rule can be defined on the basis of δ . This can be done by appending the corresponding averages to the rule, which is then of the form $\{A \rightarrow^d B [\bar{v}_1, \bar{v}_2]\}$. Another possibility is to present the rule in a linguistic form, paraphrasing the deviation δ by terms such as “slightly more”, “more”, or “much more”. In our example above, one would find the rule $\{\text{normal} \rightarrow^d \text{high} [.40, .08]\}$, which could be translated as follows: *If the weight is normal, then the performance is much higher than usual.*

Note that we have only tested for *positive* deviations. Of course, one could also represent negative deviations, using terms such as “less” or “much less” associated with values $\delta < 0$. However, this does again cause problems of redundancy: If B_1 and B_2 are complementary concepts in the sense that $B_1(y)$ and $B_2(y)$ are negatively correlated (such as low and high performance), then the positive deviation for B_1 will come along with the negative deviation for B_2 and vice versa. As in classical association analysis, we shall henceforth concentrate on positive deviations.

A second type of rule, called tendency rule and denoted $A \rightarrow^t B$, represents a gradual dependence between the concepts A and B . More precisely, it indicates that an increase in $A(x)$ comes along with an increase in $B(y)$. The validity of such a rule is judged on the basis of the regression coefficients (6–7) and the quality measure Q . For example, a simple decision principle is to reject a rule iff Q falls below a given threshold or the slope of the regression line, α , is too small: $Q < Q_{min}$ or $\alpha < \alpha_{min}$. If a rule is accepted, it might be presented in the form $\{A \rightarrow^t B [\alpha, \beta]\}$. Alternatively, a linguistic representation is possible: *The more x is in A , the more y is in B .* Again, this representation can be refined in dependence on the specific values of α and β . In our example above, the rule $\{\text{normal} \rightarrow^t \text{high} [0.65, -0.05]\}$ would be supported ($R^2 = 0.77$): *The more normal the weight, the higher the performance.*

4.3 Rules with Compound Conditions

So far, we have only considered simple rules involving two attributes. However, the approach outlined above easily extends to rules with a compound antecedent: Consider a rule of the form $A_1, \dots, A_m \rightarrow B$, where A_i is an element of the fuzzy partition of $\mathfrak{D}_{X_i}$, the domain of attribute X_i ($1 \leq i \leq m$), and B is an element of the fuzzy partition of variable Y . The antecedent of this rule stands for a conjunction of the conditions $x_i \in A_i$.

In fuzzy set theory, the logical conjunction is modeled by means of a so-called *t-norm*. This is a binary operator $\otimes : [0, 1] \times [0, 1] \rightarrow [0, 1]$ that is associative, commutative, non-decreasing in both arguments, and satisfies $\alpha \otimes 1 = 1 \otimes \alpha = \alpha$ for all $0 \leq \alpha \leq 1$. The most important t-norms are the minimum $(\alpha, \beta) \mapsto \min\{\alpha, \beta\}$, the product $(\alpha, \beta) \mapsto \alpha\beta$, and the Lukasiewicz t-norm $(\alpha, \beta) \mapsto \max\{\alpha + \beta - 1, 0\}$.

In the special case of an association $A \rightarrow B$ involving two variables X and Y , a value $A(x)$ corresponds to the degree to which $x \in A$ is satisfied (the fuzzy

truth degree of the proposition $x \in A$). In the more general case, this value is given by the conjunction $A_1(x_1) \otimes A_2(x_2) \otimes \dots \otimes A_m(x_m)$. Actually, this comes down to considering the attribute in the condition part as an m -dimensional variable $X = (X_1, \dots, X_m)$. As before, a rule can then be written in the form $A \rightarrow B$, where the fuzzy set A is defined as

$$A : \mathfrak{D}_{X_1} \times \dots \times \mathfrak{D}_{X_m} \rightarrow [0, 1], (x_1, \dots, x_m) \mapsto A_1(x_1) \otimes \dots \otimes A_m(x_m).$$

Again, one thus obtains a point $(u, v) = (A(x), B(y))$ for each transaction $(x, y) = (x_1, \dots, x_m, y) \in \mathcal{D}$ and, hence, a contingency diagram as introduced above. In other words, a rule with a compound condition part can be evaluated in the same way as a rule with a simple antecedent.

As concerns the problem of redundancy and interaction between association rules, it is important to mention that *none* of the following properties hold:

$$\begin{aligned} A_1 \rightarrow B \wedge A_2 \rightarrow B &\Rightarrow A_1, A_2 \rightarrow B \\ A_1, A_2 \rightarrow B &\Rightarrow A_1 \rightarrow B \vee A_2 \rightarrow B \end{aligned}$$

Still, some kind of pruning is clearly advisable. Especially, this concerns the relation between a rule $A \rightarrow B$ and its specializations $A^+ \rightarrow B$ with $A \subsetneq A^+$. For example, given the deviation rule $A \rightarrow B [\bar{v}_1, \bar{v}_2]$, a rule $A^+ \rightarrow B [\bar{v}_1^+, \bar{v}_2^+]$ will not be interesting if $\bar{v}_1^+ \leq \bar{v}_1$. More generally, one might adopt a *minimum improvement constraint* in order to eliminate unnecessarily complex rules [4].

4.4 Rule Mining and Computational Aspects

How does one find interesting instances of the two types of association rules introduced in this section? As already mentioned above, the first step is to find the frequent itemsets. To this end, any of the existing procedures can be used, for example the APRIORI algorithm (for quantitative attributes [9]). Note that an itemset is now a class of fuzzy sets $\{A_1, \dots, A_m\}$, where A_i is an element of the fuzzy partition of an attribute X_i (and $X_i \neq X_j$ for all $i \neq j$). The frequent itemsets determine the condition parts of the *candidate rules*.

In order to evaluate a candidate rule $A \rightarrow^t B$ one needs to compute the regression coefficients α and β as well as the quality measure Q . A look at (6–7) reveals that α and β can be derived by a single scan of the database. Afterwards, the quality measure Q (which is here taken as R^2) can be obtained, which makes one further scan necessary.

The evaluation of a rule $A \rightarrow^d B$ comes down to computing the deviation δ as well as the test statistic (8). This requires two scans of the database, since the computation of the variances s_1^2 and s_2^2 in (8) assumes the mean values $\bar{v}_1$ and $\bar{v}_2$ to be known. Therefore, these values have to be derived first. Alternatively, an approximate evaluation can be obtained on the basis of (9), which requires only a single scan.

In summary, it can be seen that the rule mining procedure is quite efficient. Apart from the search for frequent itemsets, it merely requires two additional scans of the database.

5 Concluding Remarks

We have introduced two types of quantitative association rules, referred to as deviation rules and tendency rules. The former type of rule is basically a fuzzy counterpart to the approach in [2]. The latter type of rule is able to represent gradual dependencies between attributes. This becomes possible by the use of fuzzy partitions for the attributes' domains.

Let us conclude with some remarks. (1) We have applied our approach to several data sets from the UCI repository for which we obtained rather promising results. These experimental studies are not reported here due to limited space; the technical report [7] provides a more detailed exposition. (2) So far, our approach assumes fixed underlying partitions comprised of preexisting “cognitive concepts”. On the one hand, this assumption appears especially reasonable in the fuzzy case. On the other hand, one cannot deny that the observation of data might also influence the formation of cognitive concepts. In our running example, for instance, a concept “ideal weight” (which coincides with our definition of “normal weight”) might well be established on the basis of the data. Extending the approach so as to support the discovery of such cognitive concepts is an interesting challenge for future work.

References

1. R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, pages 207–216, Washington, D. C., 1993. 200
2. Y. Aumann and Y. Lindell. A statistical theory for quantitative association rules. In *Proc. 5th ACM-SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1999. 200, 203, 211
3. S. D. Bay and M. J. Pazzani. Detecting group differences: Mining contrast sets. *Data Mining and Knowledge Discovery*, 5:213–246, 2001. 201
4. R. J. Bayardo, R. Agrawal, and D. Gunopulos. Constraint-based rule mining in large, dense databases. *Data Mining and Knowledge Discovery*, 4:217–240, 2000. 210
5. T. Fukuda, Y. Morimoto, S. Morishita, and T. Tokuyama. Mining optimized association rules for numeric attributes. In *Proc. 15th ACM Symposium of Principles of Database Systems*, 1996. 200
6. E. Hüllermeier. Implication-based fuzzy association rules. In L. De Raedt and A. Siebes, editors, *Proceedings PKDD-01, 5th European Conference on Principles and Practice of Knowledge Discovery in Databases*, number 2168 in LNAI, pages 241–252, Freiburg, Germany, September 2001. Springer-Verlag. 204
7. E. Hüllermeier. Fuzzy association rules. 21. Workshop *Interdisziplinäre Methoden der Informatik*, Universität Dortmund, 2001. 208, 211
8. W. Pedrycz. Data mining and fuzzy modeling. In *Proc. of the Biennial Conference of the NAFIPS*, pages 263–267, Berkeley, CA, 1996. 204
9. R. Skrikant and R. Agrawal. Mining quantitative association rules in large relational tables. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 1–12, 1996. 210
10. L. A. Zadeh. Fuzzy sets. *Information and Control*, 8:338–353, 1965. 204

Support Approximations Using Bonferroni-Type Inequalities

Szymon Jaroszewicz and Dan A. Simovici

University of Massachusetts at Boston, Department of Computer Science
Boston, Massachusetts 02125, USA
`{sj,dsim}@cs.umb.edu`

Abstract. The purpose of this paper is to examine the usability of Bonferroni-type combinatorial inequalities to estimation of support of itemsets as well as general Boolean expressions. Families of inequalities for various types of Boolean expressions are presented and evaluated experimentally.

Keywords: frequent itemsets, query support, Bonferroni inequalities

1 Introduction

In [MT96] a question has been raised of estimating supports of general Boolean expressions based on supports of frequent itemsets discovered by a datamining algorithm. The Maximum-Entropy approach to this estimation as well as some results, hypothesis and experiments on accuracy has been given in [PMS00, Man01, PMS01, Man02]. The accuracy of this estimation (using the inclusion-exclusion principle) is influenced by the supports of the frequent itemsets; when, for various reasons, some of these supports are missing this accuracy may be compromised. The problem has been addressed in [KLS96] but the results presented there can be applied only for the case when we know supports of all itemsets up to a given size. This is usually not the case with datamining algorithms which compute supports of only some of the itemsets of a given size.

A similar problem has been addressed in the area of statistical data protection, where it is important to assure that inferences about individual cases cannot be made from marginal totals (see [Dob01, BG99] for an overview). Those methods concentrate on obtaining the most accurate bounds possible (in order to rule out information disclosure), computational efficiency being a secondary concern. Algorithms usually involve repeated iterations over full contingency tables [BG99], branch and bound search [Dob01] or numerous applications of linear programming.

The approach we take in this paper is based on a family of combinatorial inequalities called Bonferroni inequalities [GS96]. In their original form the inequalities require that we know supports of all itemsets up to a given size. We address the problem by using the inequalities recursively to estimate supports of missing itemsets. The advantage of Bonferroni inequalities is that we can choose

an arbitrary limit on the size of the marginals involved, thus allowing for trading off accuracy for speed. Our experiments revealed that it is possible to obtain good bounds even if only marginals of small size are used.

A table is a triple $\tau = (T, H, \rho)$, where T is the name of the table, $H = \{A_1, \dots, A_n\}$ is the heading of the table and $\rho = \{t_1, \dots, t_m\}$ is a finite set of functions of the form $t_i : H \longrightarrow \bigcup_{A \in H} \text{Dom}(A)$ such that $t_i(A) \in \text{Dom}(A)$ for every $a \in A$. Following the relational database terminology we shall refer to these functions as H -tuples, or simply as tuples. If $\text{Dom}(A_i) = \{0, 1\}$ for $1 \leq i \leq n$, then τ is a binary table.

Let $\tau = (T, A_1 \cdots A_n, \rho)$ be a binary table.

An *itemset* of τ is an expression of the form $A_{i_1} \cdots A_{i_k}$. A *minterm* of τ is an expression of the form $A_{i_1}^{b_1} \cdots A_{i_k}^{b_k}$, where $b_i \in \{0, 1\}$ for $1 \leq i \leq k$ and

$$A^b = \begin{cases} A & \text{if } b = 1 \\ \bar{A} & \text{if } b = 0. \end{cases}$$

The support of a minterm $M = A_{i_1}^{b_1} \cdots A_{i_k}^{b_k}$ of a table $\tau = (T, H, \rho)$

$$\text{supp}(M) = \frac{|\{t \in \rho \mid t[A_{i_1} \cdots A_{i_k}] = (b_1, \dots, b_k)\}|}{|\rho|}.$$

Note that the support of minterms is actually a probability measure on the free Boolean algebra $\mathcal{Q}(H)$ generated by the attributes of the heading H . We refer to such polynomials as *queries*. The atoms of this algebra are the minterms and every boolean polynomial over the set H can be uniquely written as a disjunction of minterms. The least and the largest element of this Boolean algebra are denoted by $\emptyset$ and Ω , respectively.

Consider a table whose heading is $H = ABC$ and assume that the distribution of the values of the tuples in this table is given by:

A	B	C	Frequency
0	0	0	0
0	0	1	0
0	1	0	0.10
0	1	1	0.25
1	0	0	0.10
1	0	1	0.25
1	1	0	0.05
1	1	1	0.25

A run of the Apriori algorithm ([AMS⁺96]) on a dataset conforming to that distribution, with the minimum support of 0.35 will yield the following itemsets:

Itemset	Support
A	0.65
B	0.65
C	0.75
AC	0.50
BC	0.50

To estimate the unknown support of the itemset ABC we can use Bonferroni inequalities of the form:

$$\text{supp}(ABC) \geq 1 - \text{supp}(\bar{A}) - \text{supp}(\bar{B}) - \text{supp}(\bar{C}), \quad (1)$$

$$\begin{aligned} \text{supp}(ABC) \leq 1 - \text{supp}(\bar{A}) - \text{supp}(\bar{B}) - \text{supp}(\bar{C}) \\ + \text{supp}(\bar{A}\bar{B}) + \text{supp}(\bar{A}\bar{C}) + \text{supp}(\bar{B}\bar{C}). \end{aligned} \quad (2)$$

Note that since the support of AB is below the minimum support its value is not returned by the Apriori algorithm and this creates a problem for this estimation. All the itemset supports, except for $\text{supp}(\bar{A}\bar{B})$, in the previous expression can be determined from known itemset supports using inclusion-exclusion principle. For example, we have

$$\text{supp}(\bar{A}\bar{C}) = 1 - \text{supp}(A) - \text{supp}(C) + \text{supp}(AC) = 0.1.$$

Since all needed probabilities are known exactly, the lower bound (1) is easy to compute giving

$$\text{supp}(ABC) \geq 1 - 0.35 - 0.35 - 0.25 = 0.05.$$

To compute the upper bound we proceed as follows.

Since $\text{supp}(\bar{A}\bar{B})$ is not known, we apply Bonferroni inequalities recursively to get an upper bound for it. We have

$$\text{supp}(\bar{A}\bar{B}) = 1 - \text{supp}(A) - \text{supp}(B) + \text{supp}(AB),$$

and, since AB is not frequent, we know that its support is less than the 0.35 minimum support, giving

$$\text{supp}(\bar{A}\bar{B}) < 1 - \text{supp}(A) - \text{supp}(B) + \text{minsupp} = 0.05.$$

Substituting into (3) we get

$$\begin{aligned} \text{supp}(ABC) &< 1 - \text{supp}(\bar{A}) - \text{supp}(\bar{B}) - \text{supp}(\bar{C}) \\ &\quad + 0.05 + \text{supp}(\bar{A}\bar{C}) + \text{supp}(\bar{B}\bar{C}) \\ &= 1 - 0.35 - 0.35 - 0.25 + 0.05 + 0.1 + 0.1 = 0.3. \end{aligned}$$

Note that both bounds are not trivial since the lower bound is greater than 0, and the upper bound is less than the minimum support.

In the next section we present a systematic method for obtaining bounds for support of an itemset using supports of its subsets.

2 A Recursive Procedure for Computing Bonferroni Bounds from Frequent Itemsets

To obtain certain Bonferroni-type inequalities we use a technique known as the method of indicators [GS96]. For an event Q in the probability space define the

random variable I_Q called the indicator of Q as

$$I_Q = \begin{cases} 1 & \text{if } Q \text{ occurs} \\ 0 & \text{otherwise} \end{cases}$$

Then, for the expected value of I_Q we have $E(I_Q) = P(Q)$. For our specific probability space we have $E(I_Q) = \text{supp}(Q)$ for every query Q .

Let $Q_1, \dots, Q_m$ be m queries. Define the random variable J_k as

$$J_k = \sum \{I_{Q_{i_1}} \cdots I_{Q_{i_k}} \mid 1 \leq i_1 \leq \cdots \leq i_k \leq m\}.$$

Clearly, J_k takes as a value the number of k -conjunctions $Q_{i_1} \wedge \cdots \wedge Q_{i_k}$ that are satisfied. If ν_m is the number of queries that are satisfied among the m queries $Q_1, \dots, Q_m$, then we clearly have $J_k = \binom{\nu_m}{k}$, which implies

$$E\left(\binom{\nu_m}{k}\right) = E(J_k) = \sum \{\text{supp}(Q_{i_1} \wedge \cdots \wedge Q_{i_k}) \mid 1 \leq i_1 \leq \cdots \leq i_k \leq m\}.$$

See [GS96] for further details.

Using the indicator method we proved in [JSR02] the following general result:

Theorem 1. *Let $Q = I_1 \oplus I_2 \oplus \dots \oplus I_m$ be a boolean query, where $\oplus$ denotes the exclusive or operation, and $I_1, I_2, \dots, I_m$ are itemsets. The following inequalities hold for any natural number t :*

$$\begin{aligned} \sum_{k=1}^{2t} (-2)^{k-1} \sum_{i_1 < \dots < i_k} \text{supp}(I_{i_1} \wedge \dots \wedge I_{i_k}) \\ \leq \text{supp}(I_1 \oplus \dots \oplus I_m) \leq \\ \sum_{k=1}^{2t+1} (-2)^{k-1} \sum_{i_1 < \dots < i_k} \text{supp}(I_{i_1} \wedge \dots \wedge I_{i_k}). \end{aligned}$$

Note that since every query can be represented as an exclusive or of positive conjunctions, the above theorem allows us to obtain bounds for any boolean query expressed in terms of supports of positive conjunctions. However, these bounds are not always tight, and in fact, we showed in [JSR02] that for certain queries it is not possible to obtain tight bounds at all.

Since the Apriori algorithm only discovers supports of itemsets (as opposed to other types of queries), we need to express all inequalities in terms of supports of itemsets.

Theorem 2. Let $Q_1, \dots, Q_m$ be m queries in $\mathcal{Q}(H)$. The following inequalities hold for any $t \in \mathbb{N}$:

$$\begin{aligned} \sum_{k=0}^{2t+1} (-1)^k \sum_{r < i_1 < \dots < i_k \leq m} \text{supp}(Q_1 \dots Q_r Q_{i_1} \dots Q_{i_k}) \\ \leq \text{supp}(Q_1 \dots Q_r \bar{Q}_{r+1} \dots \bar{Q}_m) \leq \\ \sum_{k=0}^{2t} (-1)^k \sum_{r < i_1 < \dots < i_k \leq m} \text{supp}(Q_1 \dots Q_r Q_{i_1} \dots Q_{i_k}). \end{aligned}$$

Proof. By Rényi's Theorem [Rén58] it suffices to prove the claim for $Q_i \in \{\Omega, \emptyset\}$ for all $1 \leq i \leq m$, where Ω denotes the space of elementary events. When $Q_i = \emptyset$ for some $1 \leq i \leq r$, then both sides of the inequalities reduce to 0 and the result is immediate. For the case $Q_i = \Omega$ for all $1 \leq i \leq r$ we have $\text{supp}(Q_1 \dots Q_r \bar{Q}_{r+1} \dots \bar{Q}_m) = \text{supp}(\bar{Q}_{r+1} \dots \bar{Q}_m)$, and for all k and for all $r < i_1 < \dots < i_k \leq m$, $\text{supp}(Q_1 \dots Q_r Q_{i_1} \dots Q_{i_k}) = \text{supp}(Q_{i_1} \dots Q_{i_k})$. The result now follows from Bonferroni inequalities. $\square$

Corollary 1. Let $A_1 A_2 \dots A_r \bar{A}_{r+1} \bar{A}_{r+2} \dots \bar{A}_m$ be a minterm. The following inequalities hold for any natural number t :

$$\begin{aligned} \sum_{k=0}^{2t+1} (-1)^k \sum_{r < i_1 < \dots < i_k \leq m} \text{supp}(A_1 \dots A_r A_{i_1} \dots A_{i_k}) \\ \leq \text{supp}(A_1 \dots A_r \bar{A}_{r+1} \dots \bar{A}_m) \leq \\ \sum_{k=0}^{2t} (-1)^k \sum_{r < i_1 < \dots < i_k \leq m} \text{supp}(A_1 \dots A_r A_{i_1} \dots A_{i_k}) \end{aligned}$$

Proof. This statement follows immediately from Theorem 2. $\square$

Below we present results which form the basis of our algorithm for approximative computations of supports of itemsets. The binomial symbol $\binom{n}{k}$ will allow negative values of n , in which case its value is defined by the usual formula

$$\binom{n}{k} = \frac{n(n-1) \dots (n-k+1)}{k!}.$$

Lemma 1. For $m, k, h, s \in \mathbb{N}$ we have:

$$\sum_{k=0}^s (-1)^{s-k} \binom{m-k-1}{s-k} \binom{h}{k} = \binom{h-m+s}{s}.$$

Proof. We begin by showing that for every $a, b, c, d \in \mathbb{N}$ we have

$$\sum_{k=0}^a (-1)^k \binom{a-k}{b} \binom{c}{k-d} = (-1)^{a+b} \binom{c-b-1}{a-b-d}. \quad (3)$$

The proof is by induction on c . The basis step, $c = 0$, follows after elementary algebraic transformations. Suppose that the equality holds for numbers less than c . We have:

$$\begin{aligned}
& \sum_{k=0}^a (-1)^k \binom{a-k}{b} \binom{c}{k-d} \\
&= \sum_{k=0}^a (-1)^k \binom{a-k}{b} \binom{c-1}{k-d} + \sum_{k=0}^a (-1)^k \binom{a-k}{b} \binom{c-1}{k-d-1} \\
&= (-1)^{a+b} \binom{c-b-2}{a-b-d} + (-1)^{a+b} \binom{c-b-2}{a-b-d-1} \\
&\quad (\text{by the inductive hypothesis}) \\
&= (-1)^{a+b} \binom{c-b-1}{a-b-d}.
\end{aligned}$$

By using the complimentary combinations and Lemma 1 we can write:

$$\begin{aligned}
& \sum_{k=0}^s (-1)^{s-k} \binom{m-k-1}{s-k} \binom{h}{k} = \sum_{k=0}^s (-1)^{s+k} \binom{m-k-1}{m-s-1} \binom{h}{k} = \\
& (-1)^s \cdot \sum_{k=0}^s (-1)^k \binom{m-k-1}{m-s-1} \binom{h}{k} = (-1)^s \cdot (-1)^{2m-2-s} \binom{h-m+s}{s} \\
&= \binom{h-m+s}{s}.
\end{aligned}$$

□

Note that if $h = m$, the previous lemma implies

$$\sum_{k=0}^s (-1)^{s-k} \binom{m-k-1}{s-k} \binom{m}{k} = 1.$$

Our method of obtaining bounds is based on the following theorem

Theorem 3. *The following inequalities hold for any natural number t :*

$$\text{supp}(A_1 A_2 \dots A_m) \leq \sum_{k=0}^{2t} (-1)^k \binom{m-k-1}{2t-k} S_k \quad (4)$$

$$\text{supp}(A_1 A_2 \dots A_m) \geq \sum_{k=0}^{2t+1} (-1)^{k+1} \binom{m-k-1}{2t+1-k} S_k \quad (5)$$

where

$$S_k = \sum_{1 \leq i_1 < \dots < i_k \leq m} \text{supp}(A_{i_1} \dots A_{i_k}),$$

and $S_0 = 1$.

Proof. We use the method of indicators previously discussed.

Let ν_m be a random variable equal to the number of events $A_1, \dots, A_m$ that actually occur. By Lemma 1 we have:

$$\begin{aligned} \sum_{k=0}^s (-1)^{s-k} \binom{m-k-1}{s-k} \binom{\nu_m}{k} &= \binom{\nu_m - m + s}{s} \\ &= \begin{cases} 1 & \text{if } \nu_m = m \\ 0 & \text{if } \nu_m < m \text{ and } \nu_m \geq m - s \\ \binom{\nu_m - m + s}{s} & \text{if } \nu_m < m - s. \end{cases} \end{aligned}$$

By taking expectations of the above equation we get

$$\begin{aligned} \sum_{k=0}^s (-1)^{s-k} \binom{m-k-1}{s-k} S_k &= \text{supp}(\nu_m = m) \\ &+ \sum \left\{ \binom{\nu_m(\omega) - m + s}{s} \text{supp}(\omega) : \omega \in \Omega, \nu_m(\omega) < m - s \right\}, \end{aligned}$$

where Ω denotes the space of elementary events. Note that when $\nu_m < m - s$ the sign of $\binom{\nu_m - m + s}{s}$ is identical to that of $(-1)^s$. Replacing s by $2t$ or $2t+1$ yields the result. $\square$

3 The Estimation Algorithm

The main problem in using Bonferroni-type inequalities on collections of frequent itemsets is that some of the probabilities in the S_k sums are not known. We solved this problem by estimating the missing probabilities using Theorem 3. Given below is an algorithm that computes bounds on support of an itemset based on a collection of itemsets with known supports.

Algorithm 1.

Input: An itemset I , a natural number r , a collection $\mathcal{F}$ of itemsets, and their supports

Output: Bounds $L(I), U(I)$ on the support of I

The algorithm is implemented by functions L and U given below

Function $L(I, \mathcal{F}, r)$.

1. If $I \in \mathcal{F}$
2. return $\text{supp}(I)$
3. else
4. return $\max_{-1 \leq 2t+1 \leq r} \sum_{k=0}^{2t+1} S^L \left((-1)^{k+1} \binom{m-k-1}{2t+1-k}, k, I, \mathcal{F} \right)$

Function $U(I, \mathcal{F}, r)$.

1. If $I \in \mathcal{F}$
2. return $\text{supp}(I)$
3. else
4. $U \leftarrow \min_{0 \leq 2t \leq r} \sum_{k=0}^{2t} S^U \left((-1)^k \binom{m-k-1}{2t-k}, I, k, \mathcal{F} \right)$
5. $U \leftarrow \min\{U, \text{minsupp}, \min_{J \subset I} U(J)\}$
6. return U

The functions S^L and S^U are defined below

Function S^L (real coefficient c , itemset $I = A_1 A_2 \dots A_m$, $\mathcal{F}$, integer k)

1. If $k = 0$ return c
2. If $c \geq 0$
3. return $c \cdot \sum_{i_1 < \dots < i_k \leq m} L(A_{i_1} A_{i_2} \dots A_{i_k}, \mathcal{F}, k-1)$
4. else
5. return $c \cdot \sum_{i_1 < \dots < i_k \leq m} U(A_{i_1} A_{i_2} \dots A_{i_k}, \mathcal{F}, k-1)$

Function S^U (real coefficient c , itemset $I = A_1 A_2 \dots A_m$, $\mathcal{F}$, integer k)

1. If $k = 0$ return c
2. If $c \geq 0$
3. return $c \cdot \sum_{i_1 < \dots < i_k \leq m} U(A_{i_1} A_{i_2} \dots A_{i_k}, \mathcal{F}, k-1)$
4. else
5. return $c \cdot \sum_{i_1 < \dots < i_k \leq m} L(A_{i_1} A_{i_2} \dots A_{i_k}, \mathcal{F}, k-1)$

Of course, upper and lower bounds for itemsets are cached during computations to avoid repeated evaluations for the same itemset. The parameter r controls the maximum size of marginals (itemsets) used in the estimation.

The use of **minsupp** in step 5 of function U requires some comment. Including the value of **minsupp** in the minimum is possible only if we can determine that the estimated itemset I is not frequent. This can be done for example if $\mathcal{F}$ contains all frequent itemsets, or when $\mathcal{F}$ contains all frequent itemsets up to a given size k , and $|I| \leq k$. If we don't know whether I is frequent or not, we have to drop **minsupp** from the minimum.

For a fixed r the time required by the algorithm is a polynomial of degree r in the size of the itemset. For example, setting $r = 2$ results in time quadratic in the size of the itemset. In the next section we evaluate experimentally the computation time and examine the influence of parameter r on accuracy.

Table 1. Discovered vs. total frequent itemsets for the **mushroom** dataset

Itemset size	Min. support	18%	25%	30%	37%	43%	49%	55%	61%	73%
3	Frequent	1761	893	498	308	152	70	45	23	13
	Est. Freq.	345	244	179	127	86	54	34	19	10
	ratio (%)	19.59%	27.32%	35.94%	41.23%	56.58%	77.14%	75.56%	82.61%	76.92%
4	Frequent	4379	1769	795	368	147	48	29	16	6
	Est. Freq.	298	202	131	85	53	31	18	10	2
	ratio (%)	6.81%	11.42%	16.48%	23.10%	36.05%	64.58%	62.07%	62.50%	33.33%

4 Experimental Results

In this section we present experimental evaluation of the bounds. Our algorithm works best on dense datasets, which are more difficult to mine for frequent itemsets than sparse ones. However, the algorithm was tested on both dense and sparse data (IBM Quest data generator was used [AMS⁺96]) which is typically used for mining associations. The rest of the paper is focused on experiments performed mainly on dense databases, however some results on sparse data are also presented.

As dense databases we used the **mushroom** database from the UCI Machine Learning Archive [BM98] and census data of elderly people from the University of Massachusetts at Boston Gerontology Center available at <http://www.cs.umb.edu/~sj/datasets/census.arff.gz>. Since both datasets involve multivalued attributes, we replaced each attribute (including binary ones) with a number of Boolean attributes, one for each possible value of the original attribute. The **mushroom** dataset has 22 multivalued attributes and 8124 records. After binarization the multivalued attributes have been replaced by 128 binary attributes. The **census** data has 11 multivalued attributes (which were replaced by 29 binary ones) and 330 thousand rows. Those two datasets were chosen since they are widely available, and contain real-world data.

Before we present a detailed experimental study of the quality of bounds, we present the results of applying the bounds to a practical task. Suppose that we did not have enough time or computational resources to run the Apriori (or similar) algorithm completely, and we decided to stop the algorithm after finding frequent itemsets of size less than or equal to 2. We then use lower bounds to find frequent itemsets of size greater than 2. The experimental results for **mushroom** and **census** databases are shown in Tables 1 and 2 respectively.

The tables show, for various values of minimum support, the true number of frequent itemsets of sizes 3 and 4, the number of itemsets that we discovered to be frequent by using our bounds, and the ratio of the two numbers.

For large values of minimum support we are more likely to classify an itemset correctly than for smaller ones. The data shows that for itemsets with largest support the chances of actually being determined to be frequent without consulting the data can be as high as 80%. This tendency breaks down somewhat for extremely high values of minimum support (the last column in Table 1). The

Table 2. Ratios of discovered vs. total frequent itemsets for *census* data

Itemset size	Min. support	1%	2%	3%	5%	10%	15%	30%	50%
3	Frequent	1701	1377	1145	879	503	312	112	40
	Est. Freq.	154	149	146	137	108	90	47	21
	ratio (%)	9.05%	10.82%	12.75%	15.59%	21.47%	28.85%	41.96%	52.50%
4	Frequent	5050	3560	2728	1901	852	485	105	20
	Est. Freq.	103	98	94	85	64	48	18	3
	ratio (%)	2.04%	2.75%	3.45%	4.47%	7.51%	9.90%	17.14%	15.00%

Table 3. Ratios of discovered to total frequent itemsets, synthetic data

density	20%	30%	40%	50%	60%	70%
3-itemsets	0%	0%	2.5%	17.8%	36.6%	55.7%
4-itemsets	0%	0%	0%	2.5%	14.7%	33.5%
min. support	15%	15%	25%	25%	25%	25%

reason is that to exceed such a high value of minimum support a very tight lower bound is needed, which is not always obtainable.

Table 3 shows the ratios of frequent itemsets that were identified based on supports of their subsets for synthetic datasets of varying density (we used the synthetic market basket data generator from [AMS⁺96]). By density we mean the percentage of 1s in the table. The datasets had 50 attributes and 10000 transactions. Minimum support of 25% was used except for two cases when minimum support of 15% was necessary to get sufficiently large itemsets. The results show that the inequalities are useful only for datasets with density 50% and higher. This is however an especially important case since it is computationally very hard to obtain all frequent itemsets from such databases.

We now present an experimental analysis of the bounds obtained. The *trivial bounds* for the support of an itemset I are defined as follows. The trivial lower bound is 0; the trivial upper bound is the minimum of the upper bounds of the supports of all proper subsets of I and of the minimum support. As in the example above, here too we mine frequent itemsets with at most two items and compute bounds for larger ones.

Table 4 (a) contains the results for the *census* dataset with minimum support of 1.8%.

The parameter r in Algorithm 1 was chosen for each itemset I to be $|I| - 1$ for maximum accuracy. This causes an increase in estimation time for larger itemsets. Later in the section we present results showing that limiting the value of r can give very fast estimates with a very small impact on the quality of the bounds. All experiments were run on a 100MHz Pentium machine with 64MB of memory.

The bounds obtained are fairly accurate. The width of the interval between the lower and upper bounds varied from 0.048 to 0.019 for itemsets of size 3. Note that the estimates become more and more accurate for larger itemsets. The reason is that the bulk of large itemsets will have subsets whose support is very small, thus giving better average trivial bounds. Nontrivial upper bounds occur slightly more frequently than nontrivial lower bounds; however, lower bounds

Table 4. Results for the **census** dataset

itemset size	3	4	5	6
average interval width	0.0482797	0.0313103	0.0228579	0.0196316
average upper bound	0.0568679	0.0319395	0.0228771	0.0196316
average lower bound	0.00858817	0.000629199	1.925e-05	0
itemsets with nontrivial bounds	7.04%	0.59%	0.04%	0.00%
itemsets with nontrivial lower	4.06%	0.39%	0.02%	—
average lower improvement	0.211321	0.161151	0.0962518	—
itemsets with nontrivial upper	6.43%	0.47%	0.03%	—
average upper improvement	0.0225656	0.00983444	0.00262454	—
time [ms/itemset]	0.2	0.3	1	7

(a) 1.8% minimum support, all itemsets

itemset size	3	4	5	6
average interval width	0.102848	0.105024	0.106997	0.110767
average upper bound	0.127438	0.109572	0.107491	0.110767
average lower bound	0.0245896	0.00454846	0.00049354	0
itemsets with nontrivial bounds	20.17%	4.25%	0.58%	0.02%
itemsets with nontrivial lower	11.64%	2.82%	0.46%	—
average lower improvement	0.211321	0.161151	0.106164	—
itemsets with nontrivial upper	18.41%	3.43%	0.40%	0.02%
average upper improvement	0.0225656	0.00983444	0.00333985	0.00338427

(b) 1.8% minimum support, frequent itemsets only

itemset size	3	4	5	6
average interval width	0.171608	0.205194	0.222602	0.231362
average upper bound	0.235004	0.223174	0.225491	0.231362
average lower bound	0.0633963	0.0179804	0.00288882	0
itemsets with nontrivial bounds	48.55%	16.79%	3.40%	0.14%
itemsets with nontrivial lower	30.00%	11.16%	2.72%	—
average lower improvement	0.211321	0.161151	0.106164	—
itemsets with nontrivial upper	44.00%	13.56%	2.33%	0.14%
average upper improvement	0.0238776	0.00983444	0.00333985	0.00338427

(c) 9% minimum support, frequent itemsets only

give on average much better improvement over the trivial bounds (this is due to the fact that our trivial upper bounds are quite sophisticated, while the trivial lower bound is just assumed to be 0).

The percentage of itemsets having nontrivial bounds is quite small. However those itemsets who have high support (and thus are the most interesting) are more likely to get interesting nontrivial bounds. This can be seen in Tables 4(b) and 4(c), where up to 48% of itemsets have nontrivial bounds proving the usefulness of Theorem 3. Note that in this case the interval width increases with the size of the itemsets. This is due to the fact that for high supports we don't have large number of itemsets with low supports that would create trivial upper bounds. The conclusions were analogous for the **mushroom** database.

Table 5 shows how the choice of the argument r in Algorithm 1 influences the computation speed and the quality of the bounds. The results when r is set to the highest possible value (size of the estimated itemset minus one) is given in Table 4(a).

The results show that limiting the value of r to 2 or 3 gives a large speedup at a negligible decrease in accuracy. This is the approach we recommend. Also note that the proportion of itemsets with nontrivial bounds is higher for lower values of r . The same experiments repeated for frequent itemsets only yielded analogous results, so we omitted the data here.

Our last experimental result concerns estimating support of conjunctions allowing negated items using Corollary 1. Table 6 shows the results for the **census** dataset, with supports of all frequent 1- and 2-itemsets known (1.8% minimum support). In each of the itemsets exactly two of the items were negated. Again, the inequalities gave fairly tight bounds.

Table 5. Influence of the order of inequalities on the bounds

Census Data with 1.8% Minimum Support				
$r = 2$				
itemset size	3	4	5	6
average interval width	0.0482797	0.0315442	0.022993	0.0196671
average upper bound	0.0568679	0.0321734	0.0230122	0.0196671
average lower bound	0.00858817	0.000629199	1.925e-05	0
itemsets with nontrivial bounds	7%	1%	0.10%	0%
time [ms/itemset]	0.18	0.24	0.34	0.46
$r = 3$				
itemset size	3	4	5	6
average interval width	0.0482797	0.0313103	0.0228666	0.0196328
average upper bound	0.0568679	0.0319395	0.0228859	0.0196328
average lower bound	0.00858817	0.000629199	1.925e-05	0
itemsets with nontrivial bounds	7%	0.50%	0%	0%
time [ms/itemset]	0.18	0.3	0.53	0.92

Table 6. Estimates for itemsets with negations

Census Data with 1.8% Minimum Support					
itemset size	3	4	5	6	7
avg interval width	0.040498	0.081989	0.0668155	0.0392651	0.0180174
average upper bound	0.171319	0.120666	0.0685168	0.0392925	0.0180174
average lower bound	0.130821	0.0386768	0.00170127	2.73405e-05	0
time [ms/itemset]	0.24	0.46	0.96	2.54	5.12

5 Conclusions and Open Problems

We presented a method of obtaining bounds for support of database queries based on supports of frequent itemsets discovered by a datamining algorithm by generalizing the Bonferroni inequalities. Specialized bounds for estimating support of itemsets, itemsets with negated items, as well as bounds for arbitrary queries have been presented. An experimental evaluation of the bounds is given as well showing that the bounds are capable of providing useful approximations.

An interesting open problem is obtaining bounds for other specific types of queries. General inequalities in Theorem 1 can be used for this but the bounds they give are not always tight. It has also been shown in [JSR02] that for certain queries it is not possible to obtain any bounds at all. Nevertheless, we believe that it is possible to obtain useful bounds for a large family of practically useful queries.

Another open problem is obtaining bounds that directly use only available frequent itemsets of different sizes without estimating the S_k 's. An example is Conjecture 11 in [Man02].

It might also be useful to look at other, more sophisticated variants of Bonferroni inequalities (see [GS96]) and evaluate their relevancy for datamining.

References

- [AMS⁺96] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. Inkeri Verkamo. Fast discovery of association rules. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 307–328. AAAI Press, Menlo Park, 1996. 213, 220, 221
- [BG99] L. Buzzigoli and A. Giusti. An algorithm to calculate the lower and upper bounds of the elements of an array given its marginals. In *Statistical Data Protection (SDP'98)*, Eurostat, pages 131–147, Luxembourg, 1999. 212
- [BM98] C. L. Blake and C. J. Merz. *UCI Repository of machine learning databases*. University of California, Irvine, Dept. of Information and Computer Sciences, <http://www.ics.uci.edu/~mllearn/MLRepository.html>, 1998. 220
- [Dob01] A. Dobra. Computing sharp integer bounds for entries in contingency tables given a set of fixed marginals. Technical report, Department of Statistics, Carnegie Mellon University, <http://www.stat.cmu.edu/~adobra/bonf-two.pdf>, 2001. 212
- [GS96] J. Galambos and I. Simonelli. *Bonferroni-type Inequalities with Applications*. Springer, 1996. 212, 214, 215, 223
- [JSR02] S. Jaroszewicz, D. Simovici, and I. Rosenberg. An inclusion-exclusion result for boolean polynomials and its applications in data mining. In *Proceedings of the Discrete Mathematics in Data Mining Workshop, SIAM Datamining Conference*, Washington, D.C., 2002. 215, 223
- [KLS96] J. Kahn, N. Linial, and A. Samorodnitsky. Inclusion-exclusion: Exact and approximate. *Combinatorica*, 16:465–477, 1996. 212
- [Man01] H. Mannila. Combining discrete algorithms and probabilistic approaches in data mining. In L. DeRaedt and A. Siebes, editors, *Principles of Data Mining and Knowledge Discovery*, volume 2168 of *Lecture Notes in Artificial Intelligence*, page 493. Springer-Verlag, Berlin, 2001. 212
- [Man02] H. Mannila. Global and local methods in data mining: basic techniques and open problems. In *ICALP 2002, 29th International Colloquium on Automata, Languages, and Programming*, Malaga, Spain, June 2002. Springer-Verlag. 212, 223
- [MT96] H. Mannila and H. Toivonen. Multiple uses of frequent sets and condensed representations. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD'96)*, pages 189–194, Portland, Oregon, 1996. 212
- [PMS00] D. Pavlov, H. Mannila, and P. Smyth. Probabilistic models for query approximation with large sparse binary data sets. In *Proc. of Sixteenth Conference on Uncertainty in Artificial Intelligence (UAI-00)*, Stanford, 2000. 212
- [PMS01] D. Pavlov, H. Mannila, and P. Smyth. Beyond independence: Probabilistic models for query approximation on binary transaction data. ICS TR-01-09, University of California, Irvine, 2001. 212
- [Rén58] A. Rényi. Quelques remarques sur les probabilités des événements dépendants. *Journal de Mathématique*, 37:393–398, 1958. 216

Using Condensed Representations for Interactive Association Rule Mining

Baptiste Jeudy and Jean-François Boulicaut*

Institut National des Sciences Appliquées de Lyon
Laboratoire d'Ingénierie des Systèmes d'Information
Bâtiment Blaise Pascal, F-69621 Villeurbanne cedex, France
{Baptiste.Jeudy, Jean-Francois.Boulicaut}@lisi.insa-lyon.fr

Abstract. Association rule mining is a popular data mining task. It has an interactive and iterative nature, i.e., the user has to refine his mining queries until he is satisfied with the discovered patterns. To support such an interactive process, we propose to optimize sequences of queries by means of a cache that stores information from previous queries. Unlike related works, we use condensed representations like free and closed itemsets for both data mining and caching. This results in a much more efficient mining technique in highly correlated data and a much smaller cache than in previous approaches.

Keywords: association rule, inductive databases, knowledge cache

1 Introduction

An important data mining problem is the extraction of association rules [1]. It can be stated as follows in the context of basket analysis. The input is a transactional database where each row describes a transaction, i.e., a basket of items bought together by customers. If X is an itemset, i.e., a set of items, its frequency in the database, denoted as $\text{Freq}(X)$, is the number of rows/transactions where all items of X are true/present. An association rule is a pattern $X \Rightarrow Y$ where X and Y are itemsets, its frequency is the frequency of $X \cup Y$ and its confidence measures the conditional probability that a customer buys items from Y knowing that he bought items from X . The standard association rule mining problem concerns the computation of the frequency and confidence of all the association rules that satisfy user-defined constraints such as syntactical constraints, frequency and/or confidence constraints. These latter constraints are based on the so-called objective interestingness measures and specify that the measure (frequency, confidence) must be greater or equal to a user-defined threshold.

From the user point of view, association rule mining is an interactive and iterative process. The user defines a query by specifying various constraints on the rules he wants, e.g., the confidence must be more than 95% or the occurrence

* This research is part of the cInQ project (IST 2000-26469) that is partially funded by the European Commission IST Programme - Future and Emergent Technologies

of some items is mandatory. However, when a discovery process starts, it is difficult to figure out the collection of constraints that leads to an interesting result. The result of a data mining query is often unpredictable and the users have to produce sequences of queries until he gets an actionable collection of rules.

Computing the answer of a single association rule query is quite expensive and has motivated many research the last 5 years. It is known that the most expensive step for the standard association rule mining task is the computation of the frequent itemsets, or more generally the computation of the frequency of the interesting itemsets from which the rules will be derived. Indeed, deriving the rules needs for the frequencies of the itemsets to be able to compute the objective interestingness measures like confidence without any access to the raw data. “Pushing” the user-defined constraints can speed up this computation [16,18]. However, in highly correlated data, there are too many frequent itemsets and the task might be intractable. In this case, the use of condensed representations w.r.t. frequency queries [11] is useful. Several researchers have studied the efficient computation of condensed representations of (frequent) itemsets like the closed sets [17,4,20], the free sets [5] or the disjunct free sets [7].

Given a set $\mathcal{S}$ of pairs $(X, \text{Freq}(X))$, we consider that a condensed representation of $\mathcal{S}$ is a subset of $\mathcal{S}$ with two properties: (1) It is much smaller than $\mathcal{S}$ and faster to compute, and (2), the whole set $\mathcal{S}$ can be generated from the condensed representation with no access to the database, i.e., very efficiently. User-defined constraints can also be used to further optimize the computation of condensed representations [6,10].

However these techniques optimize only one single query. To support the optimization of sequences of queries, we can make use of the “similarities” between these queries that are often refinements of previous queries. It motivates the design of algorithms that try to use the results of already computed queries.

Contribution. In this paper, we propose an algorithm to mine interactively closed itemsets. The user defines constraints on the closed sets and can refine them in a sequence of queries. Our algorithm uses free sets as a cache to store information from the evaluation of previous queries. By using closed sets, our algorithm can be used in highly correlated data where approaches based on itemsets are not usable. Also, our cache of free itemsets is much smaller than a cache containing itemsets and our algorithm ensures that the intersection between the union of the results of all previous queries and the result of the new query is not recomputed. Finally, we do not make any assumption on the relation between two queries in the sequence, e.g., we do not require that the answer of one query is included in the answer of another. In our experiments, we show that this algorithm actually improves the performance of the extraction w.r.t. an algorithm that mines the closed sets without making use of the previous computations. The speedup is roughly equal to the relative size of the intersection between the answer to a new query and the content of the cache. We also show that the size of our cache is always smaller than a cache with itemsets and several orders of magnitude smaller in highly correlated data.

Related Work. Optimizing sequences of data mining queries by caching techniques has been studied for sequences [19] or association rules [9,2,15,8]. However none of these works makes use of condensed representations and the problem of the size of the stored information is not studied. In [15], experiments with different cache sizes are performed but no solution is given in the case of highly correlated data, i.e., when the number of frequent itemsets explodes. Also, most of these works require that some strong relation holds between the queries like inclusion or equivalence. In [14], a more general problem is addressed. A query consists of the constraints defining the subset of a relational database to be mined and the constraints on the association rules themselves. Any of these two sets of queries can be changed by the user. The authors show how to combine algorithms for incremental mining (i.e., when the database changes) with algorithms that only consider changes in the constraints on the association rules. As a result, it is not necessary to consider simultaneously changes on the mined database and on the association rule constraints. In this paper, we focus on changes on the constraints for the closed sets and we assume that the database does not change.

In Sect. 2, we provide some preliminary definitions and introduce the condensed representations based on closed sets and free sets. We propose in Sect. 3 an algorithm that computes constrained closed itemsets without a cache. It is extended in Sect. 4 to use a cache of free itemsets. Finally, we provide an experimental validation in Sect. 5 and Sect. 6 is a short conclusion.

2 Preliminary Definitions

Assume that **Items** is a finite set of symbols denoted by capital letters, e.g., $\mathbf{Items} = \{A, B, C, \dots\}$. A *transactional database* is a collection of rows where each row is a subset of **Items**. An *itemset* is a subset of **Items**. A row r *supports* an itemset S if $S \subseteq r$. The *support* (denoted $\text{support}(S)$) of an itemset S is the multi-set of all rows of the database that support S . The *frequency* of an itemset S is the cardinality of $\text{support}(S)$ and is denoted $\text{Freq}(S)$. Figure 1 provides an example of a transactional database and the supports and the frequencies of some itemsets. We use a string notation for itemsets, e.g., **AB** for $\{A, B\}$.

Definition 1 (Query). A *constraint* is a predicate from the power set of **Items** to $\{true, false\}$. A *query* is a pair (C, Db) where Db is a transactional database and C is a constraint. The result of a query $Q = (C, Db)$ is defined as the set $\text{SAT}(Q) = \{(S, \text{Freq}(S)), C(S) = true\}$.

		Itemset	Support	Frequency
$Db =$	r_1 ABCDE	A	$\{r_1, r_2, r_3\}$	3
	r_2 AB	D	$\{r_1, r_3, r_4\}$	3
	r_3 ABD	AD	$\{r_1, r_3\}$	2
	r_4 CD	ABCDE	$\{r_1\}$	1

Fig. 1. A four rows transactional database and some itemsets

A particular constraint is the minimal frequency constraint $\mathcal{C}_{\gamma\text{-freq}}$ when a frequency threshold γ is given: $\mathcal{C}_{\gamma\text{-freq}}(S) \equiv (\text{Freq}(S) \geq \gamma)$.

Example 1. Given the database Db defined in Fig. 1, let us consider the queries $Q_1 = (\mathcal{C}_{2\text{-freq}}, Db)$ and $Q_2 = (\mathcal{C}_{2\text{-freq}} \wedge \mathcal{C}_{\text{miss}}, Db)$ where $\mathcal{C}_{\text{miss}}(S) \equiv (B \notin S)$. The answers to these queries are: $\text{SAT}(Q_1) = \{(\emptyset, 4), (A, 3), (B, 3), (C, 2), (D, 3), (AB, 3), (AD, 2), (BD, 2), (CD, 2), (ABD, 2)\}$ and $\text{SAT}(Q_2) = \{(\emptyset, 4), (A, 3), (C, 2), (D, 3), (AD, 2), (CD, 2)\}$.

A classical result is that effective safe pruning can be achieved when considering anti-monotone constraints [12,16].

Definition 2 (Anti-monotonicity). *An anti-monotone constraint is a constraint $\mathcal{C}$ such that for all itemsets S, S' : $(S' \subseteq S \wedge \mathcal{C}(S)) \Rightarrow \mathcal{C}(S')$.*

The prototypical anti-monotone constraint is the frequency constraint. The constraint $\mathcal{C}_{\text{miss}}$ of Example 1 is another anti-monotone constraint and many other examples can be found, e.g., in [16]. Notice that the conjunction or the disjunction of anti-monotone constraints is anti-monotone.

2.1 Closed and Free Itemsets

The concept of closed set is classical within lattice theory and has been studied for association rule mining since the definition of the CLOSE algorithm in [17]. The collection of (frequent) closed itemsets is a useful condensed representation of the (frequent) itemsets in the case of highly correlated data [4].

Definition 3 (Closure, Closed Itemset). *The closure, denoted $cl(S)$, of an itemset S is the largest superset of S that has the same frequency than S . A closed itemset is an itemset S such that $S = cl(S)$.*

The closure operator has some useful properties that are straightforwardly derived from the definition.

Proposition 1.

- $S \subseteq cl(S)$.
- $cl(cl(S)) = cl(S)$.
- if $S \subseteq T$ then $cl(S) \subseteq cl(T)$.
- $\text{Freq}(S) = \text{Freq}(cl(S))$.

The second item of this Prop. 1 shows that the closure of an itemset is a closed itemset.

Definition 4 (Inherited Closure). *The inherited closure of an itemset S is*

$$i\_cl(S) = \bigcup_{T \subset S, |T|=|S|-1} cl(T) \setminus S.$$

The third item of Prop. 1 shows that the inherited closure of an itemset S is actually included in the closure of S . It follows from the first item of Prop. 1 that the disjoint union of S and its inherited closure is included in its closure. We can now define the proper closure of an itemset.

Definition 5 (Proper Closure). *The proper closure of an itemset S is:*

$$p\_cl(S) = cl(S) \setminus (i\_cl(S) \cup S).$$

The following proposition holds.

Proposition 2. *Let S be an itemset, the closure of S is the disjoint union of S , $i\_cl(S)$ and $p\_cl(S)$.*

Definition 6 (Free Itemset). *An itemset S is free if it is not included in the closure of any of its proper subsets.*

Indeed, it is sufficient to consider only the proper subsets of size $|S| - 1$. Let us illustrate these definitions on an example.

Example 2. Given the database of Fig. 1, $cl(A) = AB$ and $cl(C) = CD$. Therefore AC is free and $i\_cl(AC) = BD$. $p\_cl(AC) = E$ and $cl(AC) = ABCDE$. The closed itemsets are $\emptyset$, D , AB , CD , ABD and $ABCDE$. A condensed representation using closed sets of the answer of the query Q_1 of Example 1 is $\{(\emptyset, 4), (D, 3), (AB, 3), (CD, 2), (ABD, 2)\}$.

There is a strong relationship between closed and free itemsets: the set of closed itemsets is exactly the set of the closures of the free itemsets. This property is used in the following algorithms: To compute the closed sets, the algorithms mine the free itemsets and then output their closures. Let us note that free sets are special cases of δ -free sets [5] and have been independently formalized as key patterns in [3].

3 Mining Constrained Closed Itemsets

We propose an algorithm to mine closed itemsets under any anti-monotone constraint. This algorithm is an extension of the CLOSE algorithm described in [17] in which only the frequency constraint is considered.

Each itemset S is stored in a record also denoted S with four fields: $S.items$ is the list of the items in S , $S.i\_cl$ is the inherited closure, $S.p\_cl$ is the proper closure and $S.freq$ is the frequency of S . In the algorithm, we also use the macro $S.cl$ to denote $S.items \cup S.i\_cl \cup S.p\_cl$.

This is a level-wise algorithm: itemsets of size 0 are considered in the first iteration, then those of size 1, ... At each iteration, the set of candidate itemsets (Cand) is filtered to remove those that do not satisfy the constraint $\mathcal{C}_{am}$ (Step 3). Then a scan on the transactional database is performed to compute the proper closure and the frequency of each itemset in Cand. The candidate itemsets that are not frequent are removed (Step 5) and the closure of the frequent ones

are output (Step 6). Then, the candidates for the next iteration are computed using the procedure `cand_gen`. As in the APRIORI algorithm [1], a candidate is generated by joining two itemsets of size k that share the same $k - 1$ first items in lexicographic order (e.g., joining **ABC** and **ABD** produces **ABCD**). This procedure also initializes the inherited closure of each new candidate itemset according to Definition 4. Finally, the new candidate itemsets that are not free are removed (Step 8).

Algorithm 1

Input: A query $Q = (\mathcal{C}_{\gamma\text{-freq}} \wedge \mathcal{C}_{am}, Db)$ where $\mathcal{C}_{am}$ is an anti-monotone constraint.

Output: $\mathcal{O} = \{(\text{cl}(S), \text{Freq}(S)), S \text{ is free and } \mathcal{C}_{\gamma\text{-freq}}(S) \wedge \mathcal{C}_{am}(S) \text{ is true}\}$. By construction, $\mathcal{O}$ is a condensed representation of $\text{SAT}(Q)$

```

1  Cand :=  $\{(\emptyset, \emptyset, \emptyset, 0)\}$ 
2  while Cand  $\neq \emptyset$  do
3      Cand :=  $\{S \in \text{Cand}, \mathcal{C}_{am}(S.\text{items}) = \text{true}\}$ 
4      DB_pass(Cand, Db)
5      Cand :=  $\{S \in \text{Cand}, S.\text{freq} \geq \gamma\}$ 
6      Output( $\{(S.\text{cl}, S.\text{freq}), S \in \text{Cand}\}$ )
7      Cand := cand_gen(Cand)
8      Cand :=  $\{S \in \text{Cand}, S \text{ is free}\}$ 
9  od
```

An advantage of this algorithm is that it makes an active use of the anti-monotone constraint $\mathcal{C}_{am}$ to prune the search space (and not only the frequency constraint). In previous papers [6,10], we studied how to push monotone constraints (a monotone constraint is the negation of an anti-monotone one). However, dealing with sequences of queries when monotone constraints are pushed is still under progress.

Algorithm 1 computes the free itemsets that satisfy the constraint $\mathcal{C}_{\gamma\text{-freq}} \wedge \mathcal{C}_{am}$ and then output their closures. It is therefore possible that some of these closures do not satisfy the constraint $\mathcal{C}_{am}$ (but they satisfy $\mathcal{C}_{\gamma\text{-freq}}$ by the fourth item of Prop. 1).

Let us now give an algorithm that generates $\text{SAT}(Q)$ from this condensed representation.

Regeneration Algorithm

Input: The output $\mathcal{O}$ of Alg. 1 and the constraint $\mathcal{C}_{am}$.

Output: The answer $\text{SAT}(Q)$ to query $Q = (\mathcal{C}_{\gamma\text{-freq}} \wedge \mathcal{C}_{am}, Db)$.

```

1   $\mathcal{I} := \{(S, \text{Freq}(S)), \exists C \in \mathcal{O} \text{ s.t. } S \subseteq C\}$ 
2  Output( $\{(S, \text{Freq}(S)), S \in \mathcal{I} \text{ and } \mathcal{C}_{am}(S) = \text{true}\}$ )
```

Details about Step 1 can be found in previous works on closed itemsets (see, e.g., [17]) and are not provided here.

Once we know $\text{SAT}(Q)$, it is possible to derive association rules by testing (w.r.t. minimal confidence) the rules that can be built from the subsets of each set S from the answer $\text{SAT}(Q)$. Notice that an alternative would be to generate non-redundant association rules directly from the output of Alg. 1 [20].

In this process, the expensive part is Alg. 1 and we now discuss its optimization. Indeed, the computation of the answer to query Q by the regeneration algorithm and the generation of association rules do not require further access to the database such that they can be performed efficiently.

4 Caching Free Itemsets

Let us consider a new algorithm that stores information from previous extractions in a *cache* to speed up new extractions using different constraints. First, we describe the structure of this cache and its contents.

In Alg. 1 Line 4, the proper closure and the frequency of each candidate free itemset is computed during a database scan. If this information has been already computed for a previous query, it would be interesting to store it and reuse it. Therefore, we use a cache of free itemsets S with the information computed during the database scan, i.e. $\text{p\_cl}(S)$ and $\text{Freq}(S)$ ¹. The cache is simply a set of records of the form $(S.\text{items}, S.\text{p\_cl}, S.\text{freq})$.

We require that the cache is downward closed. It means that if a free itemset S is in the cache, then every subset of S that is free is also in the cache. This guarantees that if a free itemset is not in the cache then none of its super-sets is in the cache. This property is used to speed up the search of an itemset in the cache.

This cache has been implemented using a prefix tree to store the free itemsets. With this structure, the complexity of the search of an itemset in the cache is proportional to the size of the itemset and not to the size of the cache.

4.1 Algorithm 2

The proof of the completeness and soundness of Alg. 2 is not provided due to the lack of space. However, it might appear clear for a reader familiar with the use of level-wise algorithms that compute closed itemsets.

A new boolean field $S.\text{in\_cache}$ is added to the record representing each itemset. This field is used to know if the itemset is in the cache.

The difference between Alg. 1 and Alg. 2 is that the latter uses a cache. During Step 5, the flag $S.\text{in\_cache}$ is checked for every itemset in Cand . If it is false, then the itemset cannot be in the cache. If it is true, then the itemset is searched in the cache. If it is found, $S.\text{freq}$ and $S.\text{p\_cl}$ are updated, else $S.\text{in\_cache}$ is set to false. In Step 6, the frequency and proper closure of the itemsets that were not found in the cache (i.e., $S.\text{in\_cache} = \text{false}$) are computed during a database

¹ It is possible to put closed sets in the cache but some computations would be needed to generate $\text{p\_cl}(S)$ and $\text{Freq}(S)$

scan and inserted in the new cache (Step 9). During the candidate generation (Step 10), the field $S.in\_cache$ of every new candidate is initialized with the conjunction of $T.in\_cache$ for every subset T of S such that $|T| = |S| - 1$. This ensures that this field is false if and only if a subset of S is not in the cache. Since the cache is downward closed, this would mean that S cannot be in the cache.

Algorithm 2

Input: A query $Q = (\mathcal{C}_{\gamma-freq} \wedge \mathcal{C}_{am}, Db)$ where $\mathcal{C}_{am}$ is an anti-monotone constraint and a cache C .

Output: The collection $\{(cl(S), Freq(S)), S \text{ is free and } \mathcal{C}_{am}(S) \text{ is true.}\}$ and a new cache C_{new} .

```

1  Cand :=  $\{(\emptyset, \emptyset, \emptyset, 0, true)\}$ 
2   $C_{new} := C$ 
3  while Cand  $\neq \emptyset$  do
4    Cand :=  $\{S \in \text{Cand}, \mathcal{C}_{am}(S.items) = true\}$ 
5    Cache_pass(Cand, C)
6    DB_pass2(Cand, Db)
7    Insert_in_Cache(Cand,  $C_{new}$ )
8    Cand :=  $\{S \in \text{Cand}, S.freq \geq \gamma\}$ 
9    Output( $\{(S.cl, S.freq), S \in \text{Cand}\}$ )
10   Cand := cand_gen2(Cand)
11   Cand :=  $\{S \in \text{Cand}, S \text{ is free}\}$ 
12 od
```

Algorithm 2 does not make any assumption on the content of the cache except that it is downward closed. This means that it can deal with sequences of queries where no strict relation of inclusion holds between the queries, e.g., it can use results from the computation of the query $Q_3 = (\mathcal{C}_{0.1-freq}, Db)$ to speed up the computation of $Q_4 = (\mathcal{C}_{0.05-freq} \wedge \mathcal{C}, Db)$ where $\mathcal{C}(S) \equiv (S \cap AB = \emptyset)$ even though there is no containment relation between the results of these two queries.

We can formally characterize the content of the cache. Assuming that we already performed the extractions for queries $Q_1 = (\mathcal{C}_1, Db)$, $Q_2 = (\mathcal{C}_2, Db)$, $\dots$, $Q_n = (\mathcal{C}_n, Db)$, then the cache stores information on the frequency and the proper closures of all free itemsets manipulated during these extractions.

In [12], it is shown that in the case of APRIORI, the set of itemsets whose frequency is computed is the set of frequent itemsets plus its negative border, i.e., the set of minimal (w.r.t. the set inclusion) infrequent itemsets. This can be generalized in our framework.

Proposition 3. *Assuming that we already performed the extractions for queries $Q_1 = (\mathcal{C}_1, Db)$, $Q_2 = (\mathcal{C}_2, Db)$, $\dots$, $Q_n = (\mathcal{C}_n, Db)$, the cache contains the frequencies and proper closures of the free itemsets of $SAT(Q_1) \cup SAT(Q_2) \dots \cup SAT(Q_n)$ plus the minimal free itemsets that do not satisfy $\mathcal{C}_1 \wedge \mathcal{C}_2 \dots \wedge \mathcal{C}_n$.*

This property describes which information is stored in the cache at a given point. However, this property is not necessary for the completeness or the soundness of Alg. 2 (see next subsection).

4.2 Caching Strategies

In [15], several caching strategies are presented. The strategy that we use is similar to the No Replacement (NR) strategy of [15] (except that our cache is stored in a prefix tree), i.e. itemsets are added in the cache and never removed. This is motivated by the fact that our cache of free itemsets is much smaller than a cache of itemsets and therefore less likely to become full (see Sect. 5).

In the following, we discuss how to adapt the other strategies from [15] to our cache.

The Simple Replacement (SR) strategy uses the fact that it is more valuable to store in the cache the itemsets with the largest frequency because they are more likely to be used in subsequent queries. Thus, when the cache is full, the itemsets with the smallest frequency are removed to store new itemsets. This strategy is easily adaptable to our framework. Removing the free itemsets with the smallest frequency from our cache does not break the downward closure property (because the frequency is a decreasing function w.r.t. the set inclusion). Of course, in this case, Prop. 3 no longer holds.

The Benefit Replacement (BR) strategy from [15] was pointed out as the most efficient. The authors propose to store in the cache a *gsup* value for every k such that every itemset of size k whose frequency is above *gsup* is guaranteed to be in the cache. This can dramatically improve the performance if the new query has a frequency threshold above *gsup*: the algorithm just has to scan the cache to answer the query (thus saving the candidate generation steps). The main problem of this strategy is to compute *gsup*. If queries with only a frequency constraints are used, it is straightforward. With more complex queries [15] gives no solution.

However, it is possible to extend this BR strategy. Let $Q_1 = (\mathcal{C}_1, Db)$, $\dots$, $Q_n = (\mathcal{C}_n, Db)$ be a sequence of queries and $Q = (\mathcal{C}, Db)$ be the new query. If the implication $\mathcal{C}_1 \wedge \dots \mathcal{C}_n \Rightarrow \mathcal{C}$ holds, then it means that the answer to query Q is in the cache and it is possible to answer it by scanning the cache once.

This strategy shows that it would be quite valuable to combine caching techniques with algorithms that can find such implications in the queries.

5 Experimentations

In this section, we use a relative frequency instead of an absolute frequency: the relative frequency is the absolute frequency divided by the number of rows in the database.

The algorithms have been implemented in Ocaml². All the experiments were conducted on a PC under Linux operating system with an AMD Duron 700Mhz

² Developed at INRIA <http://caml.inria.fr/ocaml/index.html>

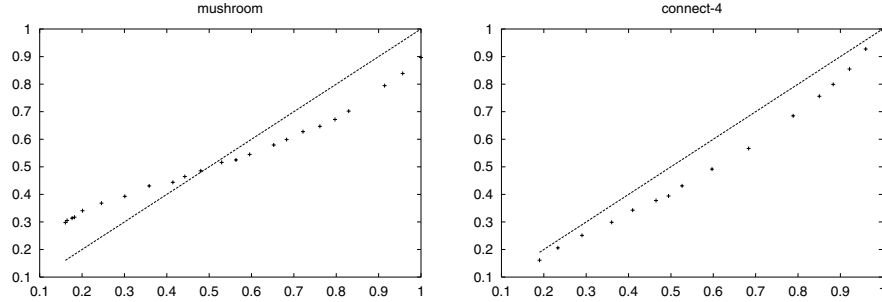


Fig. 2. The speedup versus the hit rate for the `mushroom` data set (left) and the `connect-4` data set (right)

processor and 384Mb of memory. We used two common datasets for our experiments, `connect-4` and `mushroom` from the QUEST project³. The main particularity of these data sets is that they are dense and highly correlated.

In the first experiment, we study the efficiency when using the cache. When half of the data needed by the algorithm is stored in the cache, we can expect that the computation time is half of the computation without a cache. To evaluate this efficiency, we performed an extraction to build a cache C with a query Q . Then we considered n queries $Q_1, Q_2, \dots, Q_n$. For each query Q_i , we performed two extractions, one using the cache C (duration dc_i) and another with no cache (duration dnc_i). We denote t_i the total number of candidates that are searched in the cache during Step 5 of Alg. 2 and f_i the number of them that are in the cache. We define the speedup as $1 - dc_i/dnc_i$ and the hit rate as $1 - f_i/t_i$. Figure 2 represents the hit rate versus the speedup for the two data sets `mushroom` and `connect-4`. It shows that the cache is used efficiently and that using a cache can bring about a significant speedup. In the `mushroom` data set, we can even notice that the speedup is above the hit rate for low hit rates.

Next, we made several tests to verify our claim about the fact that the complexity of the search in the cache does not depend on the size of the cache. For this, we built two caches C_1 and C_2 , C_1 was twenty times larger than C_2 . Then we performed two extractions with a query Q such that itemsets used by this query were present either in both caches or in none of them. The first extraction was made using cache C_1 and the second with C_2 . There was no significant differences between the two extractions, showing that the size of the cache has no significant impact on the performance.

Finally, we compared the size of our cache of free itemsets with the knowledge cache of [15] that uses frequent itemsets (strategy NR). For each free itemset S in our cache, we count one unit of storage for each item in S plus one unit for each item in $i\_cl(S)$ plus one unit to store the frequency, thus the total size of our cache is $\sum_{S \in C} (|S| + |i\_cl(S)| + 1)$. For a cache C' of “classical” itemsets, we

³ <http://www.almaden.ibm.com/cs/quest/>

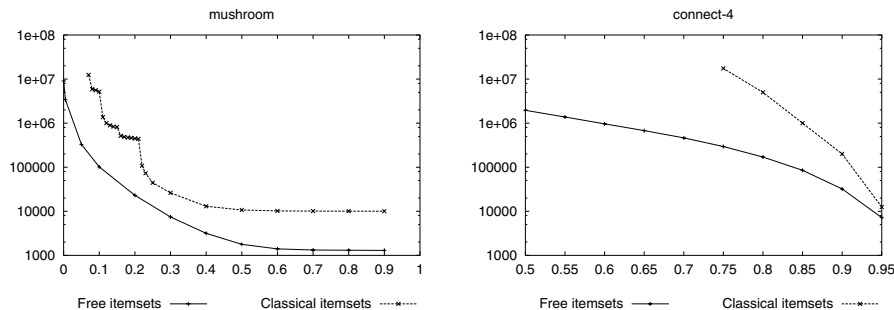


Fig. 3. Size of our cache (free sets) and the itemset cache versus the frequency threshold in the `mushroom` dataset (left) and the `connect-4` data set (right)

count one unit for each item of each itemset plus one unit for the frequency, the total size is therefore $\sum_{S \in C'} (|S| + 1)$. With these definitions, we can prove that our cache is *always* smaller than a cache using itemsets. Figure 3 shows that in practical experiments, the actual difference is up to several orders of magnitude.

6 Conclusion

In this work, we extended the CLOSE algorithm to deal efficiently with a sequence of queries using anti-monotone constraints. To achieve this, we demonstrate the added-value of condensed representations as a knowledge cache for interactive association rule mining.

This work has two major advantages versus previous works on using caches like [15]. First, the use of condensed representations allows mining in highly correlated data where other techniques are not tractable. Second, using these condensed representations leads to a cache that is orders of magnitude smaller than a traditional cache of frequent itemsets.

This cache enables an efficient evaluation of sequences of association rule mining queries and such a technique might be implemented, e.g., within the MINE RULE operator [13]. Another perspective of this work is to consider conjunctions of anti-monotone and monotone constraints and study in depth the optimizations in that wider framework.

References

1. Rakesh Agrawal, Heikki Mannila, Ramakrishnan Srikant, Hannu Toivonen, and A. Inkeri Verkamo. Fast discovery of association rules. In *Advances in Knowledge Discovery and Data Mining*, pages 307–328. AAAI Press, Menlo Park, CA, 1996. 225, 230
2. Elena Baralis and Giuseppe Psaila. Incremental refinement of mining queries. In *Proc. DaWaK'99*, volume 1676 of *LNCS*, pages 173–182. Springer-Verlag, 1999. 227

3. Yves Bastide, Rafik Taouil, Nicolas Pasquier, Gerd Stumme, and Lotfi Lakhal. Mining frequent patterns with counting inference. *SIGKDD Explorations*, 2(2):66–75, December 2000. [229](#)
4. Jean-François Boulicaut and Artur Bykowski. Frequent closures as a concise representation for binary data mining. In *Proc. PAKDD'00*, volume 1805 of *LNAI*, pages 62–73, Kyoto, JP, April 2000. Springer-Verlag. [226](#), [228](#)
5. Jean-François Boulicaut, Artur Bykowski, and Christophe Rigotti. Approximation of frequency queries by means of free-sets. In *Proc. PKDD'00*, volume 1910 of *LNAI*, pages 75–85, Lyon, F, September 2000. Springer-Verlag. [226](#), [229](#)
6. Jean-François Boulicaut and Baptiste Jeudy. Mining free-sets under constraints. In *Proc. IDEAS'01*, pages 322–329, Grenoble, F, July 2001. IEEE Computer Society. [226](#), [230](#)
7. Artur Bykowski and Christophe Rigotti. A condensed representation to find frequent patterns. In *Proc. PODS'01*, pages 267–273, Santa Barbara, California, USA, May 2001. ACM Press. [226](#)
8. Cheikh T. Diop, Arnaud Giacometti, Dominique Laurent, and Nicolas Spyrtatos. Composition of mining contexts for efficient extraction of association rules. In *Proc. EDBT'02*, Praha, CZ, March 2002. Springer-Verlag. To appear. [227](#)
9. Bart Goethals and Jan van den Bussche. On implementing interactive association rule mining. In *Proc. DMKD'99*, Philadelphia, USA, May 1999. [227](#)
10. Baptiste Jeudy and Jean-François Boulicaut. Optimization of association rule mining queries. *Intelligent Data Analysis, IOS Press*, 6(5), 2002. To appear. [226](#), [230](#)
11. Heikki Mannila and Hannu Toivonen. Multiple uses of frequent sets and condensed representations. In *Proc. SIGKDD'96*, pages 189–194, Portland, USA, August 1996. AAAI Press. [226](#)
12. Heikki Mannila and Hannu Toivonen. Levelwise search and borders of theories in knowledge discovery. *Data Mining and Knowledge Discovery*, 1(3):241–258, 1997. [228](#), [232](#)
13. Rosa Meo, Giuseppe Psaila, and Stefano Ceri. An extension of SQL for mining association rules. *Data Mining and Knowledge Discovery*, 2(2):195–224, 1998. [235](#)
14. Tadeusz Morzy, Marek Wojciechowski, and Maciej Zakrzewicz. Materialized data mining views. In *Proc. PKDD'00*, volume 1910 of *LNAI*, pages 65–74, Lyon, F, September 2000. Springer-Verlag. [227](#)
15. Biswadeep Nag, Prasad M. Deshpande, and David J. DeWitt. Using a knowledge cache for interactive discovery of association rules. In *Proc. SIGKDD'99*, pages 244–253. ACM Press, 1999. [227](#), [233](#), [234](#), [235](#)
16. Raymond Ng, Laks V. S. Lakshmanan, Jiawei Han, and Alex Pang. Exploratory mining and pruning optimizations of constrained associations rules. In *Proc. SIGMOD'98*, pages 13–24, Seattle, Washington, USA, 1998. ACM Press. [226](#), [228](#)
17. Nicolas Pasquier, Yves Bastide, Rafik Taouil, and Lotfi Lakhal. Efficient mining of association rules using closed itemset lattices. *Information Systems*, 24(1):25–46, January 1999. [226](#), [228](#), [229](#), [230](#)
18. Ramakrishnan Srikant, Quoc Vu, and Rakesh Agrawal. Mining association rules with item constraints. In *Proc. SIGKDD'97*, pages 67–73, Newport Beach, California, USA, 1997. AAAI Press. [226](#)
19. Marek Wojciechowski. Interactive constraint-based sequential pattern mining. In *Proc. ADBIS'01*, volume 2151 of *LNCS*, pages 169–181, Vilnius, Lithuania, September 2001. Springer-Verlag. [227](#)
20. Mohammed Javeed Zaki. Generating non-redundant association rules. In *Proc. SIGKDD'00*, pages 34–43, Boston, USA, August 2000. AAAI Press. [226](#), [231](#)

Predicting Rare Classes: Comparing Two-Phase Rule Induction to Cost-Sensitive Boosting

Mahesh V. Joshi¹, Ramesh C. Agarwal², and Vipin Kumar³

¹ IBM T. J. Watson Research Center
P.O. Box 704, Yorktown Heights, NY 10598, USA
joshim@us.ibm.com

² IBM Almaden Research Center
650 Harry Road, San Jose, CA 95120, USA
ragarwal@us.ibm.com

³ Dept. of Computer Science and AHPCRC, University of Minnesota
1100 S. Washington Ave, Minneapolis, MN 55415, USA
kumar@cs.umn.edu

Abstract. Learning good classifier models of rare events is a challenging task. On such problems, the recently proposed two-phase rule induction algorithm, PNRule, outperforms other non-meta methods of rule induction. Boosting is a strong meta-classifier approach, and has been shown to be adaptable to skewed class distributions. PNRule's key feature is to identify the relevant false positives and to collectively remove them. In this paper, we qualitatively argue that this ability is not guaranteed by the boosting methodology. We simulate learning scenarios of varying difficulty to demonstrate that this fundamental qualitative difference in the two mechanisms results in existence of many scenarios in which PNRule achieves comparable or significantly better performance than AdaCost, a strong cost-sensitive boosting algorithm. Even a comparable performance by PNRule is desirable because it yields a more easily interpretable model over an ensemble of models generated by boosting. We also show similar supporting results on real-world and benchmark datasets.

1 Introduction and Motivation

In many domains such as fraud detection, network intrusion detection, text categorization, and web mining, it is becoming critical to be able to learn predictive, high precision models for some important events that occur very rarely. In most of these domains, the data is available in a labeled form enabling use of classification methods. We focus on the binary classification problem in this paper. The goal is to build a model for distinguishing one rare class from the rest.

Some work has started to emerge to solve this problem [1,2,3]. In the context of rare classes, we take a stand similar to the Information Retrieval community, that a meaningful evaluation metric should reflect a balance between the recall and precision of the given rare class¹. From this recall-precision perspective,

¹ If a classifier detects m examples to be of class C , out of which l indeed belong to C , then its precision (P) for class C is l/m . If C has total of n examples in the set, then

a recently proposed two-phase rule-induction algorithm PNrule, was shown to outperform single-phase methods such as RIPPER and C4.5rules [3], because of its way of decoupling the recall and precision objectives.

In the past few years, boosting has emerged as a competitive technique. Various boosting algorithms have been proposed in the literature [5,6,7,8]. From the traditional goal of accuracy, theoretical analysis [5] has been conducted to show that boosting can always improve accuracy as the iterations progress, as long as its base learner satisfies the weak learnability criteria of the PAC theory². However, no such theoretical analysis has been done to see if boosting can always improve the recall-precision based performance that we desire for the rare classes. Recently, we compared the effect of weight update mechanisms on various boosting algorithms [9] for the rare class prediction problem. AdaCost [8], a cost-sensitive algorithm emerged as a strong algorithm as compared to all others because of its ability to implicitly emphasize both recall and precision.

Boosting results in an ensemble of models, hence we refer to it as a meta-technique as opposed to the non-meta techniques that generate a single model. In this paper, we take the case of the two strong algorithms from each category; viz. PNrule and AdaCost, and make an attempt to see if one key feature of PNrule is implicitly present in boosting. We present detailed qualitative analysis to argue that boosting does not guarantee an effect similar to this feature over its iterations. This fundamental difference in the two algorithms leads us to expect existence of many scenarios in which PNrule will be comparable to or significantly better than boosting from the recall-precision perspective. Even if PNrule performance is comparable, it is preferable in many domains such as document categorization or network intrusion detection where easy interpretability of the model is highly desirable.

We now briefly illustrate the feature of PNrule via an example. Let us assume that we are building a model for a network intrusion attack of type **remote-to-local** (r2l). It can be distinguished via rules based on attributes such as **protocol_type** (tcp, udp, etc), **number_of_logins**, and **service_type** (ftp, http, etc). But, some such rules, e.g. **service_type = http**, may also capture many false positives of the **denial-of-service** (dos) attack. In order to increase precision of r2l's model, these false positives of dos must be removed by learning rules such as **duration_of_connection < 2 seconds** $\rightarrow$ dos or **bytes_transferred < 200** $\rightarrow$ dos. Now, these rules can be learned in two ways: implicitly or explicitly. The implicit way is to add to each r2l rule the conjunctive conditions that predict *absence* of dos. The single-phase algorithms (e.g. RIPPER [10]) do an implicit learning. An explicit approach gathers all the examples covered by at least one of the r2l rules, and then explicitly learns rules for the *presence* of dos from this collection. PNrule's learning method is precisely this explicit approach. For implicit methods to perform well, the rules for the absence of dos should be learned in their

the classifier's recall (R) for C is l/n . Balance between R and P can be measured in various ways such as F -measure [4], the $R = P$ point, etc.

² A *weak* learner is an algorithm that, given $\epsilon \leq 1/2 - \gamma$ ($\gamma > 0$) and $\delta > 0$, can achieve an error rate of ϵ , with a probability of at least $(1 - \delta)$.

totality for *every* r2l rule. This may not be always possible. One such situation is when **dos** has many disjoint conditions for its presence, and its examples are split across many rules of r2l, to an extent that **dos** rules or absence thereof may not be completely learned via the piecemeal approach taken by implicit methodology. The explicit approach has an advantage in such situations by being able to collect the false positives of **dos** together and learn stronger rules to exclude them.

In this paper, our key contribution is the detailed qualitative analysis to argue that boosting does not impart an explicit learning capability to its base algorithm that learns using the implicit approach. Boosting can achieve higher recall via its intelligent yet incremental weight updating mechanism, which translates into removing the relevant false positives also in a piecemeal fashion. Thus, it may not imitate the effect of *collecting* the false positives to learn their complete or correct description, and hence fail to achieve a good balance between recall and precision. In the rest of the paper, we elaborate this reasoning further and support our arguments empirically on a wide range of synthetic and real-world scenarios.

2 Overview of Algorithms

We first briefly describe the two key algorithms being compared in this paper.

Boosting:

All the boosting algorithms (AdaBoost [5], SLIPPER [6], AdaCost [8], CSB1, CSB2 [7], and RareBoost [9]) work in iterations, each time learning a classifier model via a weak learner on a different weighted distribution of training records. After each iteration, weights on all the examples are updated in a manner that forces the weak classifier to focus more on the incorrectly classified examples in the next iteration. In the end, prediction of an example's class is made using the classifiers learned in all the iterations via a weighted voting process. For the rare class problem, a crucial factor that distinguishes different algorithms is the differences in the weight update factors [9]. For a cost-sensitive boosting algorithm AdaCost, here are the weight update equations from iteration t to $(t+1)$ for the four types of examples defined with respect to the rare class (TP: true positive, FP: false positive, TN: true negative, and FN: false negative) [9]: $TP_{t+1} \rightarrow TP_t/\gamma$, $TN_{t+1} \rightarrow TN_t/\gamma$, $FP_{t+1} \rightarrow FP_t * \gamma^{(f+1)/f}$, $FN_{t+1} \rightarrow FN_t * \gamma^2$, where $f \geq 1$ is the cost-factor given to the misclassification of the rare class (false negatives); $\gamma = e^{0.5\alpha_t}$; and α_t is the strength assigned to the vote of t^{th} classifier in the voting process. Although we will present our arguments with AdaCost in mind, they are also applicable to other boosting algorithms.

PNrule:

Given a training data-set and the target class, the PNrule algorithm [3] learns a binary two-phase model for the target class. PNrule framework was introduced first in [2], where it was also successfully extended to the multi-class problems

of the network intrusion detection domain. It was later analyzed in detail for the rare class problems in [3] to show its strength over single-phase algorithms such as RIPPER [10] and C4.5rules [11]. PNrul learns a model of disjunctive normal form (DNF) consisting of two kinds of rules: P-rules predicting presence of the target rare class C and N-rules predicting absence of C . The first phase of learning (P-phase) learns P-rules in a sequential covering manner. The goal is to achieve high recall for C by inducing high support rules. Accuracy is traded off in favor of support in the later iterations, in order to minimize exposure to the small disjuncts problem [3]. This leads to a less precise model. Then, the true and false positives covered by P-rules are collected together and the second phase (N-phase) learns N-rules to remove the false positives *collectively*. The goal is to improve overall precision, while keeping recall at an acceptable level. N-phase also follows sequential covering³.

The addition of rules in P-phase and rule growth in the N-phase are driven by two lower recall limit parameters, rp and rn , resp. Usually, these can be tuned for achieving better performance. Due to lack of space, we refer the reader to [2,3] for details of these and other parameters.

3 Boosting vs. PNrul: What Lacks in Boosting?

In this section, we present qualitative arguments to show that one crucial feature of PNrul is lacking in the boosting mechanism.

PNrul *collects* all the examples covered by the union of P-rules, and learns rules to remove false positives from this collection to improve precision. This feature is pictorially illustrated in Figure 1, which shows a snapshot of the process of learning a model to distinguish rare class C from the other class NC . First phase learns the P-rules P_i . Second phase then learns N-rules on the union of P_i -covered examples, to regain precision. This explicit learning approach of PNrul allows it to learn a strong N-rule such as N_0 . Now, a single-phase method such as RIPPER [10] has to learn N_0 *implicitly* by learning the absence of each of its pieces intersecting with each P_i . It can perform as well as PNrul only if there are sufficient number of examples available to learn a *complete* description for the absence of *each* of these pieces. As shown in [3], there exist many situations where this condition is difficult to meet, especially when the complete description of absence consists of large number of conjunctive conditions. There are other reasons also [3], such as reducing the small disjuncts problem, that motivate PNrul's preference for some less accurate P_i rules, which in turn requires learning of a rule such as N_0 . Thus, in many situations, it is crucial to have an ability to achieve an effect similar to that of learning N_0 from the *collection* of multiple P-rules.

³ PNrul's philosophy is similar to the concept of counterfactuals [12] or ripple-down-rules [13], from first order logic learning and knowledge based systems, respectively. To our knowledge, there is no evidence of their applicability and scalability to large real-world problems of propositional learning that PNrul addresses.

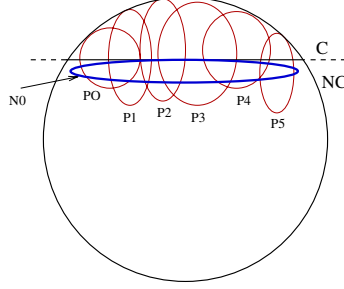


Fig. 1. PNrule's ability to collectively remove false positives with strong N-rules

Now, we try to qualitatively argue that the mechanism of boosting does not have the ability to effectively capture a rule such as N_0 , given that its base learner follows an implicit learning approach. In any given iteration, boosting instructs the weak classifier to operate on a weighted distribution of the training examples. The incremental process of weight update is illustrated in Figure 2. The figure shows first two iterations of a boosting algorithm. Part I shows the decision boundary of classifier C_1 learned in the first iteration. Within this boundary, every example is predicted as C and outside of it, all are predicted as NC.

At the end of this iteration, weights of the true positive (TP) examples of region J and the true negative (TN) examples of region L are reduced, while weights on the false negative (FN) examples of region K and false positive (FP) examples of region M are increased. Now, the weak classifier geared towards learning a model for C tries to focus more on capturing the C examples with high weight and avoid capturing the high weight NC examples. In the process, it may capture some NC examples whose weight has become small. Similarly, a weak classifier geared for learning NC's model, tries to capture more of NC's examples with higher weights and less of C's examples bearing higher weights. The net effect is to force the boundary of the next classifier to shift to C_2 (part II) that covers more of region K, less of regions J and M and possibly some of region L.

Now, after this iteration all the examples get divided into eight different types as shown by eight different regions (P-W). The weights, calculated using the AdaCost formulae of Section 2, and the ensemble-based predictions of the examples in each of these regions are shown at the second level of each tree in part III of the figure. The relative strengths of classifiers C_1 and C_2 will determine whether examples in regions Q, R, U, and V are predicted as C or NC. For example, if $\alpha_2 > \alpha_1$, then examples in U are predicted as belonging to C. Also, unless $\alpha_1 = \alpha_2$, examples in U cannot have the same prediction as examples in V. Similarly for Q and R. In fact, the final prediction of any example after a given number of iterations will be determined by the total strength of classifiers predicting it as positive versus total strength of the classifiers predicting it as negative. The effect of U and V being captured by one N-rule of the PNrule

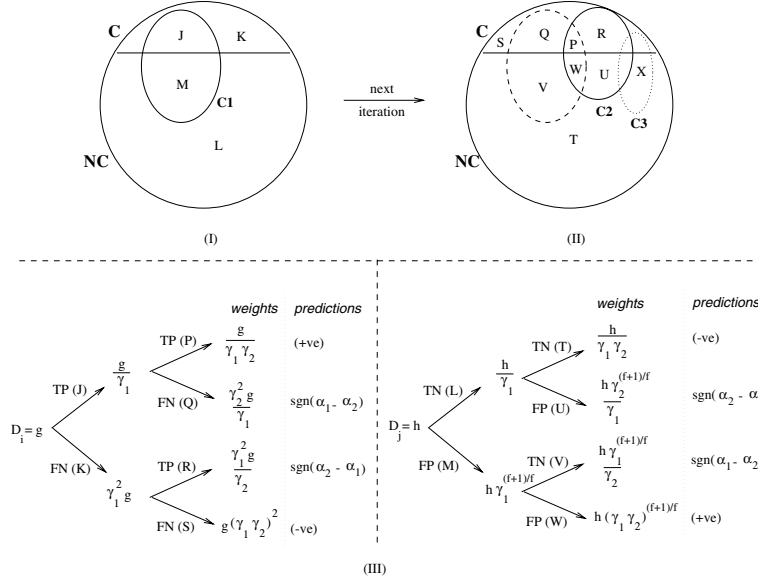


Fig. 2. Journey of the weight of a given example in boosting iterations. I. After first iteration. II. After second iteration. III. How weights change for each region in I and II, and how predictions are made. D_i is weight of example i of class C. D_j is weight of example j of class NC

algorithm, is equivalent to a simultaneous prediction of both U and V as true negatives by the ensemble of classifiers. We can clearly see that boosting cannot do this with the ensemble of C1 and C2. However, one can argue that going one more iteration where the classifier C3 covers a region as shown in part II of the figure may solve the problem. Now, the prediction for region U is $\text{sgn}(\alpha_2 - \alpha_1 - \alpha_3)$; for region V, it is $\text{sgn}(\alpha_1 - \alpha_2 - \alpha_3)$; and for region X, it is $\text{sgn}(\alpha_3 - \alpha_1 - \alpha_2)$. As long as the sum of each pair of α_1 , α_2 , and α_3 is greater than the third value, each of the regions U, V, and X can be predicted as true negative simultaneously by the 3-ensemble model, thus achieving an effect similar to learning a N-rule encompassing the three. Although boosting theory [5] suggests how to choose α for each iteration, the boosting method does not have any control over the relationship between the α values *across* iterations. Thus, the desired relationship between α_1 , α_2 , and α_3 values cannot be guaranteed. *This lack of guarantee essentially indicates the existence of situations where boosting cannot achieve the effect of collecting many false positives required for learning their stronger description*, and hence it can fail to achieve good precision for a reasonable recall level.

Generic expressions of α_t values are needed to formally support the above statement. Even for a simpler form of AdaCost, which assumes that each base learner t gives a model $h_t(x_j) \rightarrow \{0, 1\}$ (ignoring any confidence rating), we

get $\alpha_t = 0.5 \ln((1 + r_t)/(1 - r_t))$, where $r_t = 0.5 \sum_{i:TP,TN} D_t(i) + 0.5(1 + 1/f) \sum_{i:FP} D_t(i) - \sum_{i:FN} D_t(i)$. The value $D_t(i)$ is the weight of i^{th} example, which depends on the γ_t values for all the classifiers learned upto iterations $t - 1$. For example, after iteration T, weight of an example of class NC is:

$$D_T(i) = h * \frac{\prod_{\{t:h_t \text{ predicts } i \text{ as FP}\}}^T \gamma_t^{(f+1)/f}}{\prod_{\{t:h_t \text{ predicts } i \text{ as TN}\}}^T \gamma_t}, \quad (1)$$

where h is its initial weight. Each γ_t in turn depends on α_t (Section 2). Thus, D_t and α_t have a complex cyclic relationship, which makes it difficult to perform rigorous formal analysis. This is the reason we resort to the qualitative analysis and arguments.

As the expressions for r_t and α_t above indicate, if the false positive region (such as U, V, or X) of a base classifier model h_t is larger, then r_t and α_t values for h_t will be higher. Thus, the contribution of h_t to overall weight will be significant. Moreover, this region will have possibly large number of overlaps with the false positive regions of the other classifiers. So, in the ensemble-based voting process, an example in this region will get high vote as a false positive, thus diminishing its chances of getting correctly predicted. Of course, the weight update mechanism, as illustrated in the part III of Figure 2, will attempt to avoid such situation by increasing the weights of examples in this region, so that fewer and fewer future classifiers cover the region. But, as fewer classifiers cover the region (thus more classifiers treating the examples as TN), Equation 1 indicates that the weights on the examples will either stabilize or start reducing. The relative strengths of the classifiers covering the region will determine the time at which the weight increases sufficiently for the region not to be covered or decreases to an extent that the region is again vulnerable for misclassification. This is a cyclical process, and depending on when an example switches roles among FP and TN, the effect of collective removal of the false positives may be only partially achievable.

In summary, we have qualitatively argued that the incremental nature of weight update mechanism and the ensemble-based voting process are not sufficient to allow boosting to impart an explicit learning effect, similar to that of PNrule, to a base learner that uses an implicit approach.

4 Results

We now attempt to empirically support the qualitative arguments of section 3. Two variations of the IREP* [10] algorithm are used as the base learners for boosting. IREP*-1 learns C's model and NC is the default class. IREP*-2 learns a model for NC also from the whole training set⁴.

We use F_1 -measure [4] as the performance metric, defined as $(2 R P)/(R+P)$, where recall (R) and precision (P) are with respect to C: $R = TP/(TP+FN)$ and

⁴ PNrule learns NC's model using only the examples covered by at least P-rule.

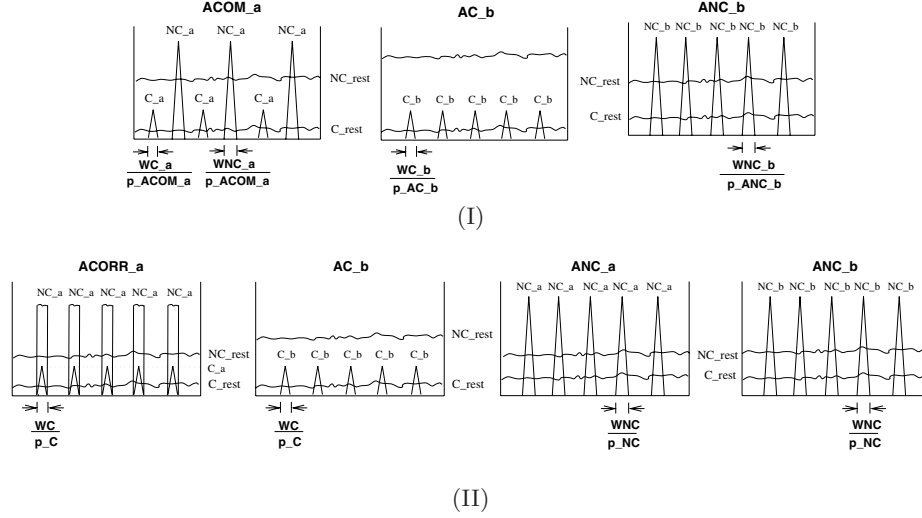


Fig. 3. Models generating synthetic data. **(I)**: No correlations among attributes. **(II)**: With correlations among attributes of types $ACORR_a$ and ANC_a for sub-class NC_a

$P = TP/(TP+FP)$. We use P instead of a false positive rate of $FP/(FP+TN)$ because we are interested in minimizing false positives of C with respect to the total examples predicted as C . Also other domains such as information retrieval which are concerned with R and P values commonly use the F_1 metric.

For AdaCost, we use various values for f : 1, 2, 5, 10. For each f value, we run 50 and 25 iterations with IREP*-1 and IREP*-2, respectively. The parameters f and the number of iterations, are chosen to yield the highest F_1 value on the validation data (one-third random sample of the training data). For PNrul, the parameters chosen are the rule evaluation function (z-number [2] or information gain [10]), rp (0.7 or 0.95), and rn (0.0 or 0.7). The optimal parameters values are used to report F_1 values on the test data.

Two different algorithms are compared by first using the p-test⁵ with 95% confidence level [14] to compare their R and P metrics individually. If either metric is better for a classifier and the other metric is comparable or better, then it is a better classifier. If one metric is better and the other worse, then the classifier with higher F_1 value is better. This method is driven more by the actual R and P comparisons rather than a pure F_1 -based comparisons.

4.1 Results on Synthetic Datasets

We generate various synthetic datasets by using two different data models described in Figure 3. Each model has different types of attributes. Type is

⁵ p-test can be thought of as a two-sided t-test around mean.

identified by the nature of the subclass distribution over the attribute. These histogram-based models were first introduced in [3] and were designed to test PNrul’s 2-phase nature. The key idea is to split the examples of rare classes into multiple peaks in the distribution and split the number of non-rare class examples into multiple subclasses. We also make each subclass distinguishable by only one attribute or two correlated attributes. For example, ANC_b can distinguish NC_b subclass of the non-rare class, because of all attributes, NC_b ’s distribution is non-random only over ANC_b , and of all subclasses, only NC_b is non-randomly distributed over ANC_b . So, for an implicit (or one-phase) rule induction approach, the descriptions of false positives (i.e. subclasses of NC) must be learned from the examples that fall in the non-peak regions of the ANC attributes. Instead, PNrul needs to learn the peak regions in the second phase where it explicitly learns false positive descriptions on the collection of examples covered by at least one P-rule. As we stated in the introduction, the goal is to see if boosting mechanism imparts an explicit learning capability to an implicit base learner. These datasets will help us do precisely that.

For each model, there are several parameters (some defined in Figure 3) such as the number of subclasses of each type, the number of peaks and the peak width for a given type of attribute (e.g. for type AC_b there are p_{AC_b} peaks with total width of WC_b), and the proportion of C vs. NC (the rarity). These can be varied to control the learning difficulty. For example, if the peaks in AC_b are wider, then more examples of NC get captured, making it difficult to achieve higher precision for reasonably high recall for C. Note that in type (II) model, the subclasses of type NC_a have distribution peaks in both $ACORR_a$ and ANC_a , that introduces a correlation between these attributes.

We generated multiple datasets with 1:20 ratio of C:NC using the model (I) with 30,000 NC examples, and 1:10 ratio using model (II) with 20,000 NC examples. Some key results are presented in Tables 1.

The left table in part (A) shows that as the peak widths⁶ WC_a and WC_b increase in model (I), the ability to remove false positives effectively is required, and PNrul outperforms AdaCost. The performance difference is less dependent on the widths of NC peaks, implying that PNrul’s approach of explicitly and collectively removing false positives is helping it and AdaCost fails to imitate this ability.

Right two tables of part (A) show similar results with the type (II) model of Figure 3. In particular, the top right table shows the effect of adding correlations. sc-0 and sc-1 both have identical parameters⁷, except that sc-0 has no attributes of type $ACORR_a$ or ANC_a (i.e. $n_{ACORR_a} = n_{ANC_a} = 0$) and sc-1 has correlations between 3 pairs of $ACORR_a$ and ANC_a type attributes. The numbers $n_{ACORR_a} + n_{AC_b}$ and $n_{ANC_a} + n_{ANC_b}$ is same for both datasets. It is difficult to discriminate between correlated parts of C_a and NC_a in sc-1,

⁶ Compare widths to a range of 50 for all attributes. Other parameters for these datasets are $n_{ACOM_a} = 3$, $p_{ACOM_a} = 3$, $n_{AC_b} = 4$, $p_{AC_b} = 5$, $n_{ANC_b} = 5$, and $p_{ANC_b} = 5$.

⁷ $p_C = 4$, $p_{NC} = 8$, $W_C = 0.2$, $W_{NC} = 2$.

thus F_1 value drops for all algorithms. But, C_a can still be discriminated against NC_b and parts of NC_a , provided an algorithm can collect and learn from only their *relevant* false positives (the explicit learning approach). PNrule has this ability. As the results indicate, AdaCost fails to impart this ability to IREP*-1 or IREP*-2. The lower right table of part (A) further corroborates our claim because as p_{NC} , the number of peaks over ANC_a and ANC_b , increases in model (II), the false positives get scattered more, requiring an effect of explicit learning.

Part (B) of Table 1 justifies the use of AdaCost as the strongest boosting-based competitor of PNrule for the two difficult datasets from each model.

Finally, we demonstrate the effect of varying the class proportions on a moderately difficult dataset snc-6 from model (I) and the representative dataset sc-1 from model (II). The results in Table 1(C) indicate that PNrule is significantly better for rarer problems (as Cfrac $\downarrow$). Especially with correlations, PNrule seems to be better even for larger proportions of C. This is interesting because the problem of separating correlated C and NC is difficult in general, and is made more difficult by the rarity. However, the ability to remove false positives collectively is required irrespective of the rarity, and on this count, PNrule outperforms AdaCost.

Due to lack of space, we have shown results on representative datasets, but similar results were observed for a wide variation of different parameters of the models.

4.2 Results on Real and Benchmark Datasets

We used the OHSU Medline document corpus of medical abstracts from years 1990 and 1991 [15]. Out of 805 topics having a population of $> 0.2\%$ (in around 148,000 total documents), we select 33 topics (least rare topic has about 5% population). Some example topics are AIDS, Aging, Anoxia, Colon, Parkinson Disease, Placenta, etc. For each topic, after stemming and stop-word removal, we intersect the top 75 distinguishing words according to the mutual information and Z-number [2] metrics, to form the attribute set for its binary classification. According to the 95% confidence p-test, PNrule vs. AdaCost comparison is 3-26-4-0; i.e., PNrule outperforms AdaCost on 3 datasets, both are comparable on 26, AdaCost is better on 4, and on no dataset is IREP* better than the two⁸. This indicates that only in 4 cases, use of AdaCost is justifiable based on its high performance despite its low interpretability. On 29 cases, using more interpretable PNrule model is preferable.

We also did experiments on datasets from the UCI machine learning repository [16]. We form a binary problem for each class having $< 35\%$ proportion in the king-rook-king, dna, csb-smoke, led+17, led, and letter datasets⁹. Out of 49 such problems, PNrule vs. AdaCost comparison is 7-23-18-1. This shows

⁸ AdaCost is sometimes worse than its base learner because of our 1-way cross-validation method of choosing the optimal number of iterations for AdaCost.

⁹ king-rook-king is recognized as a tough problem for propositional learning, other datasets were chosen because they contain many rare classes

Table 1. Results on synthetic models of Figure 3, comparing F_1 performance. Bold numbers indicate statistically better or comparable (95% confidence p-test) algorithms. **(A)** Effect of varying model parameters. **(B)** Comparing various boosting algorithms for datasets snc-9 and sc-1. ABst: AdaBoost, RB-1/2: RareBoost-1/2, SLIP: SLIPPER [6], ACst: AdaCost. CSB1 and CSB2 are from [7]. Refer to [9] for RB-1 and RB-2. **(C)** Results for varying proportion of class C in the training set (Cfrac) for datasets snc-6 and sc-1

Dset	Model (I) of Figure 3				
	$WC_{a,b}$	$WNC_{a,b}$	IREP*	ACST	(t,f) PNrule
snc-1	0.6, 1	0.6, 1	84.72	84.72	(1, 1) 85.67
snc-2	0.6, 1	0.6, 4	35.79	59.60	(11, 10) 54.41
snc-3	0.6, 1	2.4, 1	23.30	69.98	(3, 10) 72.42
snc-4	0.6, 1	2.4, 4	25.95	58.44	(16, 10) 59.42
snc-5	0.6, 4	2.4, 4	21.21	49.81	(15, 10) 58.23
snc-6	2.4, 4	0.6, 4	2.47	48.17	(4, 5) 58.20
snc-7	2.4, 4	2.4, 1	1.04	65.16	(1, 2) 74.29
snc-8	2.4, 1	2.4, 4	1.79	47.52	(18, 5) 56.65
snc-9	2.4, 4	2.4, 4	0.26	38.19	(20, 10) 45.98

Dset	Model (II) of Figure 3			
	IREP*	ACST	(t,f)	PNrule
sc-0	85.45	89.08	(3, 10)	88.45
sc-1	41.65	41.65	(1, 1)	59.03

p_{NC}		Dset: sc-1			
		IREP*	ACST	(t,f)	PNrule
2	67.75	72.00	(12, 5)	73.33	
4	32.11	60.96	(46, 5)	57.10	
8	41.65	41.65	(1, 1)	59.03	
12	41.17	50.13	(47, 5)	56.59	
20	3.90	28.57	(21, 5)	51.21	

(A)

Dset	ABst	RB-1	SLIP	RB-2	CSB1	CSB2	ACst
snc-9	26.60	31.78	16.65	32.96	30.71	24.91	45.98
sc-1	23.43	28.39	20.23	34.15	36.75	37.00	41.65

(B)

Cfrac	Dset: snc-6			
	IREP*	ACST	(t,f)	PNrule
2.44%	0.00	7.43	(3, 10)	18.53
4.76%	2.47	48.17	(4, 5)	58.20
9.09%	8.79	53.37	(14, 5)	60.30
16.67%	60.62	68.21	(9, 5)	66.12
28.57%	47.10	76.60	(17, 2)	71.82
50.00%	74.94	81.45	(4, 1)	69.11

Cfrac	Dset: sc-1			
	IREP*	ACST	(t,f)	PNrule
2.44%	0.00	0.76	(1, 5)	13.97
4.76%	1.55	15.38	(31, 10)	19.14
9.09%	41.65	41.65	(1, 1)	59.03
16.67%	3.22	45.80	(17, 5)	79.73
28.57%	23.09	59.89	(12, 2)	80.48
50.00%	65.01	78.44	(1, 2)	78.53

(C)

existence of 31 of 49 scenarios, where use of AdaCost is not justifiable. On the 17 datasets formed from the king-rook-king problem, the comparison is 6-7-4-0; i.e., on 13 of 17 datasets, PNrule is either comparable or better than AdaCost.

5 Concluding Remarks

The problem of achieving better balance between recall and precision while predicting a rare event class is challenging. In this context, we compare PNrule, a strong non-meta technique of two-phase rule induction to AdaCost, a strong cost-sensitive boosting methodology that was shown to outperform other boosting algorithms from the recall-precision perspective. The non-meta techniques have an advantage of being interpretable. We argue via detailed qualitative analysis that boosting lacks a crucial ability of PNrule to collect only the relevant

false positives and explicitly learn rules to exclude them. The arguments are supported using various synthetic and real-world datasets. PNrule is especially better for rarer classes, and when there is a strong correlation among the distinguishing rules of the rare class and the non-rare class. As an extension of the observations and arguments of this paper, one can investigate two aspects w.r.t. rare class problems: a. Does the boosting performance depend crucially on the base learner?, b. Can one improve boosting by using an explicit learning method of PNrule at its base learner?. Our work on these issues appears at the ACM KDD'2002 conference.

Acknowledgments

The contribution to this work by Prof. Vipin Kumar was supported in part by the Army High Performance Computing Research Center cooperative agreement number DAAD19-01-2-0014, the content of which does not necessarily reflect the position or the policy of the government, and no official endorsement should be inferred.

References

1. Holte, R. C., Japkowicz, N., Ling, C. X., Matwin, S.: (eds.) Learning from imbalanced data sets workshop. Technical Report WS-00-05, AAAI Press (2000) [237](#)
2. Agarwal, R. C., Joshi, M. V.: PNrule: A new framework for learning classifier models in data mining (A case-study in network intrusion detection). In: Proceedings of First SIAM Conference on Data Mining, Chicago (2001) [237](#), [239](#), [240](#), [244](#), [246](#)
3. Joshi, M. V., Agarwal, R. C., Kumar, V.: Mining needles in a haystack: Classifying rare classes via two-phase rule induction. In: Proc. of ACM SIGMOD Conference, Santa Barbara, CA (2001) 91–102 [237](#), [238](#), [239](#), [240](#), [245](#)
4. van Rijsbergen, C. J.: Information Retrieval. Butterworths, London (1979) [238](#), [243](#)
5. Schapire, R., Singer, Y.: Improved boosting algorithms using confidence-rated predictions. Machine Learning **37** (1999) 297–336 [238](#), [239](#), [242](#)
6. Cohen, W., Singer, Y.: A simple, fast, and effective rule learner. In: Proc. of Annual Conference of American Association for Artificial Intelligence. (1999) 335–342 [238](#), [239](#), [247](#)
7. Ting, K. M.: A comparative study of cost-sensitive boosting algorithms. In: Proc. of 17th Intl. Conf. on Machine Learning. (2000) 983–990 [238](#), [239](#), [247](#)
8. Fan, W., Stolfo, S. J., Zhang, J., Chan, P. K.: AdaCost: Misclassification cost-sensitive boosting. In: Proc. of 6th Intl. Conf. on Machine Learning (ICML). (1999) [238](#), [239](#)
9. Joshi, M. V., Kumar, V., Agarwal, R. C.: Evaluating boosting algorithms to classify rare classes: Comparison and improvements. In: Proc. of The First IEEE International Conference on Data Mining (ICDM), San Jose, CA (2001) [238](#), [239](#), [247](#)

10. Cohen, W. W.: Fast effective rule induction. In: Proc. of Twelfth International Conference on Machine Learning, Lake Tahoe, California (1995) 238, 240, 243, 244
11. Quinlan, J. R.: C4.5: Programs for Machine Learning. Morgan Kaufmann (1993) 240
12. Vere, S. A.: Multilevel counterfactuals for generalizations of relational concepts and productions. *Artificial Intelligence* **14** (1980) 139–164 240
13. Compton, P., Jansen, R.: A philosophical basis for knowledge acquisition. *Knowledge Acquisition* **2** (1990) 241–257 240
14. Yang, Y., Liu, X.: A re-examination of text categorization methods. In: 22nd Annual Intl. Conf. on Information Retrieval (SIGIR). (1999) 42–49 244
15. Hersh, W., Buckley, C., Leone, T., Hickam, D.: OHSUMED: An interactive retrieval evaluation and new large test collection for research. In: In Proceedings of the 17th Annual ACM SIGIR Conference. (1994) 192–201 246
16. Blake, C., Merz, C.: UCI repository of machine learning databases (1998) <http://www.ics.uci.edu/~mllearn/MLRepository.html>. 246

Dependency Detection in MobiMine and Random Matrices

Hillol Kargupta¹, Krishnamoorthy Sivakumar², and Samiran Ghosh¹

¹ Department of Computer Science and Electrical Engineering
University of Maryland Baltimore County
Baltimore, MD 21250, USA
{hillol, sghosh1}@cs.umbc.edu
<http://www.cs.umbc.edu/~hillol>

² School of Electrical Engineering and Computer Science
Washington State University
Pullman, WA 99164-2752, USA
siva@eecs.wsu.edu

Abstract. This paper describes a novel approach to detect correlation from data streams in the context of MobiMine — an experimental mobile data mining system. It presents a brief description of the MobiMine and identifies the problem of detecting dependencies among stocks from incrementally observed financial data streams. This is a non-trivial problem since the stock-market data is inherently noisy and small incremental volumes of data makes the estimation process more vulnerable to noise. This paper presents EDS, a technique to estimate the correlation matrix from data streams by exploiting some properties of the distribution of eigenvalues for random matrices. It separates the “information” from the “noise” by comparing the eigen-spectrum generated from the observed data with that of random matrices. The comparison immediately leads to a decomposition of the covariance matrix into two matrices: one capturing the “noise” and the other capturing useful “information.” The paper also presents experimental results using Nasdaq 100 stock data.

1 Introduction

Mobile computing devices like PDAs, cell-phones, wearables, and smart cards are playing an increasingly important role in our daily life. The emergence of powerful mobile devices with reasonable computing and storage capacity is ushering an era of advanced data and computationally intensive mobile applications. Monitoring and mining time-critical data streams in a ubiquitous fashion is one such possibility. Financial data monitoring, process control, regulation compliance, and security applications are some possible domains where such ubiquitous mining is very appealing.

This paper considers the problem of detecting dependencies among a set of features from financial data streams monitored by a distributed mobile data mining system called the *MobiMine*. MobiMine is *not a market forecasting system*.

It is neither a traditional system for stock selection and portfolio management. Instead it is designed for drawing the attention of the user to time critical “interesting” emerging characteristics in the stock market.

This paper explores a particular module of the MobiMine that tries to detect statistical dependencies among a set of stocks. At a given moment the system maintains a relevant amount of historical statistics and updates that based on the incoming block of data. Since the data block is usually noisy, the statistics collected from any given block should be carefully filtered and then presented to the user. This paper offers a technique for extracting useful information from individual data blocks in a data stream scenario based on some well-known results from the theory of randomized matrices. It presents a technique to extract significant Eigen-states from Data Streams (EDS) where the data blocks are noisy. The technique can also be easily applied to feature selection, feature construction, clustering, and regression from data streams. More generally, EDS offers a way to filter the observed data, so that any data mining technique (exploratory or otherwise) can later be applied on the filtered data. In the context of financial data streams, any technique for the analysing, forecasting, and monitoring stock prices can be applied on the filtered data. As such, the EDS by itself is not a market forecasting tool.

The technical approach of the proposed work is based on the observation that the distribution of eigenvalues of random matrices [1] exhibit some well known characteristics. The basic idea is to compare a “random” phenomenon with the behavior of the incoming data from the stream in the eigen space and note the differences. This results in a decomposition of the covariance matrix: one capturing the “noise” and the other capturing useful “information.” Note that the terms “noise” and “information” are used in a generic sense. In the context of financial data, the change in price of a stock is influenced by two types of factors: (a) causal factors that directly or indirectly have an influence in the current or future performance of the company. This would include earnings, revenue, and future outlook of that company, performance of competitors, state of the overall economy, etc. This corresponds to the “information” part. (b) random factors that might be completely unpredictable and totally unrelated to the performance of the company. This corresponds to the “noise” part.

The eigenvectors generated from the “information” part of the covariance matrix are extracted and stored for the chosen application. Moreover, the eigenvectors can be used to filter the observed data by projecting them onto the subspace spanned by the eigenvectors corresponding to the “information.”

Section 2 presents a brief overview of the MobiMine system. Section 3 discusses relevant theory of random matrices and then describes the EDS technique. Section 4 presents the experimental results. Section 5 concludes the work and identifies future work.

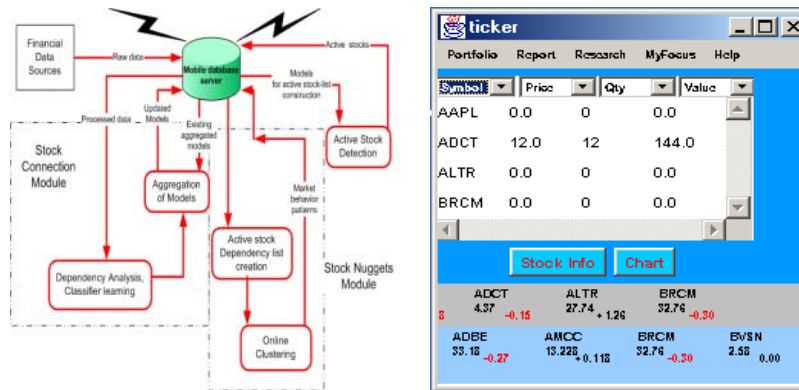


Fig. 1. (Left) The architecture of the MobiMine Server. (Right) The main interface of MobiMine. The bottom-most ticker shows the WatchList; the ticker right above the WatchList shows the stocks in the portfolio

2 The MobiMine System

This section presents an overview of the MobiMine, a PDA-based application for managing stock portfolios and monitoring the continuous stream of stock market data. The overview presented in this section covers the different modules of the MobiMine; not all of them make use of the random matrix-based techniques discussed so far in this paper. However, a general description is necessary to cast the current contribution in the context of a real application environment.

2.1 An Overview of the System

The MobiMine is a client-server application. The clients (Figure 2), running on mobile devices like hand-held PDAs and cell-phones, monitor a stream of financial data coming through the MobiMine server (Figure 1(Top)). The system is designed for currently available low-bandwidth wireless connections between the client and the server. In addition to different standard portfolio management operations, the MobiMine server and client apply several advanced data mining techniques in order to offer the user a variety of different tools for monitoring the stock market at any time from any where. Figure 1(Bottom) shows the main user interface of the MobiMine.

The main functionalities of the MobiMine are listed in the following:

1. Portfolio Management and Stock Tickers: Standard book-keeping operations on stock portfolios including stock tickers to keep an eye on the performance of the stocks in the portfolio.
2. FocusArea: Stock market data is often overwhelming. It is very difficult to keep track of all the developments in the market. Even for a full-time

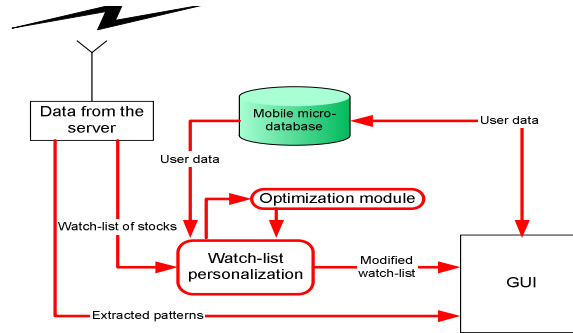


Fig. 2. The architecture of the MobiMine Client

professional following the developments all the time is challenging. It is undoubtedly more difficult for a mobile user who is likely to be busy with other things. The MobiMine system offers a unique way to monitor changes in the market data by selecting a subset of the events that is more “interesting” to the user. This is called the FocusArea of the user. It is a time varying feature and it is currently designed to support the following functionalities:

- (a) WatchList: The system applies different measures to assign a score to every stock under observation. The score is an indication of the “interesting-ness” of the stock. A relatively higher score corresponds to a more interesting stock. A selected bunch of “interesting” stocks goes through a personalization module in the client device before it is presented to the user in the form of a WatchList.
- (b) Context Module: This module offers a collection of different services for better understanding of the time-critical dynamics of the market. The main interesting components are,
 - i. StockConnection Module: This module allows the user to graphically visualize the “influence” of the currently “active” stocks on the user’s portfolio. This module detects the highly active stocks in the market and presents the causal relationship between these and the stocks in user’s portfolio, if any. The objective is to give the user a high level qualitative idea about the possible influence on the portfolio stocks by the emerging market dynamics.
 - ii. StockNuggets Module: The MobiMine Server continuously processes a data stream defined by a large number of stock features (fundamentals, technical features, evaluation of a large number of well-known portfolio managers). This module applies online clustering algorithms on the active stocks and the stocks that are usually influenced by them (excluding the stocks in the user’s portfolio) in order to identify similarly behaving stocks in a specific sector.

The StockConnection module tries to detect the effect of the market activity on user's portfolio. On the other hand, the StockNuggets module offers an advanced stock-screener-like service that is restricted to only time-critical emerging behavior of stocks.

- (c) Reporting Module: This module supports a multi-media based reporting system. It can be invoked from all the interfaces of the system. It allows the user to watch different visualization modules and record audio clips. The interface can also invoke the e-mail system for enclosing the audio clips and reports.

A detailed description of this system can be found elsewhere [2]. The following section discusses the unique philosophical differences between the MobiMine and traditional systems for mining stock data.

2.2 MobiMine: What It Is Not

A large body of work exists that addresses different aspects of stock forecasting [3,4,5,6,7,8] and selection [9,10] problem. The MobiMine is fundamentally different from the existing systems for stock forecasting and selection. First of all, it is different on the basis of philosophical point of view. In a traditional stock selection or portfolio management system the user initiates the session. User outlines some preferences and then the system looks for a set of stocks that satisfy the constraints and maximizes some objective function (e.g. maximizing return, minimizing risk). The MobiMine does not do that. Instead it initiates an action, triggered by some activities in the market. The goal is to draw user's attention to possibly time-critical information. For example, if the Intel stock is under-priced but its long time outlook looks very good then a good stock selection system is likely to detect Intel as a good buy. However, the MobiMine is unlikely to pick Intel in the WatchList unless Intel stock happens to be highly active in the market and it fits with user's personal style of investment. The Context detection module is also unlikely to show Intel in its radar screen unless Intel happens to be highly influenced by some of the highly active stocks in the market. This difference in the design objective is mainly based on our belief that mobile data mining systems are likely to be appropriate only for time-critical data. If the data is not changing right now, probably you can wait and you do not need to keep an eye on the stock price while you are having a lunch with your colleagues.

This paper focuses on the correlation-based dependency detection aspect of the used in the StockConnection module. The following section initiates the discussion.

3 Data Filtering and Random Matrices

Detecting dependencies among stocks is an important task of MobiMine for identifying the focus area of the user. Correlation analysis of time series data is a

common technique for detecting statistical dependencies among them. However, doing it online is a challenging problem since correlation must be computed from incrementally collected noisy data. At any given instant, the MobiMine can compute the correlation matrix among a set of stocks. However, the correlation may be completely misleading introduced by many factors like noise and small number of observations.

Accurate estimation of the correlation requires proper filtering of the correlation matrix. This paper considers an approach that removes the “noise” by considering the eigen values of the covariance matrix computed from the collected data. The noisy eigen-states are removed by exploiting properties of the eigen-distribution of random matrices. The eigenvalues of the covariance matrix can then be used, in conjunction with random matrix theory, to identify and filter out the noisy eigenstates.

In this section, we will first present a brief review of the theory of random matrices. We will then present the EDS, that works incrementally by observing one block of data at a time.

3.1 Introduction to Random Matrices

A random matrix X is a matrix whose elements are random variables with given probability laws. The theory of random matrices deals with the statistical properties of the eigenvalues of such matrices.

In this paper, we would be interested in the distribution of the eigenvalues of the sample covariance matrix obtained from a random matrix X . Let X be an $m \times n$ matrix whose entries X_{ij} , $i = 1, \dots, m$, $j = 1, \dots, n$ are i.i.d. random variables. Furthermore, let us assume that X_{11} has zero mean and unit variance. Consider the $n \times n$ sample covariance matrix $Y_n^{(m)} = \frac{1}{m} X X'$. Let $\lambda_{n1}^{(m)} \leq \lambda_{n2}^{(m)} \leq \dots \leq \lambda_{nn}^{(m)}$ be the eigenvalues of $Y_n^{(m)}$. Let $F_n^{(m)}(x) = (\sum_{i=1}^n U(x - \lambda_{ni}^{(m)}))/n$, be the empirical cumulative distribution function (c.d.f.) of the eigenvalues $\{\lambda_{ni}^{(m)}\}_{1 \leq i \leq n}$, where $U(x)$ is the unit step function. We will consider asymptotics such that in the limit as $N \rightarrow \infty$, we have $m(N) \rightarrow \infty$, $n(N) \rightarrow \infty$, and $\frac{m(N)}{n(N)} \rightarrow Q$, where $Q \geq 1$.

Under these assumptions, it can be shown that [11] the empirical c.d.f. $F_n^{(m)}(x)$ converges in probability to a continuous distribution function $F_Q(x)$ for every x , whose probability density function (p.d.f.) is given by

$$f_Q(x) = \begin{cases} \frac{Q\sqrt{(x-\lambda_{\min})(\lambda_{\max}-x)}}{2\pi x} & \lambda_{\min} < x < \lambda_{\max} \\ 0 & \text{otherwise,} \end{cases} \quad (1)$$

where $\lambda_{\min} = (1 - 1/\sqrt{Q})^2$ and $\lambda_{\max} = (1 + 1/\sqrt{Q})^2$.

3.2 EDS Approach for Online Filtering

Consider a data stream mining problem that observes a series of data blocks $X_1, X_2, \dots, X_s$, where X_t is an $m_t \times n$ dimensional matrix observed at time t (i.e., m_t

observations are made at time t). If the data has zero-mean, the sample covariance Cov_t based on data blocks $X_1, X_2, \dots, X_t$ can be computed in a recursive fashion as follows [12]:

$$\text{Cov}_t = \frac{\sum_{j=1}^{t-1} m_j}{\sum_{j=1}^t m_j} \left[\text{Cov}_{t-1} + \frac{m_t}{\sum_{j=1}^{t-1} m_j} \hat{\Sigma}_t \right] \quad (2)$$

where $\hat{\Sigma}_i = (X_i' X_i)/m_i$ is the sample covariance matrix computed from only the data block X_i .

In order to exploit the results from random matrix theory, we will first center and then normalize the raw data, so that it has zero mean and unit variance. This type of normalization is sometimes called Z-normalization, which simply involves subtracting the column mean and dividing by the corresponding standard deviation. Since the sample mean and variance may be different in different data blocks (in general, we do not know the true mean and variance of the underlying distribution), Equation 2 must be suitably modified. In the following, we provide the important steps involved in updating the covariance matrix incrementally. Detailed derivations can be found in [12].

Let μ_t, σ_t be the local mean and standard deviation row vectors, respectively, for the data block X_t and $\bar{\mu}_t, \bar{\sigma}_t$ be the aggregate mean and standard deviation, respectively, based on the aggregation of $X_1, X_2, \dots, X_t$. Let $\bar{X}_r, \hat{X}_r$ denote the local centered and local Z-normalized data, respectively, obtained from data block X_r ($1 \leq r \leq t$) using μ_r and σ_r . Moreover, at time t , let $\bar{X}_{r,t}, \hat{X}_{r,t}$ denote the actual centered and actual Z-normalized data obtained from data block X_r using the $\bar{\mu}_t$ and $\bar{\sigma}_t$. In particular, $\hat{X}_{r,t,[i,j]} = \bar{X}_{r,t,[i,j]} / \bar{\sigma}_{t,[j]} = (X_{r,t,[i,j]} - \bar{\mu}_{t,[j]}) / \bar{\sigma}_{t,[j]}$, where i, j denote row and column indices. Note that the aggregate mean $\bar{\mu}_t$ can be updated incrementally as follows: $\bar{\mu}_t = (\sum_{r=1}^t \mu_r m_r) / \sum_{r=1}^t m_r = (\bar{\mu}_{t-1} \sum_{r=1}^{t-1} m_r + \mu_t m_t) / \sum_{r=1}^t m_r$. Let us define Z_t to be the covariance matrix of the aggregation of centered (or zero mean) data $\bar{X}_{1,t}, \bar{X}_{2,t}, \dots, \bar{X}_{t,t}$, and z_t be the local covariance matrix of the current block $\bar{X}_t$. Note that

$$\bar{\sigma}_{t,[j]} = \sqrt{Z_{t,[j,j]}}, \text{ and } \text{Cov}_{t,[i,j]} = \frac{Z_{t,[i,j]}}{\bar{\sigma}_{t,[i]} \times \bar{\sigma}_{t,[j]}}, \quad 1 \leq i, j \leq n, \quad (3)$$

where Cov_t is the covariance matrix of the aggregated Z-normalized data $\hat{X}_{1,t}, \dots, \hat{X}_{t,t}$. Therefore, the Z-normalization problem is reduced to that of incrementally updating the covariance matrix Z_t on the centered data. Define $\bar{\Delta}_t = (\bar{\mu}_t - \bar{\mu}_{t-1})$ and $\Delta_t = (\bar{\mu}_t - \mu_t)$. It is then easy to show that [12]

$$\begin{aligned} \bar{X}_{r,t}' \bar{X}_{r,t} - \bar{X}_{r,t-1}' \bar{X}_{r,t-1} &= m_r \bar{\Delta}_t' \bar{\Delta}_t, \quad \bar{X}_{t,t}' \bar{X}_{t,t} - (\bar{X}_t)' (\bar{X}_t) = m_t \Delta_t' \Delta_t, \quad \text{and} \\ Z_t &= Z_{t-1} + \bar{\Delta}_t' \bar{\Delta}_t \sum_{r=1}^{t-1} m_r + \bar{X}_t' \bar{X}_t + m_t \Delta_t' \Delta_t \end{aligned} \quad (4)$$

The above discussion shows that the covariance matrix can be incrementally updated, which can then be used to compute eigenvectors. An online algorithm

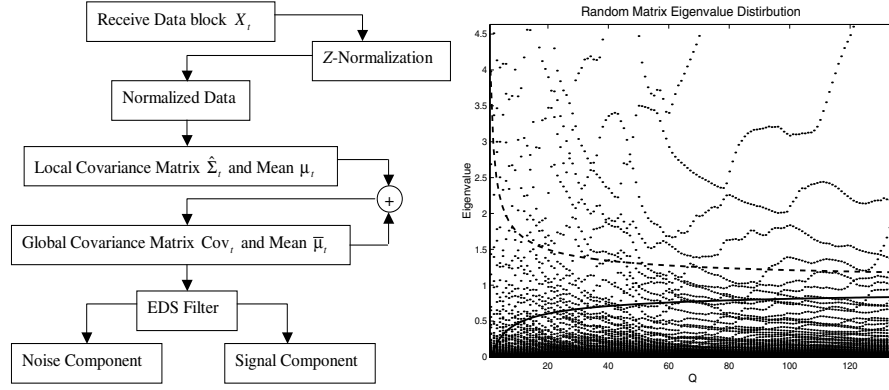


Fig. 3. (Left) The flow chart of the proposed EDS approach for mining data streams. (Right) The distribution of the eigen values, $\lambda_{\max}$ and $\lambda_{\min}$ with increasing Q . The graph is generated using the financial data set

can directly compute the eigenvectors of this matrix. However, this simplistic approach does not work well in practice due to two main problems: (a) data may be inherently noisy and (b) the number of observations (m_i) made at a given time may be small. Both of these possibilities may produce misleading covariance matrix estimates, resulting in spurious eigen-states. It is important that we filter out the noisy eigenstates and extract only those states that belong to the eigenspace representing the underlying information.

In this paper, we assume that the observed data is stationary and consists of actual information corrupted by random noise. The proposed technique decomposes the covariance matrix into two components: (1) the noise part and (2) the information part by simply comparing the eigenspace of the covariance matrix of observed data with that of a randomly generated matrix. In other words, we compare the distribution of the empirically observed eigenvalues with the theoretically known eigenvalue distribution of random matrices given by Equation 1. All the eigenvalues that fall inside the interval $[\lambda_{\min}, \lambda_{\max}]$ correspond to noisy eigenstates. Following are some of the main steps at any time t in the EDS approach:

1. Based on the current estimate of the covariance matrix Cov_t , compute the eigenvalues $\lambda_{t,1} \leq \dots \leq \lambda_{t,n}$.
2. Identify the noisy-eigenstates $\lambda_{t,i} \leq \lambda_{t,i+1} \dots \leq \lambda_{t,j}$ such that $\lambda_{t,i} \geq \lambda_{\min}$ and $\lambda_{t,j} \leq \lambda_{\max}$. Let $A_{t,n} = \text{diag}\{\lambda_{t,i}, \dots, \lambda_{t,j}\}$, be the diagonal matrix with all the noisy eigenvalues. Similarly, let $A_{t,s} = \text{diag}\{\lambda_{t,1}, \dots, \lambda_{t,i-1}, \lambda_{t,j+1}, \dots, \lambda_{t,n}\}$, be the diagonal matrix with all the non-random eigenvalues.

Let $A_{t,n}$ and $A_{t,s}$ be the matrices whose columns are eigenvectors corresponding to the eigenvalues in $A_{t,n}$ and $A_{t,s}$, respectively and $A_t = [A_{t,s} | A_{t,n}]$. Then we can decompose $\text{Cov}_t = \text{Cov}_{t,s} + \text{Cov}_{t,n}$, where $\text{Cov}_{t,s} = A_{t,s} A_{t,s}'$ is the signal

part of the covariance matrix and $\text{Cov}_{t,n} = A_{t,n}A_{t,n}'$ is the noise part of the covariance matrix. At any given time step, the signal part of the covariance matrix produces the useful non-random eigenstates and they should be used for data mining applications. Note that it suffices to compute only the eigenvalues (and eigenvectors) that fall outside the interval $[\lambda_{\min}, \lambda_{\max}]$, corresponding to the signal eigenstates. This allows computation of $\text{Cov}_{t,s}$ and hence $\text{Cov}_{t,n}$. The following section documents the performance of the EDS algorithm for detecting dependencies among Nasdaq 100 stocks.

4 Mobile Financial Data Stream Mining Using EDS

In order to study the properties of the proposed EDS technique in a systematic manner, we performed controlled experiments with Nasdaq 100 financial time-series data streams. The EDS is used to generate a “filtered” correlation matrix that detects the dependencies among different stocks. This section reports the experimental results.

The experiments reported here consider 99 companies from Nasdaq 100. We sample data every five minutes and each block of data is comprised of $m_i = 99$ rows. At any given moment the user is interested in the underlying dependencies observed from the current and previously observed data blocks.

First let us study the effect of the EDS-based filtering on the eigen-states produced by this financial time-series data. Figure 3 (Right) shows the distribution of eigenvalues from the covariance matrix (Cov_t) for different values of t . It also shows the theoretical lower and upper bounds ($\lambda_{\max}$ and $\lambda_{\min}$) at different time steps (i.e. increasing Q). The eigen-states falling outside these bounds are considered for constructing the “signal” or “information” part of the covariance matrix. The figure shows that initially a relatively large number of eigen states are identified as noisy. As time progresses and more data blocks arrive, the noise regime shrinks. It is also interesting to note that the EDS algorithm includes the lower end of the spectrum in the signal part. This is philosophically different from the traditional wisdom of considering only those eigen states with large eigen values.

In order to evaluate the online performance of the EDS algorithm we compare the eigen-space captured by the EDS at any given instant with respect to the “true” eigen-space defined by the underlying data distribution. Although data streams are conceptually infinite in size, the experiments documented in this section report results over a finite period of time. So we can benchmark the performance of the EDS with respect to the “true” eigen-space defined by the eigen vectors computed from the entire data set (all the data blocks) collected from the stream during the chosen period of observation. We first report the evolution of the signal-part of the covariance matrix as a function of time and compare that with the “true” covariance matrix generated from the entire data set. We report two different ways to compute this difference:

1. *RMSE*: The root mean square error (RMSE) is computed between the covariance matrices generated by the online EDS and the entire data set.

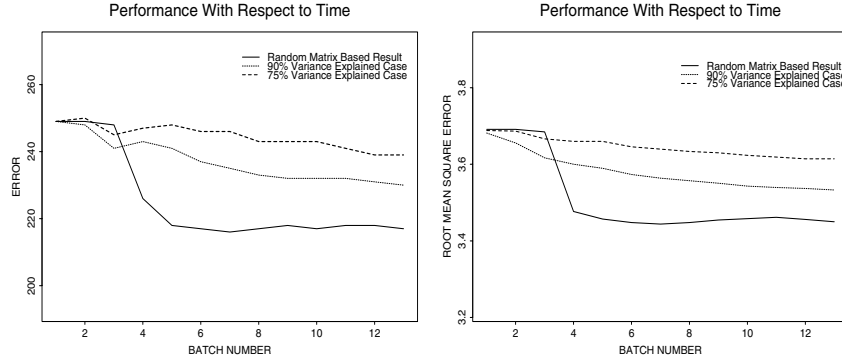


Fig. 4. The relative performance (thresholded error count in left and RMS error on right) of the EDS algorithm and the traditional approach using eigen vectors capturing 90% and 75% of the total variance. Different batch numbers correspond to different time steps

2. *Thresholded Error:* This measure first computes the difference between the estimated and true covariance matrices. If the (i, j) -th entry of the difference matrix is greater than some user given threshold θ then the value is set to 1 otherwise 0. The total number of 1's in this matrix is the observed *thresholded error* count.

Figure 4 compares the performance of our EDS algorithm with that of a traditional method that simply designates as signal, all the eigen-states that account for, respectively, 90% and 75% of the total variance. The latter corresponds a regular principal component analysis (PCA) that uses enough components to explain, respectively, 90% and 75% of the total variation in data. It shows the thresholded error count for each method, as a function of time (batch number). It is apparent that the EDS algorithm quickly outperforms the traditional method.

In order to evaluate the effect of filtering performed by the EDS algorithm, we picked a representative pair of stocks and compared the covariance between them, computed from the EDS filtered data and compare it with that using the raw data directly. We plot the absolute error between the estimated covariance value at each time step and the final covariance value.

Figure 5 depicts the comparison for the covariance between two stocks — Dell (DELL) and Amazon (AMZN). It is apparent that filtering using the EDS algorithm results in a smaller error overall.

5 Conclusions and Future Work

Although mobile computing devices are becoming more accessible and computationally powerful, their usage is still restricted to simple operations like web

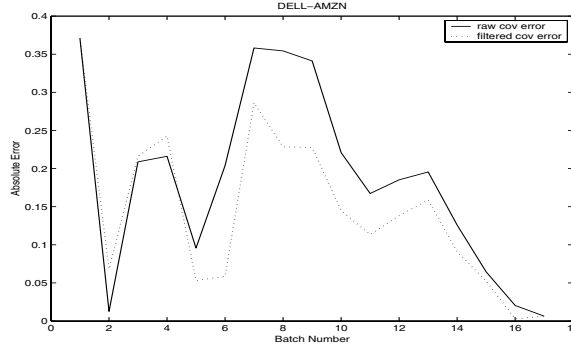


Fig. 5. Comparison of error in correlation estimates with and without EDS filtering for correlation between Dell and Amazon

surfing, checking emails, maintaining personal schedules, and taking notes. The limited battery power, restricted visual display area, and the low-bandwidth communication channel are hindering more sophisticated desktop-style applications. However, the ubiquitous aspect of these devices makes them very attractive for many time-critical applications. We need a new breed of applications for time-critical domains that can live with these resource restrictions.

This paper described one such application that performs online mining of financial data. It offered the EDS approach for extracting useful noise-free eigenstates from data streams. It shows that the EDS approach is better than the traditional wisdom of selecting the top-ranking eigenvectors guided by some user-given threshold. The EDS allows us to extract the eigenstates that correspond to non-random information that are likely to be useful from a data mining perspective. Indeed, any data mining technique, exploratory or otherwise, can be applied on the EDS filtered data.

The EDS approach works by comparing the empirically observed eigen distribution with the known distribution of random matrices. The theoretically known values of upper and lower limits of the spectrum are used to identify the boundary between noisy and signal eigen-states.

Another feature of our EDS approach is illustrated in Figure 3. As seen from the graph, the limits $\lambda_{\max}$ and $\lambda_{\min}$ both converge to 1 as the ratio Q tends to infinity (see also equation 1). In a data stream mining scenario, the number of features n is fixed, whereas the number of total number of observations m increases as each block of data is received. Hence, Q increases with time, which results in a smaller interval $[\lambda_{\max}, \lambda_{\min}]$ for the noisy eigenstates. This means that, as we observe more and more data, the EDS algorithm would potentially designate more eigenstates as signal. This is also intuitively satisfying, since most of the noise would get “averaged-out” as we observe more data.

In many real-world applications, the data stream is usually quasi-stationary. In other words, the data can be considered to be stationary over short periods

of time but the statistical behavior of the data changes — either gradually over time or due to some underlying event that triggers an abrupt change. These situations can be easily accommodated in the EDS framework. For example, one can use a fixed finite window of past observations to update the covariance matrix. Alternatively, an exponential weighting can be applied to past data blocks, thereby relying more on the recent data. Abrupt changes in the data distribution can be detected by monitoring the change in the covariance matrix or the subspace spanned by the noisy eigenstates, using an appropriate metric. Any significant deviation with respect to the past history would be an indication of an abrupt change. The EDS filter can be reset in such circumstances. We plan to pursue some of these ideas in a future publication.

Acknowledgments

The authors acknowledge supports from the United States National Science Foundation CAREER award IIS-0093353 and NASA (NRA) NAS2-37143. The authors would like to thank Li Ding for his contribution to related work and Qi Wang for performing some of the experiments.

References

1. M. L. Mehta. *Random Matrices*. Academic Press, London, 2 edition, 1991. 251
2. H. Kargupta, H. Park, S. Pittie, L. Liu, D. Kushraj, and K. Sarkar. MobiMine: Monitoring the stock market from a PDA. *ACM SIGKDD Explorations*, 3:37–47, 2001. 254
3. A. Azoff. *Neural Network Time Series Forecasting of Financial Markets*. Wiley, New York, 1994. 254
4. N. Baba and M. Kozaki. An intelligent forecasting system of stock price using neural networks. In *Proceedings IJCNN, Baltimore, Maryland*, pages 652–657, Los Alamitos, 1992. IEEE Press. 254
5. S. Cheng. A neural network approach for forecasting and analyzing the price-volume relationship in the taiwan stock market. Master’s thesis, National Jow-Tung University, Taiwan, R.O.C, 1994. 254
6. R. Kuo, L. Lee, and C. Lee. Intelligent stock market forecasting system through artificial neural networks and fuzzy delphi. In *Proceedings of World Congress on Neural Networks*, pages 345–350, San Deigo, 1996. INNS Press. 254
7. C. Lee. Intelligent stock market forecasting system through artificial neural networks and fuzzy delphi. Master’s thesis, Kaohsiung Polytechnic Institute, Taiwan, R.O.C, 1996. 254
8. J. Zirilli. *Financial Prediction Using Neural Networks*. International Thomson Computer Press, 1997. 254
9. J. Campbell, A. Lo, and A. MacKinley. *The Econometrics of Financial Markets*. Princeton University Press, USA, 1997. 254
10. G. Jang, F. Lsi, and T. Parng. Intelligent stock trading decision support system using dual adaptive-structure neural networks. *Journal of Information Science Engineering*, 9:271–297, 1993. 254

11. D. Jonsson. Some limit theorems for the eigenvalues of a sample covariance matrix. *Journal of Multivariate Analysis*, 12:1–38, 1982. 255
12. H. Kargupta, S. Ghosh, L. Ding, and K. Sivakumar. Random matrices and on-line principal component analysis from data streams. Submitted to The Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2002. 256

Long-Term Learning for Web Search Engines

Charles Kemp and Kotagiri Ramamohanarao

Department of Computer Science & Software Engineering
The University of Melbourne, Australia
{`cskemp,rao`}@`cs.mu.oz.au`

Abstract. This paper considers how web search engines can learn from the successful searches recorded in their user logs. Document Transformation is a feasible approach that uses these logs to improve document representations. Existing test collections do not allow an adequate investigation of Document Transformation, but we show how a rigorous evaluation of this method can be carried out using the referer logs kept by web servers. We also describe a new strategy for Document Transformation that is suitable for long-term incremental learning. Our experiments show that Document Transformation improves retrieval performance over a medium sized collection of webpages. Commercial search engines may be able to achieve similar improvements by incorporating this approach.

1 Introduction

Internet search engines are collecting thousands of user histories each day that could be exploited by machine learning techniques. To give a simple example, suppose that 70% of users who type the query ‘cola’ into Excite choose the third link presented on the results page. This statistic provides strong evidence that the third link should be promoted to the top of the list next time someone enters the same query.

Embedded within each user history is a set of semantic judgements. Long term research in Information Retrieval is directed towards the goal of giving machines the ability to make these judgements on their own. In the short term, however, progress is more easily achieved by taking advantage of the judgements users are already making as they interact with a collection.

Direct Hit (www.directhit.com) is the only current search engine that claims to adapt to these judgements. The owners of this system claims that it learns by monitoring which sites searchers select from the results page, and how much time the searchers spend at these sites [9]. As far as we are aware, however, the algorithms used by Direct Hit have never been published. We believe that algorithms for exploiting user histories are too valuable to be left to a single company, and that the best strategies will only be found if the research community develops an interest in this area.

A small-scale IR system might use many strategies for learning from its user histories. Over the past few decades, most of the popular approaches to machine

learning have been applied to Information Retrieval. [15] and [8] both provide surveys of this work, but to give just a few examples, neural networks [3,14], genetic algorithms [12], probabilistic models [11,17] and decision trees [8] have all been tried. [4] used clickthrough hierarchies to rank the documents.

None of these approaches, however, is an option for a large-scale system. Web search engines are already straining against the limitations of speed imposed by current technology. The additional burden imposed by any viable learning strategy must be small indeed.

Document Transformation is one strategy that is simple enough to become part of a large-scale search engine. It involves a modification of the space of document representations so that documents are brought closer to the queries to which they are relevant. Although proposed as early as 1971, Document Transformation has received little attention, and has never been adequately tested. [19, p 326] identifies the main reason for this neglect:

Document-space modification methods are difficult to evaluate in the laboratory, where no users are available to dynamically control the space modifications by submitting queries.

The development of the Internet has solved Salton's problem. Bringing users into the laboratory is still a problem, but all of a sudden it has become possible to bring the laboratory to the users. Millions of users are submitting millions of queries to web search engines every day. The user log of any of these search engines would provide ample data for a large-scale investigation of Document Transformation.

This paper, then, has two main goals. First, we hope to show that Document Transformation can improve web search engines. Second, Document Transformation is a research area in its own right, and we hope to use data from web search engines to evaluate it more rigorously than ever before.

2 The Vector Space Model

The algorithms for Document Transformation are built on top of the vector space model of Information Retrieval. An overview of this model will be given before Document Transformation is described in more detail.

The vector space model represents each document and query as a vector of concept weights, and computes the similarity between a document and a query by measuring the closeness of their corresponding vectors [19]. In the simplest case, every term is a separate concept. The similarity $S_{d,q}$ between document d and query q can therefore be calculated as $S_{d,q} = \sum_{t \in d \cap q} w_{d,t} \cdot w_{q,t}$ where $w_{d,t}$ and $w_{q,t}$ are the weights of term t in document d and query q . Given a query q , a ranked keyword search involves computing $S_{d,q}$ for all documents d in the collection, and returning the top-ranked documents to the user.

Many schemes for calculating $w_{d,t}$ and $w_{q,t}$ have been tried [24]. Most of these express the weight of a term in a document as a product of three factors:

Table 1. The BD-ACI-BCA similarity measure. W_d is the Euclidean length of the normalised document vector; W^a is the average value of W_d across the collection; f^m is the greatest value of f_t over the collection, and s is a constant with a typical value of 0.7 [22]

FACTOR	$w_{d,t}$	$w_{q,t}$
TF	$1 + \log_e f_{d,t}$	$1 + \log_e f_{q,t}$
IDF	1	$\log_e(1 + f^m/f_t)$
IDL	$1/((1-s) + s \cdot W_d/W^a)$	1

$w_{d,t} = TF \times IDF \times IDL$. The TF or ‘Term Frequency’ component is a function monotonic in $f_{d,t}$, the frequency of t in d . The IDF or ‘Inverse Document Frequency’ is monotonic in $1/f_t$, where f_t is the frequency of t across the entire collection. The IDL or ‘Inverse Document Length’ is monotonic in $1/W_d$, the reciprocal of the length of document d .

[24] found that none of the versions of $S_{d,q}$ that they tested consistently outperformed the others. The formulation they label BD-ACI-BCA, however, performed well across a range of measures, and gave the best overall performance for the precision at 20 metric. This formulation is described in Table 1, and will be used in the experiments described later.

3 Document Transformation

The hope underlying the vector space model is that a document vector will end up close to the vectors representing queries relevant to that document. Document Transformation aims to fix up cases where this goal is not achieved. Given a query and a document thought to be relevant to that query, Document Transformation is the process of moving the document vector towards the query vector.

Document Transformation was described in the late 1960s by the SMART team [5,10]. It is a close relative of Relevance Feedback, the process of refining a query vector by moving it towards documents identified as relevant in the hope that the move will also bring it closer to relevant documents that have not yet been identified [16,18]. One important difference between the two strategies is that Document Transformation alone leaves permanent effects on a system.

Several formulae for Document Transformation have been tested. One version (labelled DT1 for later reference) is:

$$D_{i+1} = (1 - \alpha)D_i + \alpha \frac{|D_i|}{|Q|} Q$$

where

D_i = the vector for the i th iteration of the document

Q = a query known to be relevant to the document
 $|Q|$ = the 1-norm of Q , that is, the sum of all weights in Q , and
 α is an experimental parameter

Document Transformation has been shown to improve retrieval performance over small and medium-sized collections [5,20]. There is no clear victor among the strategies that have been tried: different strategies perform best on different test collections [20].

4 The Test Collection

The TREC collections have become the clear first choice for researchers considering an experiment in information retrieval. Every TREC collection comes with a set of test queries, and a set of relevance judgements identifying the documents in the collection that are relevant to the test queries. Our experiments, however, required a set of user histories recording interactions with a collection. There are no standard test collections for which user histories are freely available, and we therefore decided to create our own.

The best source of user histories would be a log kept by a major search engine. Although we did not have access to one of these logs, we realised that user histories involving pages on servers at the University of Melbourne could be reconstructed from the logs kept locally. The collection chosen was therefore a set of 6644 web pages spidered from the Faculty of Engineering website at the University of Melbourne (www.ecr.mu.oz.au). User histories describing interactions with this collection were taken from the ‘referrer log’ (sic)¹ kept by the Engineering web server.

A referrer log contains information about transitions users have made between pages. Suppose that a user clicks on a link which takes her from one page to another. The HTTP standard allows the user’s browser to tell the second server the URL of the first page, and this information can be stored in the referrer log kept by the second server.

Transitions between the results page of a search engine and a page at the University of Melbourne are the only transitions relevant to this study. These transitions will be called ‘clickthroughs,’ and one example is:

```

http://www.google.com/search?hl=en&safe=off&q=
history+of+baritone+saxophone →
/~samuelrh/mushist.html

```

This clickthrough indicates that somebody found the page www.ecr.mu.oz.au/~samuelrh/mushist.html by searching for ‘history of baritone saxophone’ on Google. A simple script was written to extract queries from clickthroughs.

¹ The spelling error has become part of the HTTP standard.

4.1 Generating the Collection

The 6644 pages in the collection were spidered on September 20, 2001. All of the pages were passed through a HTML to ASCII converter. As far as we are aware, this is the largest collection that has ever been used for an investigation of Document Transformation.

4.2 The Test Set

Our test set was created by extracting queries from the 100 most recent click-throughs as of September 28. Two queries were later removed from the test set, since none of the systems tested returned any relevant documents in response to these queries. Duplicate queries were not removed from the test set: there are eight queries among the 98, for example, that refer to ‘Run DMC.’

Previous applications of machine learning to information retrieval have often suffered from inadequate test sets. For most test collections, relevance judgements are only provided for a small number of queries, and these queries are carefully chosen to overlap only slightly if at all. Redundancy can be achieved by including some, but not all of the training queries in the test set, but even this approach is not entirely satisfactory. Part of the redundancy should be due to queries that are related, but not identical. For example, ‘Bach organ music’ and ‘organ works j s bach’ are a pair of similar queries that are not identical. Generating such pairs would be a difficult task.

A better approach is to take samples of training and testing data directly from the domain under consideration. If these samples are independent, then the training and test sets should automatically contain just the right amount and type of redundancy. We have followed this approach by taking our training and test sets directly from the collection of authentic queries stored in the referer log.

4.3 The Training Set

Clickthroughs were collected over a seven week period beginning on September 6. After removing 100 clickthroughs to create the test set, a little over 4000 were left. We took the first 4000 of these for our training set.

Our training data is noisy, since a clickthrough does not necessarily indicate that a page is relevant to a query. As far as we know, all previous studies have used human-generated relevance judgements as a basis for Document Transformation. Previous studies have also used training sets much smaller than 4000 queries. The largest training set used by [5] or [20] contained only 125 queries. For both of these reasons, our work models the realities faced by web search engines more accurately than any previous study of Document Transformation.

4.4 Performance Measure

The need to generate relevance judgements for this collection and test set ruled out most of the standard metrics for assessing the performance of an IR system. We used the precision(10) metric: the average number of relevant pages

Table 2. Three strategies for Document Transformation. Q is a query relevant to document D . $|D|$ is the 1-norm of D : that is, the sum of the term weights in D

STRATEGY FORMULA	
DT1	$D = (1 - \alpha)D + \alpha \frac{ D }{ Q }Q$
DT2	$D = D + \beta Q$
DT3	$D = D_{original} + D_{learned}$, where
	$D_{learned} = \begin{cases} D_{learned} + \beta Q & \text{if } D_{learned} < l \\ (1 - \alpha)D_{learned} + \alpha \frac{ D_{learned} }{ Q }Q & \text{otherwise} \end{cases}$

among the top ten documents returned. [1] have argued that precision(10) is an appropriate metric for web search engines, since users rarely proceed past the first page of results. This metric, however, is less stable than most of the other standard metrics: if the test set is small, performance assessed using this metric may not accurately predict performance for other collections [7].

4.5 Relevance Judgements

Relevance judgements were made for the top ten documents returned by each system. The results for all experimental runs were shuffled before making these judgements. That is, when deciding whether a document was relevant to a query, we did not know whether the document had been returned by the control method, or a system that had supposedly learned, or both.

5 Systems Compared

A control system was implemented to provide a baseline for comparison. The control system has no access to the referer log, and carries out a ranked keyword search using the BD-ACI-BCA similarity measure. Other similarity measures were tried, but none led to superior results.

The remaining systems perform Document Transformation using click-throughs from the training set. Document Transformation can be implemented in many ways: the three tested here are given in Table 2.

Strategy DT1 ensures that the length of a document vector remains constant, where the length of a vector is defined as the sum of its weights. [5] and [20] both describe experiments where DT1 was found to improve retrieval performance.

Strategy DT2 is the simplest of the three: the terms in Q are weighted by β , then added to D . This strategy allows the length of a document vector to grow without bound.

Both of these strategies are susceptible to saturation of the document vectors. If enough clickthroughs are associated with a document, the effect of the terms

found in the original document becomes negligible. Previous studies of Document Transformation have not discussed this problem, probably because the training sets they use are small.

Document vector saturation, however, is a real problem for a search engine attempting to learn from thousands of queries each day. Strategy DT3 is designed to rule out this problem. Each document vector is the sum of two parts: the original part, and the learned part. The original part is the original document vector, and remains unchanged throughout the experiment. The learned part starts out as the zero vector, and is built up using a mix of strategies DT1 and DT2. DT2 is used until the vector has attained length l . After this point, DT1 is used and the length of the learned part remains constant.

This parameter l corresponds roughly to the ambition of a system. Large values of l mean that a system is prepared to modify document vectors by a large amount, and small values indicate a more conservative approach. When l is extremely large, DT3 is equivalent to DT2, and when l is zero, DT3 is equivalent to the control system.

All three strategies are computationally cheap, and could be used over large collections without a problem. [5] and [20] present several alternative strategies that we might have implemented, but we tested none of these for two main reasons. First, none of these alternative strategies has been shown to perform consistently better than DT1. Second, any properties of these strategies that have been identified in the past are unlikely to carry over to an experiment using a training set of several thousand queries. New strategies are needed to cope with large amounts of training data.

Each strategy can be applied before or after the normalisations required for the BD-ACI-BCA similarity measure (a linear transformation before normalisation is equivalent to a non-linear transformation of the normalised vectors). All of the results presented here will be for systems that perform Document Transformation first. Since the normalisation involves taking logarithms of term frequencies, these systems were less sensitive to small changes in the learning parameters than systems that normalised first.

5.1 Implementation

All of our systems were created by modifying the source code for version 11 of the SMART system. The SMART system is a public-domain implementation of the vector space model [6]. It is suitable for collections up to a few hundred megabytes in size, and has been designed for flexibility rather than efficiency. SMART performs stemming using Porter's algorithm, and allows stop words to be removed. All of our systems use both of these features.

6 Experimental Results

Figure 1(a) shows the performance of our three strategies as a function of the number of clickthroughs in the training set. α and β were chosen to maximise the performance of DT1 and DT2 for a training set of size 2000.

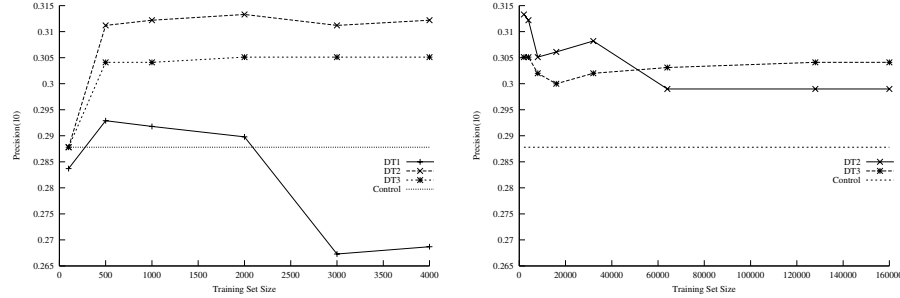


Fig. 1. Performance of the Document Transformation strategies when (a) training sets are small, and (b) for some large, artificial training sets. For DT1, $\alpha = 0.03$. For DT2, $\beta = 1.5$. For DT3, $\alpha = 0.03$, $\beta = 1.5$, and $l = 10$

DT1 has performed well over the small training sets used by previous studies, but suffers here from saturation of the document vectors. It is a little better than the control strategy for a training set of size 2000, which is not surprising since α was chosen to give the best possible performance in this situation. After this point, however, the performance of DT2 drops well below the control. Further evidence for saturation of the document vectors is provided by the first part of the DT1 curve. Even though α was chosen to maximise performance for a training set of size 2000, this value of α leads to superior performance for smaller training sets.

DT2 appears unaffected by document vector saturation, and achieves a stable improvement of around 8% over the control system.

When l is large — 1000, say — DT3 reduces to DT2 for training sets of several thousand clickthroughs. In Figure 1(a), l was set to 10 to limit the changes that could be made to the document vectors. DT3 is nearly as good as DT2 in this situation, showing that small changes to the document vectors are enough to account for most of the improvement achieved by DT2.

Even though a training set of size 4000 is larger than those considered by previous studies, it is still much smaller than the number of clickthroughs generated in a day by a popular search engine. To see how DT2 and DT3 would cope with large quantities of training data, we created some artificial training sets by concatenating copies of the 4000 genuine clickthroughs (the largest set contained 160,000 clickthroughs). The results of these experiments are presented in Figure 1(b).

Both DT2 and DT3 level out above the control system, but this time DT3 is the superior strategy. For systems learning from large numbers of clickthroughs, this result suggests that it is worth ensuring that document representations cannot be altered too drastically by the learning process.

Even though DT2 is out-performed by DT3, it does not appear to suffer from document vector saturation, probably because we are performing Docu-

ment Transformation before normalisation, and the normalisation involves logarithms of term frequencies. There should be some point after which DT2 will be susceptible to saturation, but a demonstration of this claim might require truly gigantic training sets.

7 Discussion

For large training sets, strategy DT3 achieved a stable improvement of around 6% over the control system. This improvement might have been even greater were it not for a serious problem with our test collection. Many of the pages mentioned most often in the referer log had been taken down for violating faculty guidelines about appropriate web use. Eight of the twenty most popular pages fall into this category, including the two most popular pages overall. Had they been part of the collection, these popular pages might have been expected to contribute most to the success of Document Transformation.

As a rule-of-thumb, [23] has suggested that a difference of more than 5% between IR systems is noticeable, and a difference of more than 10% is material. Our 6% improvement therefore suggests that Document Transformation is a viable strategy for improving the performance of web search engines.

7.1 Collaboration with a Commercial Search Engine

Further work in this area would profit from a large collection of web pages and a log recording interactions with that collection. The companies that own the major search engines are in the best position to meet this need. There is a good chance one of these companies would be prepared to collaborate in a large-scale study of Document Transformation: Lycos, Excite and AltaVista have all released query logs to selected researchers [2,13,21].

7.2 Better User Histories

There are many extensions of the approach presented here that would be worth investigating. Clickthroughs were the only user histories used for this study, but more complete histories would be valuable. For example, it would be useful to know how long a user spent reading a page, whether she saved it to disk or printed it, and whether she returned to the search engine to try a similar query. All of these factors would allow a more accurate judgement about whether her query was relevant to the page she found.

7.3 Dynamic Collections

We used a static collection for our experiments, but the web, of course, is dynamic. Working with a dynamic collection would require some adjustments to the strategies for Document Transformation investigated here. A clickthrough from five months ago, for example, is a less reliable guide to the contents of

a page than a clickthrough generated yesterday. A clickthrough's effect on a document representation should therefore fade with time.

Of the three strategies tested here, DT3 alone should adapt well to a dynamic collection. Once the 'learned part' of a document vector is full, later clickthroughs will reduce the impact of earlier clickthroughs. Other strategies, however, should also be tried.

If Document Transformation is successful, positive feedback loops will be created: the documents most likely to be returned will be those that have been looked at before. New documents may have little chance of breaking into one of these loops unless special measures are taken.

One way of dealing with this problem is to design a similarity measure that is sensitive to the age of a document. All other things being equal, newer documents should be regarded as more similar to a query than older documents.

7.4 Space Requirements

If Document Transformation is to be applied to large collections, the space occupied by the modified document vectors will need to be considered. To minimise storage requirements, it would be useful to set a limit on the number of terms that can be added to a vector (Strategy DT3 sets an upper bound on the length of the learned part, but the number of terms in a vector of fixed length can grow arbitrarily large as the weight of each becomes arbitrarily small). It would be worth investigating whether limiting the number of terms that can be added to a vector affects the success of Document Transformation.

8 Conclusion

Information retrieval systems may improve their performance by collecting and analysing user histories. Document Transformation is one instance of this approach based on the idea of moving a document vector towards a query known to be relevant to that document. Although Document Transformation was proposed many years ago, it has received little attention, probably because it was difficult to study in the pre-Internet era. This paper has shown how studies of Document Transformation can now easily be carried out using the logs kept by web search engines.

Our experiments have suggested that Document Transformation can improve retrieval performance over large collections of webpages. All of the user histories considered in this study were genuine: they were taken from a referer log kept by a web server at the University of Melbourne, and reflect the actions of actual web searchers rather than experimental stooges. To our knowledge, no other study of Document Transformation has attained this level of realism.

Previous strategies for Document Transformation have not been suitable for long-term use. They are susceptible to saturation of the document vectors: a process where the terms in the original document vector are gradually overpowered by terms that have later been associated with the document. This study has introduced a new strategy that overcomes this problem.

The ultimate test for Document Transformation will be whether it can improve the performance of one of the major search engines. All of the experiments described here could be repeated on a much larger scale using the logs collected by one of these search engines. Our results suggest that it would be worth collaborating with the developers of a commercial search engine on a large-scale study of Document Transformation.

References

1. Vo Ngoc Anh and Alistair Moffat. Improved retrieval effectiveness through impact transformation. In *Proceedings of the Thirteenth Australasian Database Conference*, Melbourne, Australia, in press. 268
2. Doug Beeferman and Adam Berger. Agglomerative clustering of a search engine query log. In *Proceedings of the Sixth International Conference on Knowledge Discovery and Data Mining*, pages 407–416, Boston, 2000. ACM Press. 271
3. Richard K. Belew. Adaptive information retrieval: Using a connectionist representation to retrieve and learn about documents. In *Proceedings of the Twelfth International Conference on Research and Development in Information Retrieval*, pages 11–20, Cambridge, MA, 1989. ACM Press. 264
4. Justin Boyan, Dayne Freitag, and Thorsten Joachims. A machine learning architecture for optimizing web search engines. In *Proceedings of the AAAI Workshop on Internet-Based Information Systems*. 1996. 264
5. T. Brauen. Document vector modification. In Gerard Salton, editor, *The SMART Retrieval System : Experiments in Automatic Document Processing*, pages 456–484. Prentice Hall, NJ, 1971. 265, 266, 267, 268, 269
6. Chris Buckley. Implementation of the SMART information retrieval system. Technical Report 85-686, Department of Computer Science, Cornell University, Ithaca, NY, 1985. 269
7. Chris Buckley and Ellen M. Voorhees. Evaluating evaluation measure stability. In *Proceedings of the Twenty Third Annual International Conference on Research and Development in Information Retrieval*, pages 33–40, Athens, Greece, 2000. ACM Press. 268
8. Hsinchun Chen. Machine learning for information retrieval: Neural networks, symbolic learning, and genetic algorithms. *Journal of the American Society of Information Science*, 46(3):194–216, 1995. 264
9. The Direct Hit popularity engine technology: A white paper, 1999. Available from www.directhit.com/about/products/technology_whitepaper.html. 263
10. S. Friedman, J. Maceyak, and S. Weiss. A relevance feedback system based on document transformations. In Gerard Salton, editor, *The SMART Retrieval System: Experiments in Automatic Document Processing*, pages 447–455. Prentice Hall, NJ, 1971. 265
11. Norbert Fuhr and Chris Buckley. A probabilistic learning approach for document indexing. *Information Systems*, 9(3):223–248, 1991. 264
12. M. Gordon. Probabilistic and genetic algorithms for document retrieval. *Communications of the ACM*, 31(10):1208–1218, 1988. 264
13. B. Jansen, A. Spink, J. Bateman, and T. Saracevic. Real life information retrieval: A study of user queries on the web. *SIGIR Forum*, 32(1):5–17, 1998. 271
14. K. L. Kwok. A neural network for probabilistic information retrieval. In *Proceedings of the Twelfth Annual International Conference on Research and Development in Information Retrieval*, pages 21–30, Cambridge, MA, 1989. 264

15. David D. Lewis. Learning in intelligent information retrieval. In Lawrence A. Birnbaum and Gregg C. Collins, editors, *Machine Learning: Proceedings of the Eighth International Workshop*, pages 235–239, Evanston, IL, 1991. Morgan Kaufmann. [264](#)
16. M. Maron and J. Kuhns. On relevance, probabilistic indexing and information retrieval. *Journal of the Association for Computing Machinery*, 7(3):216–244, July 1960. [265](#)
17. Benjamin Piwowarski. Learning in information retrieval: a probabilistic differential approach. In *Proceedings of the Twenty Second Annual Colloquium on Information Retrieval Research*, Cambridge, England, April 2000. [264](#)
18. J. Rocchio, Jr. Relevance feedback in information retrieval. In Gerard Salton, editor, *The SMART Retrieval System : Experiments in Automatic Document Processing*, pages 313–323. Prentice Hall, 1971. [265](#)
19. Gerard Salton. *Automatic text processing: the transformation, analysis, and retrieval of information by computer*. Addison-Wesley, Reading, MA, 1989. [264](#)
20. J. Savoy and D. Vrajitoru. Evaluation of learning schemes used in information retrieval. Technical Report CR-I-95-02, Faculty of Sciences, University of Neuchâtel, 1996. [266](#), [267](#), [268](#), [269](#)
21. Craig Silverstein, Monika Henzinger, Hannes Marais, and Michael Moricz. Analysis of a very large AltaVista query log. Technical Report 1998-014, Systems Research Center, Digital Equipment Corporation, Palo Alto, California, October 1998. [271](#)
22. Amit Singhal, Chris Buckley, and Mandar Mitra. Pivoted document length normalization. In H-P Frei, D. Harman, and P. Schäuble, editors, *Proceedings of the Nineteenth International Conference on Research and Development in Information Retrieval*, pages 21–29, New York, 1996. ACM Press. [265](#)
23. Karen Sparck Jones. Automatic indexing. *Journal of Documentation*, 30:393–432, 1974. [271](#)
24. Justin Zobel and Alistair Moffat. Exploring the similarity space. *SIGIR Forum*, 32(1):18–34, 1998. [264](#), [265](#)

Spatial Subgroup Mining Integrated in an Object-Relational Spatial Database

Willi Klösgen and Michael May

Fraunhofer Institute for Autonomous Intelligent Systems
Knowledge Discovery Team
D-53754 Sankt Augustin, Germany
{willi.kloesgen,michael.may}@ais.fraunhofer.de

Abstract. SubgroupMiner is an advanced subgroup mining system supporting multirelational hypotheses, efficient data base integration, discovery of causal subgroup structures, and visualization based interaction options. When searching for dependencies between subgroups and a target group, spatial subgroups with multirelational descriptions are explored. Search strategies of data mining algorithms are efficiently integrated with queries in an object-relational query language and executed in a database to enable scalability for spatial data.

1 Introduction: Mining Spatial Subgroups

The goal of spatial data mining is to discover spatial patterns and to suggest hypotheses about potential generators of such patterns. In this paper we focus on spatial patterns from the perspective of the subgroup mining paradigm. Subgroup Mining [7,8,9] is used to analyse dependencies between a target variable and a large number of explanatory variables. Interesting subgroups are searched that show some type of deviation, e.g. subgroups with an over proportionally high target share for a value of a discrete target variable, or a high mean for a continuous target variable.

This paper introduces the *SubgroupMiner*, an advanced subgroup mining system supporting multirelational hypotheses, efficient data base integration, discovery of causal subgroup structures, and visualization based interaction options. The goal is to provide a spatial mining tool applicable in a wide range of circumstances.

In this paper we focus on a representational issue that is at the heart of the whole approach: Representing spatial subgroups using an object-relational query language by embedding part of the search algorithm in a spatial database system (SDBS). Thus the data mining and the visualization in a Geographic Information System (GIS) share the same data. While this approach embraces the full complexity and richness of the spatial domain, most approaches to Spatial Data Mining export and pre-process the data from a SDBS. Our approach results in significant improvements in all stages of the knowledge discovery cycle:

- **Data Access:** Subgroup Mining is partially embedded in a spatial database, where analysis is performed. No data transformation is necessary and the same data is used for analysis and mapping in a GIS. This is important for the applicability of the system since pre-processing of spatial data is error-prone and complex.
- **Pre-processing and Analysis:** SubgroupMiner handles both numeric and nominal target attributes. For numeric explanatory variables on-the-fly discretisation is performed. Spatial and non-spatial joins are executed dynamically.
- **Post-processing and Interpretation:** Similar subgroups are clustered according to degree of overlap of instances to identify multicollinearities. A Bayesian network between subgroups can be inferred to support causal analysis.
- **Visualisation.** SubgroupMiner is dynamically linked to a GIS, so that spatial subgroups are visualized on a map. This allows the user to bring in background knowledge into the exploratory process, to perform several forms of interactive sensitivity analysis and to explore the relation to further variables and spatial features.

The paper is organized as follows. In Section 2, the representation of spatial data and spatial subgroups is discussed. Section 3 focuses on database integration using a sufficient statistics approach. Due to space restrictions, we will not elaborate on post-processing and visualization in this paper, but Section 4 puts the discussion into context by presenting an application example. Finally related work is summarized.

2 Representation of Spatial Data and of Spatial Subgroups

Representation of Spatial Data. Most modern Geographic Information Systems use an underlying Database Management System for data storage and retrieval. While both relational and object-oriented approaches exist, a hybrid approach based on object-relational databases is becoming increasingly popular. Its main features are:

- A *spatial data base* S is a set of relations $R_1, \dots, R_n$ such that each relation R_i in S has a geometry attribute G_i or an attribute A_i such that R_i can be linked (joined) to a relation R_k in S having a geometry attribute G_k .
- A *geometry attribute* G_i consists of ordered sets of x - y -coordinates defining points, lines, or polygons.
- Different types of spatial objects (e.g. streets, buildings) are organized in different relations R_i , called *geographical layers*. Each layer can have its own set of attributes $A_1, \dots, A_n$, called *thematic data*, and at most one geometry attribute G .
- This representation extends a purely relational scheme since the geometry attribute is non-atomic. One of its strengths is that a query can combine spatial information with attribute data describing objects located in space.

For querying multirelational spatial data a spatial database adds an operation called the *spatial join*. A spatial join links two relations each having a geometry attribute based on distance or topological relations (*disjoint*, *meet*, *equal*, *inside*, *contains*, *covers*, *coveredBy*, *overlap*) [2]. For supporting spatial joins efficiently, special purpose indexes like KD-trees or Quadrees are used.

Pre-processing vs. Dynamic Approaches. A GIS representation is a *multi-relational* description using non-atomic data types (the geometry) and applying operations from computational geometry to compute the relation between spatial objects. Since most machine learning approaches rely on single-relational data with atomic data types only, they are not directly applicable to this type of representation. To apply them, a possibility is to pre-process the data and to join relevant variables from secondary tables to a single target table with atomic values only. The join process may include spatial joins, and may use aggregation. The resulting table can be analysed using standard methods like decision trees or regression. While this approach may often be practical, it simply sidesteps the challenges posed by multi-object-relational datasets.

In contrast, Malerba et al. [17] pre-process data stored in a object-relational database to represent it in a *deductive database*. Thus, spatial intersection between objects is represented in a derived relation $\text{intersects}(X, Y)$. The resulting representation is still multi-relational, but only atomic values are permitted, and relationships in Euclidean space are reduced to qualitative relationships.

Extracting data from a SDBS S to another format has its disadvantages:

- The set of possible joins L between relations in S constrain the hypothesis space H . Since all spatial joins between geographical layers according to the topological relations described by Egenhofer [2] are meaningful, the set L is prohibitively large. If L includes the distance relation with a real-valued distance parameter, there are infinitely many possible joins. Thus, for practical and theoretical reasons, after pre-processing only part of the original space H will be represented in the transformed space H' , so that the best hypothesis in H may not be part of H' .
- Conversely, much of the pre-processing, which is often expensive in terms of computation and storage, may be unnecessary since that part of the hypothesis space may never be explored, e.g. because of early pruning.
- Pre-processing leads to redundant data storage, and in applications where data can change due to adding, deleting or updating, we suffer the usual problems of non-normalized data storage well-known from the database literature.
- Storing the respective data in different formats makes a tight integration between a GIS and the data mining method much more difficult to achieve.

An advantage of pre-processing is that once the data is pre-processed the calculation has not to be repeated, e.g. by constructing join indices [3]. However, a dynamic approach can get similar benefits from caching search results, and still have the original hypothesis space available.

For these reasons, our approach to spatial data mining relies on using a SDBS without transformation and pre-processing. Tables are dynamically joined. Variables are selected during the central search of a data mining algorithm, and inclusion depends on intermediate results of the process. Expensive spatial joins are performed only for the part of the hypothesis space that is really explored during search.

Spatial Subgroups. Subgroups are subsets of analysis objects described by selection expressions of a query language, e.g. simple conjunctive attributive selections, or multirelational selections joining several tables. *Spatial subgroups* are described by a spatial query language that includes operations on the spatial references of objects. A

spatial subgroup, for instance, consists of the enumeration districts of a city intersected by a river. A spatial predicate (*intersects*) operates on the coordinates of the spatially referenced objects *enumeration districts* and *rivers*.

Hypothesis Language. The *domain* is an object-relational database schema $S = \{R_1, \dots, R_n\}$ where each R_i can have at most one geometry attribute G_i . Multirelational subgroups are represented by a concept set $C = \{C_i\}$, where each C_i consists of a set of conjunctive attribute-value-pairs $\{C_i.A_1=v_1, \dots, C_i.A_n=v_n\}$ from a relation in S , a set of links $L=\{L_i\}$ between two concepts C_j, C_k in C via their attributes A_m, A_k , where the link has the form $C_i.A_m \theta C_k.A_m$, and θ can be '=', a distance or topological predicate (*disjoint, meet, equal, inside, contains, covered by, covers, overlap, interacts*).

For example, the subgroup “districts with high rate of migration and unemployment crossed by the M60” is represented as

$C = \{\{\text{district.migration=high, district.unemployment=high}\}, \{\text{road.name='M60'}\}\}$
 $L = \{\{\text{interacts(district.geometry, road.geometry)}\}\}$ Existential quantifiers of the links are problematic when many objects are linked, e.g. many persons living in a city or many measurements of a person. Then the condition that one of these objects has a special value combination will often not result in a useful subgroup. In this case, conditions based on *aggregates* such as counts, shares or averages will be more useful [12], [14].

These aggregation conditions are included by aggregation operations (*avg, count, share, min, max, sum*) for an attribute of a selector. An average operation on a numerical attribute additionally needs labeled intervals to be specified.

$C = (\text{district.migration} = \text{high}; \text{building.count(id)} = \text{high})$

$L = (\text{spatially\_interact}(\text{district.geometry}, \text{building.geometry}))$

Extension: Districts with many buildings.

For *buildings.count(id)*, labels *low, normal, high* and intervals are specified.

Multirelational subgroups have first been described in Wrobel [23] in an ILP setting. Our hypothesis language is more powerful due to numeric target variables, aggregations, and spatial links. Moreover, all combinations of numeric and nominal variables in the independent and dependent variables are permitted in the problem description. Numeric independent variables are discretised on the fly. This increases applicability of subgroup mining.

Representation of Spatial Subgroups in Query Languages. Our approach is based on an *object-relational* representation. The formulation of queries depends on non-atomic data-types for the geometry, spatial operators based on computational geometry, grouping and aggregation. None of these features is present in basic relational algebra or Datalog. An interesting theoretical framework for the study of spatial databases are constraint databases [15], which can be formulated as (non-trivial) extensions of relational algebra or Datalog. However, using SQL is more direct and much more practical for our purposes. The price to pay is that SQL extended by object-relational features is less amendable for theoretical analysis (but see [16]). For calculating spatial relationships spatial extensions of DBMS like *Oracle Spatial* can be used.

For database integration, it is necessary to express a multirelational subgroup as defined above as a query of a database system. The result of the query is a table representing the extension of the subgroup description. One part of this query defines the subset of the product space according to the l concepts and $l-1$ link conditions. The *from* part includes the l (not necessarily different) tables and the *where* part the $l-1$ link conditions (as they are given as strings or default options in the link specification; spatial extensions of SQL apply a special syntax for the spatial operations). Additionally the *where* part includes the conditions associated to the definition of selectors of concepts. Then the aggregation conditions are applied and finally the product space is projected to the target table (using the DISTINCT feature of SQL).

The complexity of the SQL statement is low for a single relational subgroup. Only the attributive selectors must be included in the *where* part of the query. For multirelational subgroups without aggregates and no distinction of multiple instances, the *from* part must manage possible duplicate uses of tables, and the *where* part includes the link conditions (transformed from the link specification) and the attributive selectors. For aggregation queries, a nested two-level select statement is necessary, first constructing the multirelational attributive part and then generating the aggregations. Multiple instances of objects of one table are treated by including the table in the *from* part several times and the distinction predicate in the *where* part.

The space of subgroups to be explored within a search depends on the specification of a relation graph which includes tables (object classes) and links. For spatial links the system can automatically identify geometry attributes by which spatial objects are linked, since there is at most one such attribute. A relation graph constrains the multirelational hypothesis space in a similar way as attribute selection constrains it for single relations.

3 Database Integration of Subgroup Mining

Subgroup Mining Search. This paper focuses on database integration of spatial subgroup mining. The basic subgroup mining algorithm is well-documented and only summarized here. Different subgroup patterns (e.g. for continuous or discrete target variables), search strategies and quality functions are described in [8,9].

The search is arranged as an iterated *general to specific, generate and test procedure*. In each iteration, a number of parent subgroups is expanded in all possible ways, the resulting specialized subgroups are evaluated, and the subgroups are selected that are used as parent subgroups for the next iteration step, until a prespecified iteration depth is achieved or no further significant subgroup can be found. There is a natural partial ordering of subgroup descriptions. According to the partial ordering, a specialization of a subgroup either includes a further selector to any of the concepts of the description or introduces an additional link to a further table.

The statistical significance of a subgroup is evaluated by a quality function. As a standard quality function, SubgroupMiner uses the classical binomial test to verify if the target share is significantly different in a subgroup:

$$\frac{p-p_0}{\sqrt{p_0(1-p_0)}} \sqrt{n} \sqrt{\frac{N}{N-n}} \quad (1)$$

This z-score quality function based on comparing the target group share in the subgroup (p) with the share in its complementary subset balances four criteria: size of subgroup (n), relative size of subgroup with respect to total population size (N), difference of the target shares ($p-p_0$), and the level of the target share in the total population (p_0). The quality function is symmetric with respect to the complementary subgroup. It is equivalent to the χ^2 -test of dependence between subgroup S and target group T , and the correlation coefficient for the (binary) subgroup and target group variables. For continuous target variables and the deviating mean pattern, the quality function is similar, using mean and variance instead of share p and binary case variance $p_0(1-p_0)$.

Evaluation of Contingency Tables. To evaluate a subgroup description, a contingency table is statistically analyzed (tab 1). It is computed for the extension of the subgroup description in the target object class. To get these numbers, a multirelational query is forwarded to the database. Contingency tables must be calculated in an efficient way for the very many subgroups evaluated during a search task.

Tab 1. Contingency table for target migration=high vs. unemployment=high

		Target		
		migration = high	¬migration = high	
Subgroup	unemployment=high,	16	19	35
	¬unemployment=high	47	496	543
		63	515	578

Sufficient Statistics Approach. We use a two-layer implementation [22], where evaluation of contingency tables is done in SQL, while the search manager is implemented in Java. A sufficient statistics approach is applied by which a single SQL query provides the aggregates that are sufficient to evaluate all successor subgroups. In the data server layer, within *one pass* over the database all contingency tables are calculated that are needed for the next search level. Thus not each single hypothesis queries the database, but a (next) population of hypotheses is treated concurrently to optimize data access and aggregation needed by these hypotheses. The search manager receives only aggregated data from the database so that network traffic is reduced. Besides offering scaling potential, such an approach includes the advantage of development ease, portability, and parallelization possibilities.

Construction of Query. The central component of the query is the selection of the multirelational parent subgroup. This is why representation of multirelational spatial subgroup in SQL is required. To generate the aggregations (cross tables) for a parent subgroup, a nested select-expression is applied for multirelational parents. From the product table, first the expansion attribute(s), key-attribute for the primary table and target attribute are projected and aggregates calculated for the projection. Then the cross tables (target versus expansion attribute) are calculated. Efficient calculation of

several cross tables, however, is difficult in SQL-implementations. An obvious solution could be based on building the union of several group-by operations (of target and expansion attributes). Although, in principle, several parallel aggregations could be calculated in one scan over the database, this is not optimised in SQL implementations. Indeed each union operation unnecessarily performs an own scan over the database. Therefore, to achieve a scalable implementation (at least for single relational and some subtypes of multirelational or spatial applications), the group-by operation has been replaced by explicit sum operations including case statements combining the different value combinations. Thus for each parent, only one scan over the database (or one joined product table) is executed. Further optimisations are achieved by combining those parents that are in the same joined product space (to eliminate unnecessary duplicate joins).

4 Application and Experiments

Application to UK Census Data. In this section we put the previous discussion in context. We describe a practical example that shows the interaction between spatial subgroup mining and a GIS mapping tool. The application has been developed within the IST-SPIN!-project, that integrates a variety of spatial analysis tools into a spatial data mining platform based on Enterprise Java Beans [18,19]. Besides Subgroup Mining these are Spatial Association rules [17], Bayesian Markov Chain Monte Carlo and the Geographical Analysis Machine GAM [20].

Our application are UK 1991 census data for Stockport, one of the ten districts in Greater Manchester, UK. Census data provide aggregated information on demographic attributes such as persons per household, cars per household, unemployment, migration, long-term-illness. Their lowest level of aggregation are so called *enumeration districts*. Also available are detailed geographical layers, among them streets, rivers, buildings, railway lines, shopping areas. Data are provided to the project by the partners Manchester University and Manchester Metropolitan University.

Assume we are interested in enumeration districts with a high migration rate. We want to find out how those enumeration districts are characterized, and especially what distinguishes them from other enumeration districts not having a high migration rate. Spatial subgroup discovery helps to answer this question by searching the hypothesis space for interesting deviation patterns with respect to the target attribute.

The target attribute T is then *high migration rate*. A concept C found in the search is *Enumeration districts with high unemployment crossed by a railway line*. Note that this subgroup combines spatial and non-spatial features. The deviation pattern is that the proportion of districts satisfying the target T is higher in districts that satisfy pattern C than in the overall population ($p(T|C) > p(T)$).

Another – this time purely spatial – subgroup found is *Enumeration district crossed by motorway M60*. This spatial subgroup induces a homogenous cluster taking the form of a physical spatial object. Spatial objects can often act as causal proxies for causally relevant attributes not part of the search space.

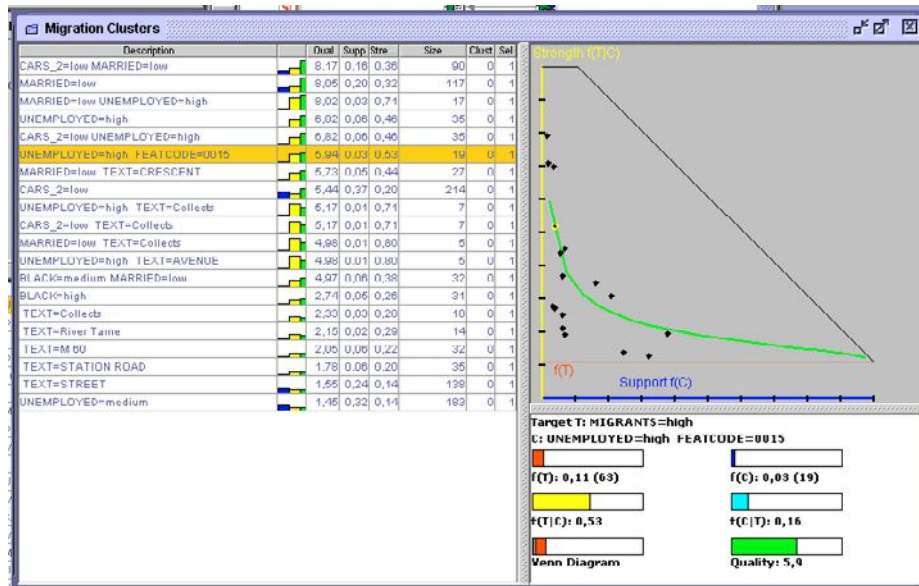


Fig 1. Overview on subgroups found showing the subgroup description (left). Bottom right side shows a detail view for the overlap of the concept C (e.g. located near a railway line) and the target attribute T (high unemployment rate). The window on the right top plots $p(T|C)$ against $p(C)$ for the subgroup selected on the left and shows *isolines* as theoretically discussed in [8]

A third – this time non-spatial – subgroup found is *Enumeration districts with low rate of households with 2 cars and low rate of married people*. By spotting the subgroup on the map we note that is a spatially inhomogeneous group, but with its center of gravity directed towards the center of Stockport.

The way data mining results are presented to the user is essential for their appropriate interpretation. We use a combination of cartographic and non-cartographic displays linked together through simultaneous dynamic highlighting of the corresponding parts. The user navigates in the list of subgroups (fig. 1), which are dynamically highlighted in the map window (fig. 2). As a mapping tool, the SPIN!-platform integrates the CommonGIS system [1], whose strengths lies in the dynamic manipulation of spatial statistical data. Figure 1 and 2 show an example for the migrant scenario, where the subgroup discovery method reports a relation between districts with high migration rate and high-unemployment.

Scalability Results. Spatial analysis is computationally demanding. In this section we summarize preliminary results on scalability. The simplest subgroup query provides all the information sufficient for the evaluation of all single relational successors of a set of single relational parent subgroup descriptions. These descriptions are constructed for one iteration step of specialization in the target object class including only attributes from this target class. Especially when the target object class contains many attributes that are used for descriptions of subgroups and the other (secondary) object classes contain much fewer attributes, these descriptions will constitute the main part of the search space.

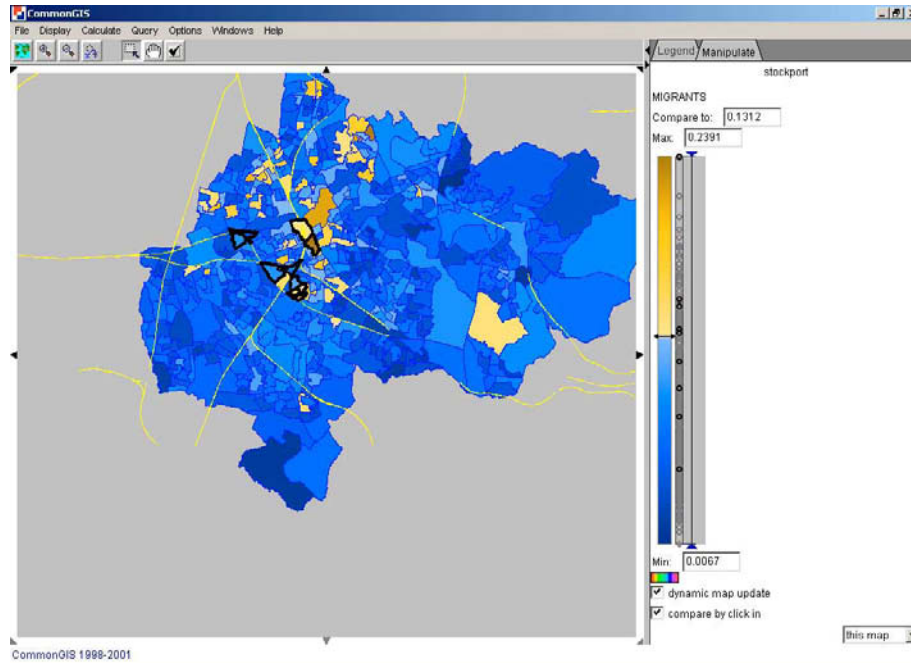


Fig. 2. Enumeration districts satisfying the subgroup description C (high unemployment rate and crossed by a railway line) are highlighted with a thicker black line. Enumeration districts also satisfying the target (high migration rate) are displayed in a lighter color

The performance requirements will strongly increase when multirelational subgroups are evaluated, because joins of several tables are needed. Two types of multirelational queries can be distinguished following two specialization possibilities. A multirelational subgroup can be specialized by adding a further conjunctive selector to any of its concepts or by adding a further concept in the concept sequence via a new link. The multirelational case involving many new links still requires many dynamic joins of tables and is not generally scalable.

The one-scan solution is nearly linear in the number of tuples in the one relational case (independent of the number of attributes, if this number is small thus that the cross table calculation is dominated by the organization of the scan. For many attributes, computation time is also proportional in the total number of discrete attribute values), and calculating sufficient statistics needs for large databases several orders less time than the version based on union operators which needs many scans. A detailed analysis of computation times for the different query versions and types of multirelational applications is performed in a technical report [11].

A further optimisation is achieved by substituting the parallel cross table calculation in SQL by a stored procedure, which is run (as the SQL query) in the database. It sequentially scans the (product) table and incrementally updates the cells of the cross tables. We are currently evaluating the performance of this solution compared with the SQL implementation. The SQL implementation, however, is easy portable to other data base systems. Only some specific expressions (case statement) must be adapted.

5 Related Work

Subgroup mining methods have been first extended for multirelational data by Wrobel [22]. SubgroupMiner allows flexible link conditions, an extended definition of multirelational subgroups including numeric targets, aggregate operations in links, spatial predicates, and is database integrated.

Knobbe et al. [12] and Krogel et al. [14], although in a non-spatial domain, apply a static pre-processing step that transforms a multirelational representation into a single table. Then, standard data mining methods such as decision trees can be applied. Static pre-processing typically has the disadvantages summarized in sec. 2 and must be restricted to avoid generating impractically large propositional target datasets.

Malerba and Lisi [17] apply an ILP approach for discovering association rules between spatial objects using first order logic (FOL) both for data and subgroup description language. They operate on a deductive relational database (based on Datalog) that extracts data from a spatial database. This transformation includes the pre-calculation of all spatial predicates, which as before ([12]) can be unnecessarily complex. Also the logic-based approach cannot handle numeric attributes and needs discretizations of numerical attributes of (spatial) objects. On the other side, the expressive power of Datalog allows to specify prior knowledge, e.g. in the form of rules or hierarchies. Thus the hypothesis language is in this respect more powerful than the hypothesis language in SubgroupMiner. In [17] the same UK census data set is used, as both approaches are developed within the scope of the IST-10536-SPIN! Project [4].

In [12] it is pointed out that aggregations (*count*, *min*, *avg*, *sum* etc.) are more powerful than ILP approaches to propositionalisation, which typically induce binary features, expressed in FOL restricting to existence aggregates.

Ester et al. [3] define neighborhood graphs and neighborhood indices as novel data structures useful for speeding up spatial queries, and show how several data mining methods can be built upon them. Koperski et al. [13] propose a two-level search for association rules, that first calculates coarse spatial approximations and performs more precise calculations to the result set of the first step.

Several approaches to extend SQL to support mining operations have been proposed, e.g. to derive sufficient statistics minimizing the number of scans [5]. Especially for association rules, a framework has been proposed to integrate the query for association rules in database queries [6]. For association rules, also architectures for coupling mining with relational database systems have been examined [21]. Siebes and Kersten [23] discuss approaches to optimize the interaction of subgroup mining (KESO) with DBMSs. While KESO still requires a large communication overhead between database system and mining tool, database integration for subgroup mining based on communicating sufficient aggregates has not been implemented before.

6 Conclusion and Future Work

Two-layer database integration of multirelational subgroup-mining search strategies has proven as an efficient and portable architecture. Scalability of subgroup mining for large datasets has been realized for single relational and multi-relational applications with a not complex relation graph. The complexity of a multirelational application mainly depends of the number of links, the number of secondary attributes to be selected, the depth of the relation graph, and the aggregation operations. Scalability is also a problem, when several tables are very large. Some spatial predicates are expensive to calculate. Then sometimes a grid for approximate (quick) spatial operations can be selected that is sufficiently accurate for data mining purposes.

We are currently investigating caching options to combine static and dynamic links, so that links can be declared as static in the relation graph. The join results are stored and need not be calculated again. The specification of textual link conditions and predicates in the relation graph that are then embedded into a complex SQL query has proven as a powerful tool to construct multirelational spatial applications.

Spatial analysis requires further advancements of subgroup mining systems. Basic subgroup mining methods discover correlations or dependencies between a target variable and explanatory variables. Spatial subgroups typically overlap with attributive subgroups. For the actionability of spatial subgroup mining results, it is important to analyze the causal relationships of these attributive and spatial variables. These relationships are analyzed by constraint-based Bayesian network techniques. Details of the causal analysis methods for subgroup mining are presented in [10].

Acknowledgments

Work was partly funded by the European Commission under IST-1999-10563 SPIN! – Spatial Mining for Data of Public Interest. We would like to thank Jim Petch, Keith Cole and Mohammed Islam from Manchester University and Chrissie Gibson from Manchester Metrop. Univ. for making available census data.

References

1. G. Andrienko, Andrienko, N. Interactive Maps for Visual Data Exploration, *International Journal of Geographical Information Science* 13(5), 355-374, 1999
2. M. J. Egenhofer. Reasoning about Binary Topological Relations, *Proc. 2nd Int. Symp. on Large Spatial Databases*, Zürich, Switzerland, 143-160, 1991
3. M. Ester, Frommelt, A., Kriegel, H.P., Sander, J. Spatial Data Mining: Database Primitives, Algorithms and Efficient DBMS Support, *Data Mining and Knowledge Discovery*, 2, 1999
4. IST-10536-SPIN!-project web site, <http://www.ccg.leeds.ac.uk/spin/>
5. G. Graefe, Fayyad, U., Chaudhuri, S. On the efficient gathering of sufficient statistics for classification from large SQL databases. *Proc. of the 4th Intern. Conf. on Knowledge Discovery and Data Mining*, Menlo Park: AAAI Press, 204-208, 1998

6. T. Imielinski, Virmani, A. A Query Language for Database Mining. *Data Mining and Knowledge Discovery*, Vol. 3, Nr. 4, 373–408, 2000
7. W. Klösgen. Visualization and Adaptivity in the Statistics Interpreter EXPLORA. In *Proceedings of the 1991 Workshop on KDD*, ed. Piatetsky-Shapiro, G., 25-34, 1991
8. W. Klösgen. Explora: A Multipattern and Multistrategy Discovery Assistant. *Advances in Knowledge Discovery and Data Mining*, eds. U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, Cambridge, MA: MIT Press, 249–271, 1996
9. W. Klösgen. Subgroup Discovery. Chapter 16.3 in: *Handbook of Data Mining and Knowledge Discovery*, eds. Klösgen, W., Zytkow, J., Oxford University Press, New York, 2002
10. W. Klösgen. Causal Subgroup Mining. To appear
11. W. Klösgen, May, M. Database Integration of Multirelational Subgroup Mining. Technical Report. Fraunhofer Institute AIS, Sankt Augustin, Germany, 2002
12. Knobbe, de Haas, M., Siebes, A. Propositionalisation and Aggregates. In *Proc. PKDD 2001*, eds. De Raedt, L., Siebes, A., Berlin:Springer, 277-288, 2001
13. K. Koperski, Adhikary, J. Han, J. Spatial Data Mining, Progress and Challenges, Vancouver, Canada, Technical Report, 1996
14. M. Krogel, Wrobel, S. Transformation-Based Learning Using Multirelational Aggregation, *Proc. ILP 2001*, eds. Rouveirol, C., Sebag, M., Springer, 142-155, 2001
15. G. Kuper, Libkin, L., Paredaens (eds.). *Constraint Databases*, Berlin:Springer, 2000
16. L. Libkin, Expressive Power of SQL, *Proc. of the 8th International Conference on Database Theory (ICDT01)*, eds. Bussche, J., Vianu, V., Berlin:Springer, 1-21, 2001
17. D. Malerba, Lisi, F.. Discovering Associations between Spatial Objects: An ILP Application. *Proc. ILP 2001*, eds. Rouveirol, C., Sebag, M., Berlin: Springer, 156-163, 2001
18. M. May. Spatial Knowledge Discovery: The SPIN! System. *Proc. of the 6th EC-GIS Workshop, Lyon*, ed. Fullerton, K., JRC, Ispra, 2000
19. M. May, Savinov, A. An Architecture for the SPIN! Spatial Data Mining Platform, *Proc. New Techniques and Technologies for Statistics, NTTS 2001*, 467-472, Eurostat, 2001
20. Openshaw, S., Turton, I., Macgill, J. and Davy, J. Putting the Geographical Analysis Machine on the Internet, in Gittings, B. (ed.) *Innovations in GIS 6*, 1999
21. S. Sarawagi, Thomas, S., Agrawal, R. Integrating Association Rule Mining with Relational Database Systems. *Data Mining and Knowledge Discovery*, 4, 89-125, 2000
22. Siebes, Kersten, M. KESO: Minimizing Database Interaction. *Proc. of the 3rd Intern. Conf. on Knowledge Discovery and Data Mining*, Menlo Park: AAAI Press, 247-250, 1998
23. S. Wrobel. An Algorithm for Multi-relational Discovery of Subgroups. In *Proc. of First PKDD*, eds. Komorowski, J., Zytkow, J., Berlin:Springer, 78-87, 1997

Involving Aggregate Functions in Multi-relational Search

Arno J. Knobbe^{1,2}, Arno Siebes², and Bart Marseille¹

¹Kiminkii,

Vondellaan 160, NL-3521 GH Utrecht, The Netherlands

a.knobbe@kiminkii.com,

b.marseille@kiminkii.com

²Utrecht University,

P.O. box 80 089, NL-3508 TB Utrecht, The Netherlands

siebes@cs.uu.nl

Abstract The fact that data is scattered over many tables causes many problems in the practice of data mining. To deal with this problem, one either constructs a single table by propositionalisation, or uses a Multi-Relational Data Mining algorithm. In either case, one has to deal with the non-determinacy of one-to-many relationships. In propositionalisation, aggregate functions have already proven to be powerful tools to handle this non-determinacy. In this paper we show how aggregate functions can be incorporated in the dynamic construction of patterns of Multi-Relational Data Mining.

1 Introduction

This paper presents a new paradigm for dealing with multi-relational data which involves aggregate functions. In [5, 6] the potential of these aggregate functions was demonstrated. In those papers, the aggregate functions were computed statically during a propositionalisation phase. Most MRDM algorithms compute new candidates dynamically. In our approach we combine the dynamic nature of MRDM with the power of aggregate functions by computing them on the fly.

The presented paradigm is centered around a new pattern-language that relies heavily on the use of aggregate functions. These aggregate functions are a powerful and generic way of dealing with the non-determinacy which is central to the complexity of Multi-Relational Data Mining (MRDM). Like many of the state-of-the-art MRDM approaches, we will use a top-down search that progressively ‘discovers’ relevant pieces of substructure by involving more tables. However, whenever a new table is involved over a one-to-many association, a range of aggregate functions (including the well-known existential expressions) can be applied to capture features of the local substructure. The proposed pattern language supports a mixture of existential expressions and aggregates.

The algorithms in [5,6] show that aggregate functions provide a unique way of characterizing groups of records. Rather than testing for the occurrence of specific

records in the group, which is the predominant practice in traditional algorithms, typically the group as a whole is summarized. It turns out that these algorithms work well on databases that combine a high level of non-determinacy with large numbers of (numeric) attributes.

Although producing good results, these propositionalisation approaches have some disadvantages. These are due to the fact that all multi-relational features are constructed statically during a preprocessing stage, before the actual searching is done. In contrast, most MRDM algorithms select a new set of relevant features dynamically, based on a subset of examples under investigation, say at a branch in a decision tree. Our approach overcomes the limitations of the propositionalisation approach, by introducing aggregate functions dynamically.

The work in this paper builds on a previously published MRDM framework [3]. The pattern language in this framework (Selection Graphs) will be extended to deal with our need for aggregation. Edges in the graph, which originally represented existential constraints, are annotated with aggregate functions that summarise the substructure selected by the connected sub-graph. Moreover, in [3] we also present a number of multi-relational data mining primitives in SQL. These primitives can be used to get statistics for a set of refinements using a single query to a database. Such primitives are an essential ingredient of a scalable data mining architecture. Section 5 presents a new collection of primitives to support the new constructs in our extended pattern language. Data mining primitives are especially important for the presented paradigm, as information about appropriate refinements involving aggregate functions can often only be obtained from the data, not from domain knowledge.

The structure of this paper is as follows. In Section 2 we introduce aggregate functions and give some important properties. In the next section we generalize Selection Graphs to include aggregate functions. The fourth section discusses the refinement operations on these Generalised Selection Graphs (GSG) and in Section 5 we show how these refinements can be tested using SQL. Our experimental results are given in Section 6. Our conclusions are formulated in the final section of this paper.

2 Aggregate Functions

Aggregate functions are functions on sets of records. Given a set of records and some instance of an aggregate function, we can compute a feature of the set which summarizes the set. We will be using aggregates to characterize the structural information that is stored in tables and associations between them. Consider two tables P and Q , linked by a one-to-many association A . For each record in P , there are multiple records in Q , and A thus defines a grouping over Q . An aggregate function can now be used to describe each group by a single value. Because we have a single value for every record in P , we can think of the aggregate as a virtual attribute of P . Given enough aggregate functions, we can characterize the structural information in A and Q . Note that, depending on the multiplicity of A , there may be empty groups in Q . These will be naturally treated by aggregates such as *count*. Other aggregates, such as *min*, will produce NULL values.

Aggregate functions typically (but not necessarily) involve not only the structural information in A , but also one or more attributes of Q . In theory we could even consider aggregate functions that include further tables connected to Q . For every association, we could thus define an infinite amount of aggregate functions. In order to obtain a manageable set of refinements per hypothesis, we will restrict the set of aggregate functions to the following list (all of which are part of standard SQL): *count*, *count distinct*, *min*, *max*, *sum* and *avg*. These functions are all real-valued, and work on either none or one attribute. It should be noted that we can still define large numbers of aggregate functions with this restricted list, by putting conditions on records in Q . However, these complex aggregate functions are discovered by progressive refinements.

Because we will consider varying sets of records during our search by adding a range of conditions, we will need to examine how the value of aggregate functions depends on different sets of records. The following observations will be relevant for selecting proper refinements (see section 4).

Definition 1.

- An aggregate function f is *ascending* if, given sets of records S and S' , $S' \subseteq S \Rightarrow f(S') \leq f(S)$.
- An aggregate function f is *descending* if, given sets of records S and S' , $S' \subseteq S \Rightarrow f(S') \geq f(S)$.

Lemma 1.

- Aggregate functions *count*, *count distinct* and *max* are ascending.
- Aggregate function *min* is descending.
- Aggregate functions *sum* and *avg* are neither ascending nor descending.

Example 1. In the mutagenesis problem database [8], there are tables for the concepts *molecule*, *atom* and *bond*. Promising aggregate functions to describe the relation between *molecule* and *atom* are: *count(atom)*, *min(atom.charge)*, *avg(atom.charge)* etc. We might also have similar functions for C-atoms in a molecule, or for C-atoms involved in a double bond. Clearly the count decreases if we only consider a subset of atoms, such as the C-atoms. In contrast, the minimum charge will increase.

3 Generalized Selection Graphs

We introduced Selection Graphs (SG) in [3, 4] as a graphical description of sets of objects in a multi-relational database. Objects are covered by a Selection Graph if they adhere to existential conditions, as well as attribute conditions. Every node in the graph corresponds to a table in the data model, and every edge corresponds to an association. If an edge connects two nodes, the corresponding association in the data model will connect two tables that correspond to the two nodes. The graphical structure of the Selection Graph thus reflects that of the underlying data model, and consequently the set of possible Selection Graphs is determined by the data model. For a structured example to be covered by a given SG, it will have to exhibit the same graphical structure as the SG. Next to the graphical conditions, an SG also holds

attribute conditions. Every node contains a, possibly empty, set of conditions on the attributes of the associated table. Finally, the target table in the data model, which identifies the target concept, appears at least once in any SG.

As [3] demonstrates, there is a simple mapping from Selection Graphs to SQL. A similar mapping to first-order logic exists. However, with these mappings we lose the intuitive graphical nature of the link between pattern language and declarative bias language. Selection Graphs are simply an intuitive means of presentation because of their close link with the underlying data model. Secondly, the refinement steps in the mining process are simple additions of the SGs. We will show how aggregate functions are a natural generalization of the existential conditions represented by edges in an SG, which makes Selection Graphs an ideal starting point for our approach.

To support aggregate functions with SGs we have to extend the language with a selection mechanism based on local structure. In particular, we add the possibility of *aggregate conditions*, resulting in Generalized Selection Graphs (GSG).

Definition 2. An *aggregate condition* is a triple (f, o, v) where f is an aggregate function, o a comparison operator, and v a value of the domain of f .

Definition 3. A *generalized selection graph* is a directed graph (N, E) , where N is a set of triples (t, C, s) , t is a table in the data model and C is a, possibly empty, set of conditions on attributes in t of type $t.a$ operator c ; the *operator* is one of the usual comparison operators such as $=$ and $>$. The flag s has the possible values *open* and *closed*. E is a set of tuples (p, q, a, F) where $p, q \in N$ and a is an association between $p.t$ and $q.t$ in the data model. F is a non-empty set of aggregate conditions with the same aggregate function. The generalized selection graph contains at least one node n_0 (the *root node*) that corresponds to the target table t_0 .

Generalized Selection Graphs are simply SGs with two important extensions. First, the use of edges to express existential constraints is generalized by adding an aggregate condition. Secondly, nodes may be either *open* or *closed*, which determines the possible refinements (see next section). A given Generalized Selection Graph (GSG) can be interpreted as follows. Every node n represents a set of records in a single table in the database. This set is governed by the combined restriction of the set of attribute conditions C and the aggregate conditions produced by each of the subgraphs connected to n . Each subgraph represents a similar set of records, which can be turned into a virtual attribute of n by the aggregate function. We thus have a recursive definition of the set of records represented by the root node. Each record corresponds to one example that is covered by the GSG. We will use the same recursive construct in a translation procedure for SQL, which is addressed in section 5.

Note that the simple language of Selection Graphs is contained in GSG. The concept of a selection edge [3] is replaced by the equivalent aggregate condition $(count, >, 0)$. In fact, we will use the directed edge without any aggregate condition as a purely syntactic shorthand for this aggregate condition.

4 Refinements

The majority of ILP and MRDM algorithms traverse a search space of hypotheses which is ordered by θ -subsumption [1,9]. This ordering provides a syntactic notion of generality: by simple syntactic operations to hypotheses we can derive clauses that are more specific. The two basic forms of operation are:

- add a condition to an existing node in the Selection Graph (roughly corresponds to a substitution in FOL).
- add a new node to the selection graph (add a literal to the clause).

These so-called refinements are a prerequisite to the top-down search which is central to so many Data Mining algorithms. This approach works smoothly under the following (often implicit) assumption: clause c is at least as general as clause c' if c θ -subsumes c' . That is, θ -subsumption is a good framework for generality. This assumption holds for the Selection Graphs, introduced in [3]. However, this is not the case for the Generalized Selection Graphs that we are considering in this paper. Simple substitutions of the above mentioned style do not necessarily produce more specific patterns.

Example 2. Consider the molecular database from example 1. Assume we are refining a promising hypothesis: the set of molecules with up to 5 atoms.



A simple substitution would produce the set of molecules with up to 5 C-atoms. Unfortunately, this is a superset of the original set. The substitution has not produced a specialization.

The problem illustrated by example 2 lies with the response of the aggregate condition to changes of the aggregated set resulting from added attribute conditions. Refinements to a subgraph of the GSG only work well if the related aggregate condition is *monotone*.

Definition 4. An aggregate condition (f, o, v) is *monotone* if, given sets of records S and S' , $S' \subseteq S$, $f(S') o v \Rightarrow f(S) o v$.

Lemma 2.

- Let $A = (f, \geq, v)$ be an aggregate condition. If f is ascending, then A is monotone.
- Let $A = (f, \leq, v)$ be an aggregate condition. If f is descending, then A is monotone.
- The following aggregate conditions are monotone: $(count, \geq, v)$, $(count\ distinct, \geq, v)$, $(max, \geq, v)$, $(min, \leq, v)$.

Lemma 2 shows that only a few aggregate conditions are safe for refinement. A subgraph of the GSG that is joined by a non-monotone aggregate condition should be left untouched in order to have only refinements that reduce the coverage. The flag s at each node stores whether nodes are safe starting points for refinements. *Closed* nodes will be left untouched. This leads us to the following collection of refinements:

- **add attribute condition.** An attribute condition is added to C of an *open* node.
- **add edge with aggregate condition.** This refinement adds a new edge and node to an *open* node according to an association and related table in the data model. The set F of the new edge contains a single aggregate condition. If the aggregate condition is monotone, the new node is *open*, and *closed* otherwise.
- **add aggregate conditions.** This refinement adds an aggregate condition to an existing edge. If any of the aggregate conditions is non-monotone, the related node will be *closed*, otherwise *open*.

Theorem 1. The operators **add attribute condition**, **add edge with aggregate condition**, and **add aggregate conditions** are proper refinements. That is, they produce less general expressions.

The above-mentioned classes of refinements imply a manageable set of actual refinements. Each *open* node offers opportunity for refinements by each class. Each attribute in the associated table gives rise to a set of **add attribute conditions**. Note that we will be using primitives (section 5) to scan the domain of the attribute, and thus naturally treat both nominal and numeric values. Each association in the data model connected to the current table gives rise to **add edge with aggregate conditions**. Our small list of SQL aggregate functions form the basis for these aggregate conditions. Finally, existing edges in the GSG can be refined by applying **add aggregate condition**.

5 Database Primitives

The use of database primitives for acquiring sufficient statistics about the patterns under consideration is quite widespread in KDD. A set of multi-relational primitives related to the limited language of Selection Graphs was given in [3]. The purpose of these primitives was to compute the support for a set of hypotheses by a single query to the database. These counts form the basis for the interestingness measures of choice, related to a set of refinements. A single query can for example assess the quality of an attribute condition refinement of a given Selection Graph, for all possible values of the attribute involved. The counts involved are always in terms of numbers of examples, i.e. records in the target table.

An elegant feature of the primitives related to Selection Graph is that they can be expressed in SQL using a single `SELECT` statement. In relational algebra this means that every primitive is counting over selections on the Cartesian product of the tables involved. Both the joins represented by selection edges, as well as the attribute conditions appear as simple selections. This works well for Selection Graphs because they describe combinations of particular records, each of which appears as a tuple in the Cartesian product. The Generalized Selection Graphs introduced in this paper are about properties of *groups* of records rather than the occurrence of individual combinations of records. Therefore the primitives for GSGs are computed using nested selections.

We start by introducing a basic construct which we will use to define our range of primitives. Every node in a GSG represents a selection of records in the associated

table. If we start at the leafs of the GSG and work back to the root, respecting all the selection conditions, we can compute the selection of records in the target table. This is achieved as follows. First we produce a list of groups of records in a table Q at a leaf node by testing on the aggregate condition. Each group is identified by the value of the foreign key:

```
SELECT foreign-key
FROM  $Q$ 
WHERE attribute-conditions
GROUP BY foreign-key
HAVING aggregate-conditions
```

We then join the result Q' with the parent table P to obtain a list of records in P that adheres to the combined conditions in the edge and leaf node :

```
SELECT  $P$ . primary-key
FROM  $P$ ,  $Q'$ 
WHERE  $P$ . primary-key =  $Q'$ . foreign-key
```

This process continues recursively up to the root-node, resulting in a large query of nested SELECT statements. The second query can be extended with the grouping construct of the first query for the next edge in the sequence. This results in exactly one SELECT statement per node in the GSG. This process is formalised in figure 1. The different primitives are all variations on this basic construct C .

5.1 CountSelection

The CountSelection primitive simply counts the number of examples covered by the GSG:

```
SELECT count(*)
FROM  $C$ 
```

5.2 Histogram

The Histogram primitive computes the distribution of values of the class attribute within the set of examples covered by the GSG.

```
SELECT class, count(*)
FROM  $C$ 
GROUP BY class
```

5.3 NominalCrossTable

The NominalCrossTable primitive can be used to determine the effect of an attribute refinement involving a nominal attribute on the distribution of class values. All pairs of nominal values and class values are listed with the associated count. As the nominal attribute typically does not appear in the target table, we cannot rely on the basic construct to produce the NominalCrossTable. Rather, we will have to augment the basic construct with bookkeeping of the possible nominal values.

```

SelectSubGraph (node  $n$ )
 $S = \text{'SELECT ' + } n.\text{Name}() + \text{'.' '}$ 
if ( $n.\text{IsRootNode}()$ )
     $S.\text{add} (n.\text{PrimaryKey}())$ 
else
     $S.\text{add} (n.\text{ForeignKey}())$ 
 $S.\text{add} (\text{' FROM ' + } n.\text{Name}())$ 
for each child  $i$  of  $n$  do
     $S_i = \text{SelectSubGraph}(i)$ 
     $S.\text{add} (\text{' , ' + } S_i + \text{' S' + } i)$ 

 $S.\text{add} (\text{' WHERE ' + } n.\text{AttributeConditions}())$ 
if ( $\neg n.\text{IsLeaf}()$ )
    for each child  $i$  of  $n$  do
         $S.\text{add} (\text{' AND ' + } n.\text{Name}() + \text{'.' ' + } n.\text{PrimaryKey}() +$ 
             $\text{' = ' + } S_i + \text{'.' ' + } i.\text{ForeignKey}())$ 

if ( $\neg n.\text{IsRootNode}()$ )
     $S.\text{add} (\text{' GROUP BY ' + } n.\text{Name}() + \text{'.' ' + } n.\text{ForeignKey}())$ 
     $S.\text{add} (\text{' HAVING ' + } n.\text{ParentEdge}().\text{AggregateCondition}())$ 

Return  $S$ 

```

Fig. 1. The select-subgraph algorithm

The idea is to keep a set of possible selections, one for each nominal value, and propagate these selections from the table that holds the nominal attribute, up to the target table. These selections all appear in one query, simply by grouping over the nominal attribute in the query which aggregates the related table. The basic construct is extended, such that an extra grouping attribute X is added to all the nested queries along the path from the rood-node to the node which is being refined:

```

SELECT  $X$ , foreign-key
FROM  $Q$ 
GROUP BY  $X$ , foreign-key
HAVING aggregate-conditions

```

The extra attribute in this query will reappear in all (nested) queries along the path to the rood-node. The actual counting is done in a similar way to the Histogram:

```

SELECT  $X$ , class, count(*)
FROM  $C$ 
GROUP BY  $X$ , class

```

5.4 NumericCrossTable

The NumericCrossTable primitive can be used to obtain a list of suitable numeric refinements together with the effect this has on the class-distribution. It is very similar to the NominalCrossTable, but requires yet a little more bookkeeping. This is because a given record may satisfy a number of numeric conditions, whereas it will only satisfy a single nominal condition. The process starts by producing a list of combinations of records and thresholds, and then proceeds as before. Note that there is a version of the NumericCrossTable for each numeric comparison operator.

```
SELECT b.X, a.foreign-key
FROM Q a, (SELECT DISTINCT X FROM Q) b
WHERE a.X ≤ b.X
GROUP BY b.X, a.foreign-key
HAVING aggregate-conditions
```

5.5 AggregateCrossTable

The AggregateCrossTable is our most elaborate primitive. It can be used to obtain a list of suitable aggregate conditions for a given aggregate function. The AggregateCrossTable primitive demonstrates the power of data mining primitives because the list of suitable conditions can only be obtained through extensive analysis of the database. It cannot be obtained from the domain knowledge or a superficial scan when starting the mining process.

We will treat candidate aggregate conditions as virtual attributes of the parent table. A first query is applied to produce this virtual attribute. Then the NumericCrossTable primitive is applied with the virtual attribute as the candidate. The first step looks as follows:

```
SELECT foreign-key, aggregate-function
FROM Q
GROUP BY foreign-key
```

6 Experiments

The aim of our experiments is to show that we can benefit from using a MRDM framework based on GSG rather than SG. If we can prove empirically that substantial results can be obtained by using a pattern-language with aggregate functions, then the extra computational cost of GSG is justified.

The previous sections propose a generic MRDM framework that is independent of specific algorithms. In order to perform our experiments however, we have implemented a rule-discovery algorithm that produces collections of rules of which the body consists of GSG and the head is a simple condition on the target attribute. The discovery algorithm offers a choice of rule evaluation measures as proposed in [7] of which two (Novelty and Accuracy) have been selected for our experiments. Rules are ranked according to the measure of choice.

6.1 Mutagenesis

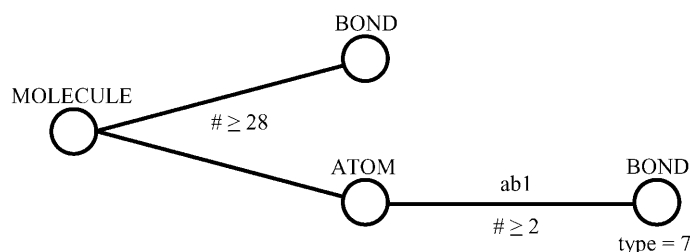
For our first experiment, we have used the Mutagenesis database with background ‘B3’ [8]. This database describes molecules falling in two classes, *mutagenic*, and *non-mutagenic*. The description consists of the atoms and the bonds that make up the compound. In particular, the database consists of 3 tables that describe directly the graphical structure of the molecule (**molecule**, **atom**, and **bond**). We have used the so-called ‘regression-friendly’ dataset for B3 that consists of 188 molecules, of which 125 (66.5%) are *mutagenic*.

Table 1. Summary of results for Mutagenesis

		Novelty		Accuracy	
		GSG	SG	GSG	SG
Best	Nov.	0.173	0.146	0.121	0.094
	Acc.	0.948	0.948	1.0	1.0
	Cov.	115	97	68	53
Top 10	Nov.	0.170	0.137	0.115	0.088
	Acc.	0.947	0.914	1.0	1.0
	Cov.	113.2	103.4	64.6	49.6

Table 1 presents a summary of our results. The left half gives the results when optimizing for Novelty. The right half does this for Accuracy. The top half gives the measures for the best rule, whereas the bottom half gives the averages for the best 10 rules. We see that all results for GSG are better than those of SG. In those cases where both SG and GSG have accuracy 1.0, which cannot be bettered, GSG has a far larger coverage.

A typical example of a result is given below. This GSG describes the set of all molecules that have at least 28 bonds as well as an atom that has at least two bonds of type 7. This result cannot be found with standard MRDM algorithms because of the condition of at least 28 bonds. It would also be difficult for the propositionalisation approaches because there first is a condition on the existence of an atom and then an aggregate condition on bonds of type 7 of that atom.



6.2 Financial

Our second experiment concerns the Financial database, taken from the Discovery Challenge organised at PKDD '99 and PKDD 2000 [10]. The database consists of 8 tables describing the operations of customers of a Czech bank. Among the 682 customers with a loan, we aim to identify subgroups with a high occurrence of bad

loans. Overall, 76 loans (11.1%) are bad. For all rules a minimum of 170 examples (25%) was required.

Table 2 summarizes the results obtained for Financial. Clearly, all results for GSG are significantly better than those for SG, in terms of both Novelty and Accuracy. Admittedly, the results were obtained with rules of lower Coverage, but this was not an optimisation criterion.

Table 2. Summary of results for Financial

		Novelty		Accuracy	
		GSG	SG	GSG	SG
Best	Nov.	0.052	0.031	0.051	0.025
	Acc.	0.309	0.190	0.314	0.196
	Cov.	178	268	172	199
Top 10	Nov.	0.050	0.027	0.047	0.023
	Acc.	0.287	0.178	0.297	0.185
	Cov.	194.5	281.8	171.4	208.3

7 Conclusion

In this paper we have presented a new paradigm for Multi-Relational Data Mining. The paradigm depends on a new way of capturing relevant features of local substructure in structured examples. This is achieved by extending an existing pattern language with expressions involving so-called aggregate functions. Although such functions are very common in relational database technology and are provided by all RDBMSes, they have only sporadically been used in a MRDM context. This paper shows that this is unfortunate and that aggregates are a powerful as well as natural way of dealing with the complex relationships between components of the structured examples under investigation. As our treatment concerns a generic framework based on an extended pattern language, it can be used to derive a range of top-down mining algorithms. We have performed experiments with a single rule discovery algorithm, but other approaches are clearly supported.

We have illustrated the usefulness of our paradigm on two well-known database, Mutagenesis [8] and Financial [10]. The experimental results show a significant improvement over traditional MRDM approaches. Moreover, the resulting patterns are intuitive and easily understandable. The positive results suggest future research into the benefits of non-top-down search strategies to overcome the current exclusion of non-monotone refinements.

References

1. Džeroski, S., Lavrač, N., An Introduction to Inductive Logic Programming, In [2]
2. Džeroski, S., Lavrač, N., *Relational Data Mining*, Springer-Verlag, 2001
3. Knobbe, A. J., Blockeel, H., Siebes, A., Van der Wallen, D.M.G. *Multi-Relational Data Mining*, In Proceedings of Benelearn '99, 1999

4. Knobbe, A. J. Siebes A, Van der Wallen D.M.G., *Multi-Relational Decision Tree Induction*. In proceedings of PKDD'99, LNAI 1704, pp. 378-383, 1999
5. Knobbe A. J., De Haas, M., Siebes, A., *Propositionalisation and Aggregates*, In Proceedings of PKDD 2001, LNAI 2168, pp. 277-288, 2001
6. Krogel, M. A., Wrobel, S., *Transformation-Based Learning Using Multi-relational Aggregation*, In Proceedings of ILP 2001, LNAI 2157, pp. 142-155. 2001
1. Lavrač, N., Flach, P., Zupan, B., *Rule Evaluation Measures: A Unifying View*, In Proceedings of ILP '99, 1999
8. Srinivasan, A., King, R.D., Bristol, D.W., *An Assessment of ILP-Assisted Models for Toxicology and the PTE-3 Experiment*, In Proceedings of ILP '99, 1999
9. Van Laer, W., De Raedt, L., *How to Upgrade Propositional Learners to First Order Logic: A Case Study*, In [2]
10. Workshop notes on Discovery Challenge PKDD '99, 1999

Information Extraction in Structured Documents Using Tree Automata Induction

Raymond Kosala¹, Jan Van den Bussche²,
Maurice Bruynooghe¹, and Hendrik Blockeel¹

¹ Katholieke Universiteit Leuven, Department of Computer Science
Celestijnenlaan 200A, B-3001 Leuven, Belgium
{Raymondus.Kosala,Maurice.Bruynooghe,Hendrik.Blockeel}@cs.kuleuven.ac.be

² University of Limburg (LUC), Department WNI, Universitaire Campus
B-3590 Diepenbeek, Belgium
jan.vandenbussche@luc.ac.be

Abstract. Information extraction (IE) addresses the problem of extracting specific information from a collection of documents. Much of the previous work for IE from structured documents formatted in HTML or XML uses techniques for IE from strings, such as grammar and automata induction. However, such documents have a tree structure. Hence it is natural to investigate methods that are able to recognise and exploit this tree structure. We do this by exploring the use of tree automata for IE in structured documents. Experimental results on benchmark data sets show that our approach compares favorably with previous approaches.

1 Introduction

Information extraction (IE) is the problem of transforming a collection of documents into information that is more readily digested and analyzed [6]. There are basically two types of IE: IE from unstructured texts and IE from (semi-) structured texts [15]. Classical or traditional IE tasks from unstructured natural language texts typically use various forms of linguistic pre-processing. With the increasing popularity of the Web and the work on information integration from the database community, there is a need for structural IE systems that extract information from (semi-) structured documents. Building IE systems manually is not feasible and scalable for such a dynamic and diverse medium as the Web [16]. Another problem is the difficulty in porting IE systems to new applications and domains if it is to be done manually. To solve the above problems, several machine learning techniques have been proposed such as inductive logic programming, e.g. [8,2], and induction of delimiter-based patterns [16,22,7,14,3]; methods that can be classified as grammatical inference techniques.

Some previous work on IE from structured documents [16,14,3] uses grammatical inference methods that infer regular languages. However structured documents such as HTML and XML documents (also annotated unstructured texts) have a tree structure. Therefore it is natural to explore the use of tree automata for IE from structured documents. Indeed, tree automata are well-established

and natural tools for processing trees [5]. An advantage of using the more expressive tree formalism is that the extracted field can depend on its structural context in a document, a context that is lost if the document is linearized into a string.

This paper reports on the use of k -testable tree languages, a kind of tree automata formalism, for the extraction of information from structured documents.

In Section 2 we recall how information extraction can be formulated as a grammatical inference problem and provide some background on tree automata and unranked tree languages. Then in Section 3 we describe our methodology and give the details of the k -testable tree algorithm we have used for our prototype implementation. Finally we present our experimental results in Section 4, discussion and related work in Section 5 and conclusions in Section 6.

2 Tree Grammar Inference and Information Extraction

2.1 Grammatical Inference

Grammatical inference refers to the process of learning rules from a set of labelled examples. It belongs to a class of inductive inference problems [20] in which the target domain is a formal language (a set of strings over some alphabet Σ) and the hypothesis space is a family of grammars. It is also often referred to as automata induction, grammar induction, or automatic language acquisition. It is a well-established research field in AI that goes back to Gold's work [11]. The inference process aims at finding a minimum automaton (the *canonical automaton*) that is *compatible* with the examples. The compatibility with the examples depends on the applied quality criterion. Quality criteria that are generally used are exact learning in the limit of Gold [11], query learning of Angluin [1] and probably approximately correct (PAC) learning of Valiant [24]. There is a large body of work on grammatical inference, see e.g. [20].

In regular grammar inference, we have a finite alphabet Σ and a regular language $L \subseteq \Sigma^*$. Given a set of examples that are in the language (S^+) and a (possibly empty) set of examples not in the language (S^-), the task is to infer a deterministic finite automaton (DFA) A that accepts the examples in S^+ and rejects the examples in S^- .

Following Freitag [7], we recall how we can map an information extraction task to a grammar inference task. We preprocess each document into a sequence of tokens (from an alphabet Σ). In the examples, the field to be extracted is replaced by the special token x . Then the learner has to infer a DFA for a language $L \subseteq (\Sigma \cup \{x\})^*$ that accepts the examples where the field to be extracted is replaced by x .

2.2 Tree Languages and Information Extraction

Given a set V of ranked labels (a finite set of function symbols with arities), one can define a set of trees, denoted as V^T , as follows: a label of rank 0 is a tree,

if f/n is a label of rank $n > 0$ and $t_1, \dots, t_n$ are trees, then $f(t_1, \dots, t_n)$ is a tree. We represent trees by ground terms, for example with $a/2, b/1, c/0 \in V$, the tree below is represented by the term $a(b(a(c, c)), c)$.

A deterministic tree automaton (DTA) M is a quadruple (V, Q, Δ, F) , where V is a set of ranked labels (a finite set of function symbols with arities), Q is a finite set of states, $F \subseteq Q$ is a set of final or accepting states, and $\Delta : \bigcup_k V_k \times Q^k \rightarrow Q$ is the transition function, where V_k denotes the subset of V consisting of the arity- k labels. For example, $\delta_k(v, q_1, \dots, q_k) \rightarrow q$, where $v/k \in V_k$ and $q, q_i \in Q$, represents a transition.

A DTA usually processes trees bottom up. Given a leaf labeled $v/0$ and a transition $\delta_0(v) \rightarrow q$, the state q is assigned to it. Given a node labeled v/k with children in state $q_1, \dots, q_k$ and a transition $\delta_k(v, q_1, \dots, q_k) \rightarrow q$, the state q is assigned to it. We say a tree is accepted when a tree has at least one node with an accepting state $q \in F$ assigned to it.

Grammatical inference can be generalized from string languages to tree languages. Rather than a set of strings over an alphabet Σ given as example, we are now given a set of trees over a ranked alphabet V . Rather than inferring a standard finite automaton compatible with the string examples, we now want to infer a compatible tree automaton. Various algorithms for this kind of tree automata induction have been developed (e.g., [18,19]).

A problem, however, in directly applying tree automata to tree-structured documents such as HTML or XML documents, is that the latter trees are “unranked”: the number of children of a node is not fixed by the label, but is varying. There are two approaches to deal with this situation:

1. The first approach is to use a generalized notion of tree automata towards unranked tree formalisms (e.g., [17,23]). In such formalisms, the transition rules are of the form $\delta(v, e) \rightarrow q$, where e is a regular expression over Q that describes a sequence of states.
2. The second approach is to encode unranked trees into ranked trees, specifically, binary trees, and to use existing tree automata inference algorithms for inducing the tree automaton.

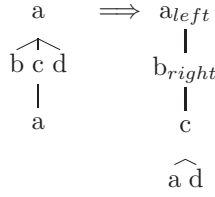
In this paper we follow the second approach, because it seems less complicated. An advantage is that we can use existing learning methods that work on ranked trees. A disadvantage is that we have to preprocess the trees before applying the algorithm.

Using the symbol T to denote unranked trees and F to denote a sequence of unranked trees (a forest), the following grammar defines unranked trees:

$$\begin{aligned} T &::= a(F), a \in V & F &::= \epsilon \\ & & F &::= T, F \end{aligned}$$

There are well-known methods of encoding unranked trees by binary trees which preserve the expressiveness of the original unranked trees. The one we use can be formally defined with the following recursive function *encode* (with *encode_f* for the encoding of forests):

$$\begin{aligned}
\text{encode}(T) &\stackrel{\text{def}}{=} \text{encode}_f(T) \\
\text{encode}_f(a(F_1), F_2) &\stackrel{\text{def}}{=} \begin{cases} a & \text{if } F_1 = F_2 = \epsilon \\ a_{\text{right}}(\text{encode}_f(F_2)) & \text{if } F_1 = \epsilon, F_2 \neq \epsilon \\ a_{\text{left}}(\text{encode}_f(F_1)) & \text{if } F_1 \neq \epsilon, F_2 = \epsilon \\ a(\text{encode}_f(F_1), \text{encode}_f(F_2)) & \text{otherwise} \end{cases}
\end{aligned}$$



Informally, the first child of a node v in an unranked tree T is encoded as the left child of the corresponding node v' of T' , the binary encoding of T , while the right sibling of a node v in tree T is encoded as the right child of v' in T' . To distinguish between a node with one left child and a node with one right child, the node is annotated with left and right respectively. For example, the unranked tree $a(b, c(a), d)$ is encoded into binary tree $a_{\text{left}}(b_{\text{right}}(c_{\text{right}}(a, d)))$, or pictorially shown on the left.

Note that the binary tree has exactly the same number of nodes as the original tree.

3 Approach and the Algorithm



On the left, a simplified view of a representative document from the datasets that we use for the experiment is shown.¹ In this dataset, the fields to be extracted are the fields following the “Alt. Name” and “Organization” fields. A document consists of a variable number of records. In each record the number of occurrences of the fields to be extracted is also variable (from zero to several occurrences). Also the position where they occur is not fixed. The fact that the fields to be extracted follow the “Alt. Name” and “Organization” field suggests that the task is not too difficult. However this turns out not to be the case as we can see from the results of several state-of-the-art systems in Section 4.

Our approach for information extraction has the following characteristics:

- Some IE systems preprocess documents to split them up in small fragments. This is not needed here as the tree structure takes care of this: each node has an annotated string. Furthermore the entire document tree can be used

¹ It is important to keep in mind that the figure only shows a rendering of the document, and that in reality it is a tree-structured HTML document.

as training example. This is different from some IE systems that only use a part of the document as training example.

- Strings stored at the nodes are treated as single labels. If extracted, the whole string is returned.
- A tree automaton can extract only one type of field, e.g. the field following 'Organization'. In order to extract multiple fields, a different automaton has to be learned for each field of interest.
- Examples used during learning contain a single node labeled with x . If the document contains several fields of interest, then several examples are created from it. In each example, one field of interest is replaced by an x .

The learning procedure is as follows:

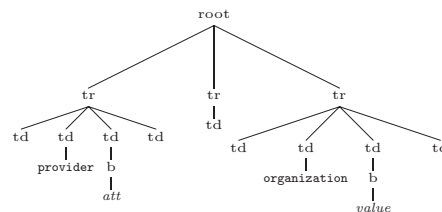
1. Annotate each example:
 - Replace the label of the node to be extracted by the special symbol x .
 - Parse the example document into a tree.
2. Run a tree automaton inference algorithm on the examples and return the inferred automaton.

The extraction procedure is as follows:

1. Parse the document into a tree.
2. Repeat for all text nodes:
 - Replace the text label of one text node by the special label x .
 - Run the automaton.
 - If the parse tree is accepted by the automaton, then output the original text of the node labeled with x .

Note that the extraction procedure can output several text nodes.

An implementation issue is how we deal with the contents of the various text nodes in the documents. The input to the algorithm consists of trees with *all* text strings at the leaves changed to 'CDATA'² except one that we call *distinguishing context*. The abstraction of the text strings to CDATA is done to get a generalization of the tree patterns of the information that we want to extract. This could be easily done when parsing. Representing each different text string as a separate label is undesirable since it would lead to over-specification. Roughly speaking a distinguishing context is the text content of a node that is 'useful' for the identification of the field of interest. An example of the usefulness of the distinguishing context can be seen in the following depiction of a document of the kind already shown at the beginning of this Section:



² CDATA is the keyword used in XML document type descriptions to indicate text strings [25].

Suppose we like to extract the field ‘value’ and the text label **organization** always precedes the field ‘value’. In such case we call the text label **organization** a *distinguishing context* (for the field ‘value’). If the labels **provider** and **organization** are both replaced by **CDATA** then any automaton that extracts the ‘value’ node will also extract the ‘att’ node. Indeed, the distinguishing context **provider** vs. **organization** has disappeared.

In our experiments we always use one distinguishing context for each field of interest when learning and testing the automaton. The distinguishing context is chosen automatically. Our method is to find the invariant text label that is nearest to the field of interest in the dataset. For example the text ‘Organization:’ is the invariant text label that is nearest to the organization name in the HTML document figure at the beginning of this Section. As distance measure we use the length of the shortest path in the document tree (for example the distance of a node to its parent is one; to its sibling, two; to its uncle, three.).

3.1 The k -Testable Algorithm

Our approach to information extraction using tree automata induction, presented in the previous section, can in principle be tried with any tree automata inference algorithm available. In our prototype implementation, we have chosen one of the more useful and practical algorithms available, namely, the k -testable algorithm [18]. This algorithm is parameterized by a natural number k , and the name comes from the notion of a “ k -testable tree language”. Informally, a tree language (set of trees) is k -testable if membership of a tree in the language can be determined just by looking at all the subtrees of length k (also intermediate ones). The k -testable algorithm is capable of identifying in the limit any k -testable tree language from positive examples only. Since information extraction typically has a locally testable character, it seems very appropriate to use in this context. The choice of k is performed automatically using cross-validation, choosing the smallest k giving the best results.

For the sake of completeness, we describe the algorithm here. We need the following terminology. Given a tree $t = v(t_1...t_m)$, $length(t)$ is the number of edges on the longest path between the root and a leaf. The (singleton) set $r_k(t)$ of root trees of length k is defined as:

$$r_k(v(t_1...t_m)) = \begin{cases} v & \text{if } k = 1 \\ v(r_{k-1}(t_1)...r_{k-1}(t_m)) & \text{otherwise} \end{cases} \quad (1)$$

The set $f_k(t)$ of fork trees of length k is defined as:

$$f_k(v(t_1...t_m)) = \bigcup_{j=1}^m f_k(t_j) \cup \begin{cases} \emptyset & \text{if } length(v(t_1...t_m)) < k - 1 \\ r_k(v(t_1...t_m)) & \text{otherwise} \end{cases} \quad (2)$$

Finally, the set $s_k(t)$ of subtrees of length k is defined as:

$$s_k(v(t_1 \dots t_m)) = \bigcup_{j=1}^m s_k(t_j) \cup \begin{cases} \emptyset & \text{if } \text{length}(v(t_1 \dots t_m)) > k-1 \\ v(t_1 \dots t_m) & \text{otherwise} \end{cases} \quad (3)$$

Example 1. For example, if $t = a(b(a(b, x)), c)$ then $r_2(t) = \{a(b, c)\}$; $f_2(t) = \{a(b, c), b(a), a(b, x)\}$; and $s_2(t) = \{a(b, x), b, x, c\}$.

The procedure to learn the tree automaton [18] is shown below. The algorithm takes as input a set of trees over some ranked alphabet V ; these trees serve as positive examples. The output is a tree automaton (V, Q, Δ, F) .

Let T be the set of positive examples.

$Q = \emptyset$; $F = \emptyset$; $\Delta = \emptyset$;

For each $t \in T$,

- Let $\mathcal{R} = r_{k-1}(t)$, $\mathcal{F} = f_k(t)$ and $\mathcal{S} = s_{k-1}(t)$.
- $Q = Q \cup \mathcal{R} \cup r_{k-1}(\mathcal{F}) \cup \mathcal{S}$
- $F = F \cup \mathcal{R}$
- for all $v(t_1, \dots, t_m) \in \mathcal{S}$: $\Delta = \Delta \cup \{\delta_m(v, t_1, \dots, t_m) = v(t_1, \dots, t_m)\}$
- for all $v(t_1, \dots, t_m) \in \mathcal{F}$: $\Delta = \Delta \cup \{\delta_m(v, t_1, \dots, t_m) = r_{k-1}(v(t_1, \dots, t_m))\}$

Example 2. Applying the algorithm on the term of Example 1 for $k = 3$, we obtain:

- $\mathcal{R} = r_2(t) = \{a(b, c)\}$, $\mathcal{F} = f_3(t) = \{a(b(a), c), b(a(b, x))\}$ and $\mathcal{S} = s_2(t) = \{a(b, x), b, x, c\}$.
- $Q = \{a(b, c), b(a), a(b, x), b, x, c\}$
- $F = \{a(b, c)\}$
- transitions:
 - $a(b, x) \in \mathcal{S} : \delta_2(a, b, x) = a(b, x)$
 - $b \in \mathcal{S} : \delta_0(b) = b$
 - $x \in \mathcal{S} : \delta_0(x) = x$
 - $c \in \mathcal{S} : \delta_0(c) = c$
 - $a(b(a), c) \in \mathcal{F} : \delta_2(a, b(a), c) = a(b, c)$
 - $b(a(b, x)) \in \mathcal{F} : \delta_1(b, a(b, x)) = b(a)$

With more (and larger) examples, more transitions are created and generalisation occurs: also trees different from the given ones will be labeled with an accepting state (a state from F).

4 Experimental Results

We evaluate the k -testable method on the following semi-structured data sets: a collection of web pages containing people's contact addresses which is called the Internet Address Finder (IAF) database and a collection of web pages about stock quotes which is called the Quote Server (QS) database. There are 10 example documents in each of these datasets. The number of fields to be extracted is respectively 94 (IAF organization), 12 (IAF alt name), 24 (QS date), and 25 (QS vol). The motivation to choose these datasets is as follows. Firstly they are benchmark datasets that are commonly used for research in information extraction, so we can compare the results of our method directly with the results of other methods. Secondly they are the only (online available) datasets that, to the best of our knowledge, require the extraction on the whole node of a tree and not a part of a node. These datasets are available online from RISE.³

We use the same criteria that are commonly used in the information retrieval research for evaluating our method. Precision P is the number of correctly extracted objects divided by the total number of extractions, while recall R is the number of correct extractions divided by the total number of objects present in the answer template. The F1 score is defined as $2PR/(P + R)$, the harmonic mean of P and R . Table 1 shows the results we obtained as well as those obtained by some current state-of-the-art methods: an algorithm based on Hidden Markov Models (HMMs) [10], the Stalker wrapper induction algorithm [16] and BWI [9]. The results of HMM, Stalker and BWI are adopted from [9]. All tests are performed with ten-fold cross validation following the splits used in [9]⁴. Each split has 5 documents for training and 5 for testing. We refer to the related work section for a description of these methods.

As we can see from Table 1 our method performs better in most of the test cases than the existing state-of-the-art methods. The only exception is the field date in the Quote Server dataset where BWI performs better. We can also see that the k -testable algorithm always gets 100 percent of precision. Like most algorithms that learn from positives only, k -testable generalises very cautiously, and thus is oriented towards achieving high precision rather than high recall. The use of a tree language instead of a string language, which increases the expressiveness of the hypothesis space, apparently makes it possible in these cases to avoid incorrect generalisations.

5 Discussion and Related Work

The running time of the k -testable algorithm in Section 3.1 is $O(k m \log m)$, where m is the total length of the example trees. The preprocessing consists of parsing, conversion to the binary tree representation (linear in the size of the document) and the manual insertion of the label x . Our prototype implementation was tested on a Pentium 166 Mhz PC. For the two datasets that we test

³ <http://www.isi.edu/~muslea/RISE/>

⁴ We thank Nicholas Kushmerick for providing us with the datasets used for BWI.

Table 1. Comparison of the results

	<i>IAF - alt. name</i>			<i>IAF - organization</i>			<i>QS - date</i>			<i>QS - volume</i>		
	<i>Prec</i>	<i>Recall</i>	<i>F1</i>	<i>Prec</i>	<i>Recall</i>	<i>F1</i>	<i>Prec</i>	<i>Recall</i>	<i>F1</i>	<i>Prec</i>	<i>Recall</i>	<i>F1</i>
HMM	1.7	90	3.4	16.8	89.7	28.4	36.3	100	53.3	18.4	96.2	30.9
Stalker	100	-	-	48.0	-	-	0	-	-	0	-	-
BWI	90.9	43.5	58.8	77.5	45.9	57.7	100	100	100	100	61.9	76.5
<i>k</i> -testable	100	73.9	85	100	57.9	73.3	100	60.5	75.4	100	73.6	84.8

above the average training time ranges from less than a second to some seconds for each k learned.

The time complexity of the extraction procedure is $O(n^2)$ where n is the number of nodes in the document. This runtime complexity depends on the number of nodes in the document where each time it has to substitute one of the nodes with x when running the automaton. For every node in the document tree the automaton has to find a suitable state for the node. With a suitable data structure for indexing the states the find operation on the states can be implemented to run in constant time. In our implementation the learned automata extract the document in seconds including preprocessing using a rudimentary indexing for the find operation.

Doing some additional experiments on various data, we learned that the value of k has a lot of impact on the amount of generalisation: the lower k the more generalisation. On the other hand, when the distance to the distinguishing context is large, then a large k is needed to capture the distinguishing context in the automaton. This may result in a too specific automaton having a low recall. In the future we plan to investigate methods to further generalise the obtained automaton.

There have been a lot of methods that have been used for IE problems, some are described in [15,22]. Many of them learn wrappers based on regular expressions. BWI [9] is basically a boosting approach in which the weak learner learns a simple regular expression with high precision but low recall. Chidlovskii et al. [3] describe an incremental grammar induction approach; their language is based on a subclass of deterministic finite automata that do not contain cyclic patterns. Hsu and Dung [14] learn separators that identify the boundaries of the fields of interest. These separators are described by strings of fixed length in which each symbol is an element of a taxonomy of tokens (with fixed strings on the lowest level and concepts such as punctuation or word at higher levels). The HMM approach in Table 1 was proposed by Freitag and McCallum [10]. They learn a hidden Markov model, solving the problem of estimating probabilities from sparse data using a statistical technique called shrinkage. This model has been shown to achieve state-of-the-art performance on a range of IE tasks. Freitag [7] describes several techniques based on naive-Bayes, two regular language inference algorithms, and their combinations for IE from unstructured texts. His results demonstrate that the combination of grammatical inference techniques with naive-Bayes improves the precision and accuracy of the extrac-

tion. The Stalker algorithm [16] induces extraction rules that are expressed as simple landmark grammars, which are a class of finite automata. Stalker performs extraction guided by a manually built *embedded catalog* tree, which is a tree that describes the structure of fields to be extracted from the documents. WHISK [22] is a system that learns extraction rules with a top-down rule induction technique. The extraction rules of WHISK are based on a form of regular expression patterns.

Compared to our method the methods mentioned above use methods to learn string languages while our method learns a more expressive tree language. Compared to HMMs and BWI our method does not require the manual specification of the windows length for the prefix, suffix and the target fragments. Compared to Stalker and BWI our method does not require the manual specification of the special tokens or landmarks such as “>” or “;”. Compared to Stalker our method works directly on document trees without the need for manually building the *embedded catalog* tree.

Despite the above advantages, there are some limitations of our method compared to the other methods. Firstly, the fact that our method only outputs the whole node seems to limit its application. One way to make our method more applicable is to do two level extraction. The first level extracts a whole node of the tree and the second extracts a part of the node using a string-based method. Secondly, our method works only on structured documents. This is actually a consequence of using tree automata inference. Indeed our method cannot be used for text-based IE, and is not intended for it. Thirdly, our method is slower than the string-based method because it has to parse, convert the document tree and substitute each node with x when extracting the document. Despite these limitations the preliminary results suggest that our method works better in the two structured domains than the more generally applicable string-based IE methods.

WHIRL is a ‘soft’ logic system that incorporates a notion of textual similarity developed in the information retrieval community. WHIRL has been used to implement some heuristics that are useful for IE in [4]. In this sense WHIRL is not a wrapper induction system but rather a logic system that is programmed with heuristics for recognizing certain types of structure in HTML documents. Hong and Clark [13] propose a technique that uses stochastic context-free grammars to infer a coarse structure of the page and then uses some user specified rules based on regular expressions to do a finer extraction of the page. Sakamoto et al. [21] propose a certain class of wrappers that use the tree structure of HTML documents and propose an algorithm for inducing such wrappers. They identify a field with a path from root to leaf, imposing conditions on each node in the path that relate to its label and its relative position among siblings with the same label (e.g., “2nd child with label ”). Their hypothesis language corresponds to a subset of tree automata.

Besides the k -testable algorithm proposed in this paper, we have also experimented with Sakakibara’s reversible tree algorithm [19]. Preliminary results with this algorithm suggested that it generalises insufficiently on our data sets, which is why we did not pursue this direction further.

6 Conclusion

We have motivated and presented a novel method that uses tree automata induction for information extraction from structured documents. We have also demonstrated on two datasets that our method performs better in most cases than the string-based methods that have been applied on those datasets. These results suggest that it is worthwhile to exploit the tree structure when performing IE tasks on structured documents.

As future work we plan to test the feasibility of our method for more general IE tasks on XML documents. Indeed, until now we have only performed experiments on standard benchmark IE tasks that can also be performed by the previous string-based approaches, as discussed in the two previous sections. However, there are tasks that seem clearly beyond the reach of string-based approaches, such as extracting the second item from a list of items, where every item itself may have a complex substructure. Of course, experimental validation remains to be performed. Interestingly, recent work by Gottlob and Koch [12] shows that all existing wrapper languages for structured document IE can be captured using tree automata, which strongly justifies our approach.

Other directions to explore are to incorporate probabilistic inference; to infer unranked tree automata formalisms directly; and to combine unstructured text extraction with structured document extraction.

Acknowledgements

We thank the anonymous reviewers for their helpful feedbacks. This work is supported by the FWO project query languages for data mining. Hendrik Blockeel is a post-doctoral fellow of the Fund for Scientific Research of Flanders.

References

1. D. Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, 1988. 300
2. M. E. Califf and R. J. Mooney. Relational learning of pattern-match rules for information extraction. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence and Eleventh Conference on Innovative Applications of Artificial Intelligence*, pages 328–334. AAAI Press / The MIT Press, 1999. 299
3. B. Chidlovskii, J. Ragetli, and M. de Rijke. Wrapper generation via grammar induction. In *11th European Conference on Machine Learning, ECML'00*, pages 96–108, 2000. 299, 307
4. W. W. Cohen. Recognizing structure in web pages using similarity queries. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence and Eleventh Conference on Innovative Applications of Artificial Intelligence*, pages 59–66, 1999. 308
5. H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. *Tree Automata Techniques and Applications*. Available on: <http://www.grappa.univ-lille3.fr/tata>, 1999. 300

6. J. Cowie and W. Lehnert. Information extraction. *Communications of the ACM*, 39(1):80–91, 1996. 299
7. D. Freitag. Using grammatical inference to improve precision in information extraction. In *ICML-97 Workshop on Automata Induction, Grammatical Inference, and Language Acquisition*, 1997. 299, 300, 307
8. D. Freitag. Information extraction from HTML: Application of a general learning approach. In *Proceedings of the Fifteenth Conference on Artificial Intelligence AAAI-98*, pages 517–523, 1998. 299
9. D. Freitag and N. Kushmerick. Boosted wrapper induction. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Innovative Applications of AI Conference*, pages 577–583. AAAI Press, 2000. 306, 307
10. D. Freitag and A. McCallum. Information extraction with HMMs and shrinkage. In *AAAI-99 Workshop on Machine Learning for Information Extraction*, 1999. 306, 307
11. E. M. Gold. Language identification in the limit. *Information and Control*, 10(5):447–474, 1967. 300
12. G. Gottlob and K. Koch. Monadic datalog over trees and the expressive power of languages for web information extraction. In *21st ACM Symposium on Principles of Database Systems*, June 2002. To appear. 309
13. T. W. Hong and K. L. Clark. Using grammatical inference to automate information extraction from the web. In *Principles of Data Mining and Knowledge Discovery*, pages 216–227, 2001. 308
14. C.-N. Hsu and M.-T. Dung. Generating finite-state transducers for semi-structured data extraction from the Web. *Information Systems*, 23(8):521–538, 1998. 299, 307
15. I. Muslea. Extraction patterns for information extraction tasks: A survey. In *AAAI-99 Workshop on Machine Learning for Information Extraction*, 1999. 299, 307
16. I. Muslea, S. Minton, and C. Knoblock. Hierarchical wrapper induction for semistructured information sources. *Journal of Autonomous Agents and Multi-Agent Systems*, 4:93–114, 2001. 299, 306, 308
17. C. Pair and A. Quere. Définition et étude des bilangages réguliers. *Information and Control*, 13(6):565–593, 1968. 301
18. J. Rico-Juan, J. Calera-Rubio, and R. Carrasco. Probabilistic k -testable tree-languages. In A. Oliveira, editor, *Proceedings of 5th International Colloquium, ICGI 2000, Lisbon (Portugal)*, volume 1891 of *Lecture Notes in Computer Science*, pages 221–228. Springer, 2000. 301, 304, 305
19. Y. Sakakibara. Efficient learning of context-free grammars from positive structural examples. *Information and Computation*, 97(1):23–60, 1992. 301, 308
20. Y. Sakakibara. Recent advances of grammatical inference. *Theoretical Computer Science*, 185(1):15–45, 1997. 300
21. H. Sakamoto, H. Arimura, and S. Arikawa. Knowledge discovery from semistructured texts. In S. Arikawa and A. Shinohara, editors, *Progress in Discovery Science - Final Report of the Japanese Discovery Science Project*, volume 2281 of *LNAI*, pages 586–599. Springer, 2002. 308
22. S. Soderland. Learning information extraction rules for semi-structured and free text. *Machine Learning*, 34(1-3):233–272, 1999. 299, 307, 308
23. M. Takahashi. Generalizations of regular sets and their application to a study of context-free languages. *Information and Control*, 27:1–36, 1975. 301
24. L. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984. 300

25. Extensible markup language (XML) 1.0 (second edition). W3C Recommendation 6 October 2000. www.w3.org. 303

Algebraic Techniques for Analysis of Large Discrete-Valued Datasets

Mehmet Koyutürk¹, Ananth Grama¹, and Naren Ramakrishnan²

¹ Dept. of Computer Sciences, Purdue University
W. Lafayette, IN, 47907, USA
{koyuturk,ayg}@cs.purdue.edu
<http://www.cs.purdue.edu/people/ayg>

² Dept. of Computer Science, Virginia Tech.
Blacksburgh, VA, 24061, USA
naren@cs.vt.edu
<http://people.cs.vt.edu/~ramakris/>

Abstract. With the availability of large scale computing platforms and instrumentation for data gathering, increased emphasis is being placed on efficient techniques for analyzing large and extremely high-dimensional datasets. In this paper, we present a novel algebraic technique based on a variant of semi-discrete matrix decomposition (SDD), which is capable of compressing large discrete-valued datasets in an error bounded fashion. We show that this process of compression can be thought of as identifying dominant patterns in underlying data. We derive efficient algorithms for computing dominant patterns, quantify their performance analytically as well as experimentally, and identify applications of these algorithms in problems ranging from clustering to vector quantization. We demonstrate the superior characteristics of our algorithm in terms of (i) scalability to extremely high dimensions; (ii) bounded error; and (iii) hierarchical nature, which enables multiresolution analysis. Detailed experimental results are provided to support these claims.

1 Introduction

The availability of large scale computing platforms and instrumentation for data collection have resulted in extremely large data repositories that must be effectively analyzed. While handling such large discrete-valued datasets, emphasis is often laid on extracting relations between data items, summarizing the data in an error-bounded fashion, clustering of data items, and finding concise representations for clustered data. Several linear algebraic methods have been proposed for analysis of multi-dimensional datasets. These methods interpret the problem of analyzing multi-attribute data as a matrix approximation problem. Latent Semantic Indexing (LSI) uses truncated singular value decomposition (SVD) to extract important associative relationships between terms (features) and documents (data items) [1]. Semi-discrete decomposition (SDD) is a variant of SVD, which restricts singular vector elements to a discrete set, thereby requiring less

storage [11]. SDD is used in several applications ranging from LSI [10] and bump-hunting [14] to image compression [17], and has been shown to be effective in summarizing data.

The main objective of this study is to provide an efficient technique for error-bounded approximation of large discrete valued datasets. A non-orthogonal variant of SDD is adapted to discrete-valued matrices for this purpose. The proposed approach relies on successive discrete rank-one approximations to the given matrix. It identifies and extracts attribute sets well approximated by the discrete singular vectors and applies this process recursively until all attribute sets are approximated to within a user-specified tolerance. A rank-one approximation of a given matrix is estimated using an iterative heuristic approach similar to that of Kolda et al. [10]. This approach of error bounded compression can also be viewed as identifying dominant patterns in the underlying data.

Two important aspects of the proposed technique are (i) the initialization schemes; and (ii) the stopping criteria. We discuss the efficiency of different initialization schemes for finding rank-one approximations and stopping criteria for our recursive algorithm. We support all our results with analytical as well as experimental results. We show that the proposed method is superior in identifying dominant patterns while being scalable to extremely high dimensions.

2 Background and Related Research

An $m \times n$ rectangular matrix A can be decomposed into $A = U\Sigma V^T$, where U is an $m \times r$ orthogonal matrix, V is an $n \times r$ orthogonal matrix and Σ is an $r \times r$ diagonal matrix with the diagonal entries containing the singular values of A in descending order. Here r denotes the rank of matrix A . The matrix $\tilde{A} = u\sigma_1v^T$ is a rank-one approximation of A , where u and v denote the first rows of matrices U and V respectively. If we think of a matrix as a multi-attributed dataset with rows corresponding to data items and columns corresponding to features, we can say that each 3-tuple consisting of a singular value σ_k , k^{th} row in U , and k^{th} row in V represents a pattern in A , whose strength is characterized by $|\sigma_k|$. The underlying data represented by matrix A is summarized by truncating the SVD of A to a small number of singular values. This method, used in Latent Semantic Indexing (LSI), finds extensive application in information retrieval [1].

Semi-discrete decomposition (SDD) is a variant of SVD, where the values of the entries in matrices U and V are constrained to be in the set $\{-1,0,1\}$ [11]. The main advantage of SDD is the small amount of storage required since each vector component requires only 1.5 bits. In our algorithm, since we always deal with 0/1 valued attributes, vector elements can be further constrained to the set $\{0,1\}$, requiring only 1 bit of storage. SDD has been applied to LSI and shown to do as well as truncated SVD using less than one-tenth the storage [10]. McConnell and Skillicorn show that SDD is extremely effective in finding outlier clusters in datasets and works well in information retrieval for datasets containing a large number of small clusters [14].

A recent thread of research has explored variations of the basic matrix factorization theme. Hofmann [8] shows the relationship between the SVD and an aspect model involving factor analysis. This allows the modeling of co-occurrence of features and data items indirectly through a set of *latent variables*. The solution to the resulting matrix factorization is obtained by expectation maximization (not by traditional numerical analysis). In [12], Lee and Seung impose additive constraints on how the matrix factors combine to model the given matrix; this results in what they call a ‘non-negative matrix factorization.’ They show its relevance to creating parts-based representations and handling polysemy in information retrieval.

Other work on summarizing discrete-attributed datasets is largely focused on clustering very large categorical datasets. A class of approaches is based on well-known techniques such as *vector-quantization* [4] and *k-means clustering* [13]. The *k-modes* algorithm [9] extends k-means to the discrete domain by defining new dissimilarity measures. Another class of algorithms is based on similarity graphs and hypergraphs. These methods represent the data as a graph or hypergraph to be partitioned and apply partitioning heuristics on this representation. Graph-based approaches represent similarity between pairs of data items using weights assigned to edges and cost functions on this similarity graph [3,5,6]. Hypergraph-based approaches observe that discrete-attribute datasets are naturally described by hypergraphs and directly define cost functions on the corresponding hypergraph [7,16]. Formal connections between clustering and SVD are explored in [2]; this thread of research focuses on first solving a continuous clustering relaxation of a discrete clustering problem (using SVD), and then subsequently relating this solution back via an approximation algorithm. The authors assume that the number of clusters is fixed whereas the dimensionality and the number of data items could change.

Our approach differs from these methods in that it discovers naturally occurring patterns with no constraint on cluster sizes or number of clusters. Thus, it provides a generic interface to the problem which may be used for in diverse applications. Furthermore, the superior execution characteristics of our approach make it particularly suited to extremely high-dimensional attribute sets.

3 Proximus: A Framework for Error-Bounded Compression of Discrete-Attribute Datasets

Proximus is a collection of algorithms and data structures that rely on modified SDD to find error-bounded approximations to discrete attributed datasets. The problem of error-bounded approximation can also be thought of as finding dense patterns in sparse matrices. Our approach is based on recursively finding rank-one approximations for a matrix A , i.e. finding two vectors x and y that minimize the number of nonzeros in the matrix $|A - xy^T|$, where x and y have size m and n respectively. The following example illustrates the concept:

Example 1

$$A = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} [1 \ 1 \ 0] = xy^T$$

Here, vector y is the *pattern vector*, which is the best approximation for the objective (error) function given. In our case, this vector is $[1 \ 1 \ 0]$. Vector x is the *presence vector* representing the rows of A that are well approximated by the pattern described by y . Since all rows contain the same pattern in this rank-one matrix, x is a vector of all ones. We clarify the discussion with a slightly non-trivial example.

Example 2

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \end{bmatrix} \approx \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \end{bmatrix} [0 \ 0 \ 1 \ 0 \ 1] = \begin{bmatrix} 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \end{bmatrix}$$

In this example, the matrix A is not a rank-one matrix as before. The pattern vector here is $[0 \ 0 \ 1 \ 0 \ 1]$ and the corresponding presence vector is $[1 \ 1 \ 0 \ 1]$. This presence vector indicates that the pattern is dominant in the first, second and fourth rows of A . A quick examination of the matrix confirms this. In this way, a rank-one approximation to a matrix can be thought of as decomposing the matrix into a pattern vector and a presence vector which signifies the presence of the pattern.

Using a rank-one approximation for the given matrix, we partition the row set of the matrix into sets A_0 and A_1 with respect to vector x as follows: the i^{th} row of the matrix is put into A_1 if the i^{th} entry of x is 1, it is put into A_0 otherwise. The intuition behind this approach is that the rows corresponding to 1's in the presence vector are the rows of a maximally connected submatrix of A . Therefore, these rows have more similar non-zero structures among each other compared to the rest of the matrix. This partitioning can also be interpreted as creating two new matrices A_0 and A_1 . Since the rank-one approximation for A gives no information about A_0 , we further find a rank-one approximation and partition this matrix recursively. On the other hand, we use the representation of the rows in A_1 in the pattern vector y to check if this representation is sufficient via some stopping criterion. If so, we decide that matrix A_1 is adequately represented by matrix xy^T and stop; else, we recursively apply the same procedure for A_1 as for A_0 .

3.1 Mathematical Formulation

The problem of finding the optimal rank-one approximation for a discrete matrix can be stated as follows.

Definition 1 Given matrix $A \in \{0, 1\}^m \times \{0, 1\}^n$, find $x \in \{0, 1\}^m$ and $y \in \{0, 1\}^n$ that minimize the error:

$$\|A - xy^T\|_F^2 = |\{a_{ij} \in |A - xy^T| : a_{ij} = 1\}|. \quad (1)$$

In other words, the error for a rank-one approximation is the number of non-zero entries in the residual matrix. For example, the error for the rank-one approximation of Example 2 is 4. As discussed earlier, this problem can also be thought of as finding maximum connected components in a graph. This problem is known to be NP-complete and there exist no known approximation algorithms or effective heuristics in literature.

Here, we use a linear algebraic method to solve this problem. The idea is directly adopted from the algorithm for finding singular values and vectors in the computation of an SVD. It can be shown that minimizing $\|A - xy^T\|_F^2$ is equivalent to maximizing the quantity $x^T Ay / \|x\|_2^2 \|y\|_2^2$ [11]. If we assume that y is fixed, then the problem becomes:

Definition 2 Find $x \in \{0, 1\}^m$ to maximize $x^T s / \|x\|_2^2$, where $s = Ay / \|y\|_2^2$.

This problem can be solved in $O(m + n)$ time as shown in the following theorem and corollary.

Theorem 1 If the solution to problem of Defn. 2 has exactly J non-zeros, then the solution is

$$x_j = \begin{cases} 1, & \text{if } 1 \leq j \leq J \\ 0, & \text{otherwise} \end{cases}$$

where the elements of s , in sorted order, are

$$s_{i_1} \geq s_{i_2} \geq \dots \geq s_{i_m}.$$

The proof can be found in [15].

Corollary 1 The problem defined in Defn. 2 can be solved in $O(m + n)$ time.

Proof

The entries of s can be sorted via counting sort in $O(n)$ time as the entries of $\|y\|_2^2 s = Ay$ are bounded from above by n and have integer values. Having them sorted, the solution described in Theorem 1 can be estimated in $O(m)$ time since $J \leq m$, thus the corollary follows. $\square$

The foregoing discussion also applies to the problem of fixing x and solving for y . The underlying iterative heuristic is based on the above theorem, namely we start with an initial guess for y , we solve for x and fix the resulting x to solve for y . We iterate in this way until no significant improvement can be achieved. The proposed recursive algorithm for summarizing a matrix can now be described formally as follows: Using a rank-one approximation, matrix A is split into two submatrices according to the following definition:

Definition 3 Given a rank-one approximation, $A \approx xy^T$, a split of A with respect to this approximation is defined by two sub-matrices A_1 and A_0 where

$$A(i) \in \begin{cases} A_1, & \text{if } x(i) = 1 \\ A_0, & \text{otherwise} \end{cases}$$

for $1 \leq i \leq m$. Here, $A(i)$ denotes the i^{th} row of A .

Then, both A_1 and A_0 are matrices to be approximated and this process continues recursively. This splitting-and-approximating process goes on until one of the following conditions holds.

- $h(A_1) < \epsilon$ where $h(A_1)$ denotes the hamming radius of A_1 , i.e., the maximum of the hamming distances of the rows of A_1 to the pattern vector y . ϵ is a pre-determined threshold.
- $x(i) = 1 \forall i$, i.e. all the rows of A are present in A_1 .

If one of the above conditions holds, the pattern vector of matrix A_1 is identified as a dominant pattern in the matrix. The resulting approximation for A is represented as $\tilde{A} = UV^T$ where U and V are $m \times k$ and $n \times k$ matrices containing the presence and pattern vectors of identified dominant patterns in their rows respectively and k is the number of identified patterns.

3.2 Initialization of Iterative Process

While finding a rank-one approximation, initialization is crucial for not only the rate of convergence but also the quality of the solutions since a wrong choice can result in poor local optima. In order to have a feasible solution, the initial pattern vector should have a magnitude greater than zero, i.e., at least one of the entries in the initial pattern vector should be equal to one. Possible procedures for finding an initial pattern vector include:

- **All Ones:** Set all entries of the initial pattern vector to one. This scheme is observed to be poor since the solution converges to a rough pattern containing most of the rows and columns in the matrix.
- **Threshold:** Set all entries corresponding to columns that have nonzero entries more than a selected threshold to one. The threshold can be set to the average number of nonzeros per column. This scheme can also lead to poor local optima since the most dense columns in the matrix may belong to different independent patterns.
- **Maximum:** Set only the entry corresponding to the column with maximum number of nonzeros to one. This scheme has the risk of selecting a column that is shared by most of the patterns in the matrix since it typically has a large number of nonzeros.
- **Partition:** Take the column which has nonzeros closest to half of the number of rows and select the rows which have a nonzero entry on this column. Then apply the *threshold* scheme taking only the selected rows into account.

This approach initializes the pattern vector to the center of a roughly identified cluster of rows. This scheme has the nice property of starting with an estimated pattern in the matrix, thereby increasing the chance of selecting columns that belong to a particular pattern.

All of these schemes require $O(m + n)$ time. Our experiments show that *partition* performs best among these schemes as intuitively expected. We select this scheme to be the default for our implementation and the experiments reported in Section 4 are performed with this initialization. More powerful initialization schemes can improve the performance of the algorithm significantly. However, it is important that the initialization scheme must not require more than $\Theta(m + n)$ operations since it will dominate the runtime of the overall algorithm if it does.

3.3 Implementation Details

As the proposed method targets handling vectors of extremely high dimensions, the implementation and design of data structures are crucial for scalability, both in terms of time and space. In our implementation, we take advantage of the discrete nature of the problem and recursive structure of the proposed algorithm.

Data Structures The discrete vectors are stored as binary arrays, i.e., each group of consecutive W (word size) entries are stored in a word. This allows us to reduce the memory requirement significantly as well as to take advantage of direct binary operations used in matrix-vector multiplications. The matrices are stored in a row-compressed format which fits well to the matrix-splitting procedure based on rows. Figure 1 illustrates the row-compressed representation of a sample matrix A and the result of splitting this matrix into A_0 and A_1 ,

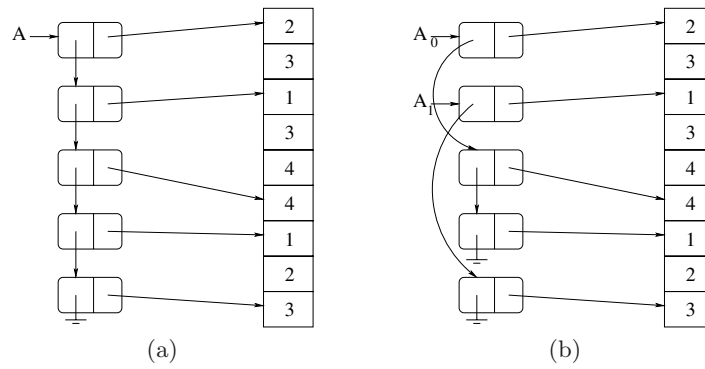


Fig. 1. Illustration of underlying data structure: (a) Original matrix (b) Resulting matrices after split, in row-compressed format

where

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad A_0 = \begin{bmatrix} A(1) \\ A(3) \\ A(4) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \end{bmatrix} \quad A_1 = \begin{bmatrix} A(2) \\ A(5) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

In this format, the column id's of the nonzero entries are stored in an array such that the non-zero entries of a row are stored in consequent locations. The list of rows of the matrix is a linked list in which each row has an additional pointer to the start of its non-zero entries in the non-zero list. Since the columns of the original matrix are never partitioned in our recursive implementation, the matrices appearing in the course of the algorithm can be easily created and maintained. While splitting a matrix, it is only necessary to split the linked list containing the rows of the matrix as seen in Figure 1(b). This is particularly important as splitting large sparse structures can be a very significant (and often dominant) overhead as we learnt from our earlier implementations.

Matrix Computations The heuristic used to estimate rank-one approximations necessitates the computation of Ax and $A^T y$ alternately. Although a row-compressed format is suitable for the computation of Ax , it is more difficult to perform the computation of $A^T y$ since each column of A is multiplied with y in this operation. However, it is not necessary compute A^T to perform this operation. In our implementation, we compute $s = A^T y$ with the following algorithm based on a row-compressed format:

```

initialize  $s(i) = 0$  for  $1 \leq i \leq n$ 
for  $i \leftarrow 1$  to  $m$  do
  if  $y(i) = 1$  then
    for  $\forall j \in \text{nonzeros}(A(i))$  do
       $s(j) \leftarrow s(j) + 1$ 

```

This algorithm simply multiplies each row of A with the corresponding entry of y and adds the resulting vector to s and requires $O(nz(A))$ time in the worst-case.

This combination of restructured matrix transpose-vector product and suitable core data structures makes our implementation extremely fast and scalable.

Stopping Criteria As discussed in Section 3.1, one of the stopping criteria for the recursive algorithm is if each row in the matrix is well approximated by the pattern (i.e., the column singular vector is a vector of all ones). However, this can result in an undesirable local optimum. Therefore, in this case we check if $h(A) < \epsilon$, i.e., the hamming radius around the pattern vector for all the row vectors is within a prescribed bound. If not, we further partition the matrix into

two based on hamming distance according to the following rule.

$$A(i) \in \begin{cases} A_1, & \text{if } h(A(i), y) < r \\ A_0, & \text{otherwise} \end{cases}$$

for $1 \leq i \leq m$. Here $h(A(i), y)$ denotes the hamming distance of row i to the pattern vector y and r is the prescribed radius of the virtual cluster defined by A_1 . The selection of r is critical for obtaining the best cluster in A_1 . In our implementation, we use an adaptive algorithm that uses a sliding window in n -dimensional space and detects the center of the most sparse window as the boundary of the virtual cluster.

4 Experimental Results

In order to illustrate the effectiveness and computational efficiency of the proposed method, we conducted several experiments on synthetic data specifically generated to test the methods on problems where other techniques typically fail (such as overlapping patterns, low signal-to-noise ratios). In this section we present the results of execution on a number of test samples to show the approximation capability of Proximus, analyze the structure of the patterns discovered by Proximus and demonstrate the scalability of Proximus in terms of various problem parameters.

4.1 Data Generation

Test matrices are generated by constructing a number of patterns, each consisting of several distributions (mixture models). Uniform test matrices consist of uniform patterns characterized by a set of columns that may have overlapping patterns. For example, the test matrix of Figure 2(a) contains four patterns with column sets of cardinality 16 each. A pattern overlaps with at most two other patterns at four columns, and the intersection sets are disjoint. This simple example proves to be extremely challenging for conventional SVD based techniques as well as k-means clustering algorithms. The SVD-based techniques tend to identify aggregates of overlapping groups as dominant singular vectors. K-means clustering is particularly susceptible for such examples to specific initializations.

Gaussian matrices are generated as follows: a distribution maps the columns of the matrix to a Gaussian probability function of distribution $\mathcal{N}(\mu, \sigma)$, where $1 \leq \mu \leq n$ and σ are determined randomly for each distribution. Probability function $p(i)$ determines the probability of having a non-zero on the i^{th} entry of the pattern vector. A pattern is generated by superposing a number of distributions and scaling the probability function so that $\sum_{i=1}^n p(i) = E_{nz}$ where E_{nz} is the average number of non-zeros in a row, which is determined by $E_{nz} = \delta n$. δ is a pre-defined density parameter. The rows and columns of generated matrices are randomly ordered to hide the patterns in the matrix (please see first row of plots in Figure 2 for sample inputs).

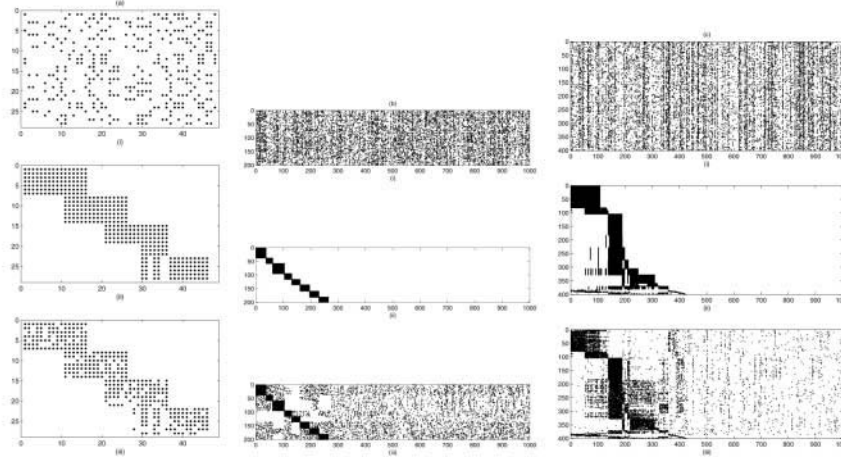


Fig. 2. Performance of Proximus on a (a) 28×48 uniform matrix with 4 patterns intersecting pairwise, (b) 200×1000 Gaussian matrix with 10 patterns each consisting of one distribution; (c) 400×1000 Gaussian matrix with 10 patterns each consisting of at most 2 distributions

4.2 Results

Effectiveness of Analysis As the error metric defined in the previous section depends on the nature of the pattern matrix, it does not provide useful information for evaluating the performance of the proposed method. Thus, we qualitatively examine the results obtained on sample test matrices. Figure 2(a) shows the performance of Proximus on a small uniform test matrix. The first matrix in the figure is the original generated and reordered matrix with 4 uniform patterns. The second matrix is the reordered approximation matrix which is estimated as XY^T where X and Y are 28×5 and 48×5 presence and pattern matrices containing the information of 5 patterns detected by Proximus. The 5th pattern is characterized by the intersection of a pair of patterns in the original matrix. The matrix is reordered in order to demonstrate the presence (and extent) of detected patterns in input data.

The performance of Proximus on a simple Gaussian matrix is shown on Figure 2(b). In this example, the original matrix contains 10 patterns containing one distribution each and Proximus was able to detect all these patterns as seen in the figure.

Figure 2(c) shows a harder instance of the problem. In this case, the 10 patterns in the matrix contain at most 2 of 7 Gaussian distributions. The patterns and the distributions they contain are listed in Table 4.3(a). The matrix is of dimension 400×1000 , and each group of 40 rows contain a pattern. As seen in the figure, Proximus was able to detect most of the patterns existing in the

matrix. Actually, 13 significant patterns were identified by Proximus, most dominant 8 of which are displayed in Table 4.3(b). Each row of Table 4.3(b) shows a pattern detected by Proximus. The first column contains the number of rows conforming to that pattern. In the next 10 columns, these rows are classified into original patterns in the matrix. Similarly, the last 7 columns show the classification of these rows due to distributions in the original data. For example, 31 rows contain the same detected pattern shown in the second row of the table, 29 of which contain pattern P7 and other two contain pattern P8 originally. Thus, distributions D4 and D7, both having 29 rows containing them, dominated this pattern. Similarly, we can conclude that the third row is dominated by pattern P1 (distribution D2), and the fourth and seventh rows are characterized by distribution D5. The interaction between the most dominant distribution D5 and the distributions D1, D3, D4 and D6 shows itself in the first row. Although this row is dominated by D5, several other distributions are classified in this pattern since these distributions share some pattern with D5 in the original matrix. These results clearly demonstrate the power of Proximus in identifying patterns even for very complicated datasets.

4.3 Runtime Scalability

Theoretically, each iteration of the algorithm for finding a rank-one approximation requires $O(nz(A))$ time since a constant number of matrix-vector multiplications dominates the runtime of an iteration. As the matrices created during the recursive process are more sparse than the original matrix, the total time required to estimate the rank-one approximation for all matrices at a level in the recursion tree is asymptotically less than the time required to estimate the rank-one approximation for the initial matrix. Thus, the total runtime of the algorithm is expected to be $O(nz(A))$ with a constant depending on the number of dominant patterns in the matrix which determines the height of the recursion tree. The results displayed in Figure 3 illustrate the scalability of the algorithm in terms of number of columns, rows and non-zeros in the matrix. These experiments are performed by:

1. varying the number of columns, where number of rows and the average number of non-zeros per column are set to constant values 1000 and 50 respectively.
2. varying the number of rows, where number of columns and the average number of non-zeros per row are set to constant values 1000 and 50 respectively.
3. varying the number non-zeros, where the average non-zero density in rows is set to constant value 50 and the number of rows and columns are kept equal.

All experiments are repeated with different randomly generated matrices 50 times for all values of the varying parameter. The reported values are the average run-times over these 50 experiments. In cases 1. and 2. above, the number of non-zeros grows linearly with the number of rows and columns, therefore, we expect

Table 1. (a) Description of patterns in the original data, (b) Classification of patterns detected by Proximus by original patterns and distributions

Pattern	Distributions	Number of rows	Patterns										Distributions						
			P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	D1	D2	D3	D4	D5	D6	D7
P1	D2	105			14		26	20	4	12	7	22	14		20	26	101	26	4
P2	D3, D6	31							29	2						29	2		29
P3	D1, D5	24	24											24					
P4	D2, D7	24								8	16						24		
P5	D5, D6	23	13		10									23					10
P6	D3, D5	23	2	21										2	21			21	
P7	D4, D7	23								15	7						22		
P8	D5	22											13	2		5	19		2
P9	D5	20			13	2		1											
P10	D4, D5																		

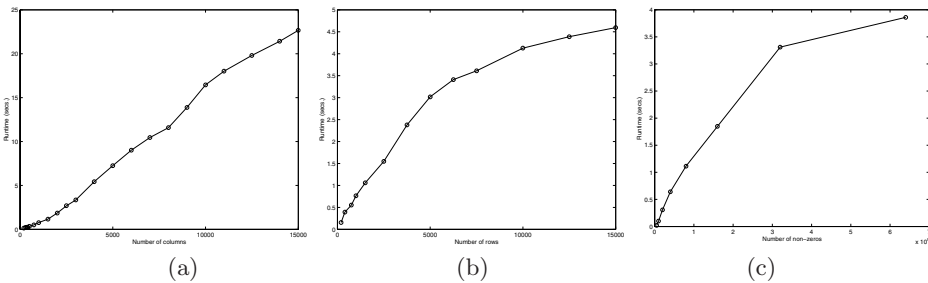


Fig. 3. Runtime of Proximus (secs.) with respect to (a) number of columns (b) number of rows (c) number of non-zeros in the matrix

to see an asymptotically linear runtime. As seen in Figure 3, the runtime of Proximus is asymptotically linear in number of columns. The runtime shows an asymptotically sublinear behavior with growing number of rows. This is because each row appears in at most one matrix at a level of the recursion tree. The behavior of runtime is similar when increasing the number of non-zeros.

5 Conclusions and Ongoing Work

In this paper, we have presented a powerful new technique for analysis of large high-dimensional discrete valued attribute sets. Using a range of algebraic techniques and data structures, this technique achieves excellent performance and scalability. The proposed analysis tool can be used in applications such as dominant and deviant pattern detection, collaborative filtering, clustering, bounded error compression, and classification. Efforts are currently under way to demonstrate its performance on real applications in information retrieval and in bioinformatics on gene expression data.

Acknowledgements

This work is supported in part by National Science Foundation grants EIA-9806741, ACI-9875899, ACI-9872101, EIA-9984317, and EIA-0103660. Computing equipment used for this work was supported by National Science Foundation and by the Intel Corp. The authors would like to thank Profs. Vipin Kumar at the University of Minnesota and Christoph Hoffmann at Purdue University for many useful suggestions.

References

1. M. W. Berry, S. T. Dumais, and G. W. O'Brien. Using Linear Algebra for Intelligent Information Retrieval. *SIAM Review*, Vol. 37(4):pages 573–595, 1995. 311, 312
2. P. Drienas, A. Frieze, R. Kannan, S. Vempala, and V. Vinay. Clustering in Large Graphs and Matrices. In *Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 291–299, 1999. 313
3. D. Gibson, J. Kleinberg, and P. Raghavan. Clustering Categorical Data: An Approach Based on Dynamical Systems. *VLDB Journal*, Vol. 8(3–4):pages 222–236, 2000. 313
4. R. M. Gray. Vector Quantization. *IEEE ASSP Magazine*, Vol. 1(2):pages 4–29, 1984. 313
5. S. Guha, R. Rastogi, and K. Shim. ROCK: A Robust Clustering Algorithm for Categorical Attributes. *Information Systems*, Vol. 25(5):pages 345–366, 2000. 313
6. G. Gupta and J. Ghosh. Value Balanced Agglomerative Connectivity Clustering. In *Proceedings of the SPIE conference on Data Mining and Knowledge Discovery III*, April 2001. 313
7. E. H. Han, G. Karypis, V. Kumar, and B. Mobasher. Hypergraph-Based Clustering in High-Dimensional Datasets: A Summary of Results. *Bulletin of the IEEE Technical Committee on Data Engineering*, Vol. 21(1):pages 15–22, March 1998. 313
8. T. Hofmann. Probabilistic Latent Semantic Indexing. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 50–57, 1999. 313
9. Z. Huang. A Fast Clustering Algorithm to Cluster Very Large Categorical Data Sets in Data Mining. In *Proceedings of the ACM SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery*, 1997. 313
10. T. G. Kolda and D. P. O'Leary. A Semidiscrete Matrix Decomposition for Latent Semantic Indexing in Information Retrieval. *ACM Transactions on Information Systems*, Vol. 16(4):pages 322–346, October 1998. 312
11. T. G. Kolda and D. P. O'Leary. Computation and Uses of the Semidiscrete Matrix Decomposition. *ACM Transactions on Mathematical Software*, Vol. 26(3):pages 416–437, September 2000. 312, 315
12. D. D. Lee and H. S. Seung. Learning the Parts of Objects by Non-Negative Matrix Factorization. *Nature*, Vol. 401:pages 788–791, 1999. 313
13. J. MacQueen. Some Methods for Classification and Analysis of Multivariate Observations. In *Proceedings of the Fifth Berkeley Symposium*, volume 1, pages 281–297, 1967. 313

14. S. McConnell and D. B. Skillicorn. Outlier Detection using Semi-Discrete Decomposition. Technical Report 2001-452, Dept. of Computing and Information Science, Queen's University, 2001. 312
15. D. P. O'Leary and S. Peleg. Digital Image Compression by Outer Product Expansion. *IEEE Transactions on Communications*, Vol. 31(3):pages 441–444, 1983. 315
16. M. Ozdal and C. Aykanat. Clustering Based on Data Patterns using Hypergraph Models. *Data Mining and Knowledge Discovery*, 2001. Submitted for publication. 313
17. S. Zyto, A. Grama, and W. Szpankowski. Semi-Discrete Matrix Transforms (SDD) for Image and Video Compression. Purdue University, 2002. Working manuscript. 312

Geography of Differences between Two Classes of Data

Jinyan Li and Limsoon Wong

Laboratories for Information Technology
21 Heng Mui Keng Terrace, Singapore 119613
{jinyan,limsoon}@lit.org.sg

Abstract. Easily comprehensible ways of capturing main differences between two classes of data are investigated in this paper. In addition to examining individual differences, we also consider their neighbourhood. The new concepts are applied to three gene expression datasets to discover diagnostic gene groups. Based on the idea of prediction by collective likelihoods (PCL), a new method is proposed to classify testing samples. Its performance is competitive to several state-of-the-art algorithms.

1 Introduction

An important problem in considering two classes of data is to discover significant differences between the two classes. This type of knowledge is useful in biomedicine. For example, in gene expression experiments [1,6], doctors and biologists wish to know genes or gene groups whose expression levels change sharply between normal cells and disease cells. Then, these genes or their protein products can be used as diagnostic indicators or drug targets of that specific disease.

Based on the concept of emerging patterns [3], we define a *difference* as a set of conditions that most data of a class satisfy but none of the other class satisfy. We investigate the geography—properties of neighbourhoods—of these differences. The differences include those corresponding to boundary rules for separating the two classes, those at the same level of significance in one class, and those at lower part of the boundaries. After examining these neighbourhoods, we can identify differences that are more interesting. We first discuss our ideas in a general sense. Then we apply the methods to three gene expression datasets [1,6] to discover interesting gene groups. We also use the discovered patterns to do classification and prediction.

Suppose we are given two sets of relational data where a fixed number of *features* (also called *attributes*) exist. Every feature has a range of numeric real values or a set of categorical values. A *condition* (also called *item*) is defined as a pair of a feature and its value. An example of a condition (an item) is “the expression of *gene_x* is less than 1000”. We denote this condition by *gene_x@(−∞, 1000)*, where the feature is *gene_x* and its value is $(-\infty, 1000)$. An *instance* (or a sample) is defined as a set of conditions (items) with a cardinality equal to the number of features in the relational data.

A *pattern* is a set of conditions. A pattern is said to *occur* in an instance if the instance contains it. For two classes of instances, a pattern can have a very high occurrence (equivalently, frequency) in one class, but can change to a low or even zero occurrence in the other class. Those patterns with a significant occurrence change are called emerging patterns (EPs) [3]. Here, our differences are those described by EPs.

This paper is organized as follows: Firstly, we present a formal description of the problems, including the definition of *boundary* EPs, *plateau spaces*, and *shadow patterns*, and present a related work. Then we describe convex spaces and prove that all plateau spaces satisfy convexity. This property is useful in concisely representing large pattern spaces. We also try to categorize boundary EPs using the frequency of their subsets. Then we present our main results, patterns discovered from biological data, and explain them in both biological and computational ways. To show the potential of our patterns in classification, we propose a new method that sums the collective power of individual patterns. Our accuracy is better than other methods. Then we briefly report our recent progress on a very big gene expression dataset which is about the subtype classification and relapse study of Acute Lymphoblastic Leukemia.

2 Problems and Related Work

Three types of patterns—*boundary* EPs, *plateau* EPs, and *shadow patterns*—are investigated in this work. Let us begin with a definition of emerging patterns.

Definition 1. *Given two classes of data, an emerging pattern is a pattern whose frequency in one class is non-zero but in the other class is zero.*

Usually, the class in which an EP has a non-zero frequency is called the EP’s *home* class or its own class. The other class in which the EP has the zero frequency is called the EP’s *counterpart* class.

2.1 Boundary EPs

Many EPs may have very low frequency (e.g. 1 or 2) in their home class. So boundary EPs are proposed to capture big differences between the two classes:

Definition 2. *A boundary EP is an EP whose proper subsets are not EPs.*

How do boundary EPs capture big differences? If a pattern contains less number of items (conditions), then the frequency (probability) that it occurs in a class becomes larger. Removing any one item from a boundary EP thus increases its home class frequency. However, by definition of boundary EPs, the frequency of any of its subsets in the counterpart class must be non-zero. Therefore, boundary EPs are maximally frequent in their home class. They separate EPs from non-EPs. They also distinguish EPs with high occurrence from EPs with low occurrence.

Efficient discovery of boundary EPs has been solved in our previous work [12]. Our new contribution in this work is the ranking of boundary EPs. The number of boundary EPs is sometimes large. The top-ranked patterns can help users understand applications better and easier. We also propose a new algorithm to make use of the frequency of the top-ranked patterns for classification.

2.2 Plateau EPs and Plateau Spaces

Next we discuss a new type of emerging patterns. If one more condition (item) is added to a boundary EP, generating a superset of the EP, the new EP may still have the same frequency as the boundary EP's. We call those EPs having this property plateau EPs:

Definition 3. *Given a boundary EP, all its supersets having the same frequency are called its plateau EPs.*

Note that boundary EPs themselves are trivially their plateau EPs. Next we define a new space, looking at all plateau EPs as a whole.

Definition 4. *All plateau EPs of all boundary EPs with the same frequency are called a plateau space (or simply, a P-space).*

So, all EPs in a P-space are at the same significance level in terms of their occurrence in both their home class and counterpart class. Suppose the home frequency is n , then the P-space is specially denoted P_n -space.

We will prove that all P-spaces have a nice property called *convexity*. This means a P-space can be succinctly represented by its most *general* and most *specific* elements.¹ We study how P-spaces contribute to the high accuracy of our classification system.

2.3 Shadow Patterns

All EPs defined above have the same *infinite* frequency growth-rate from their counterpart class to their home class. However, all proper subsets of a boundary EP have a *finite* frequency growth-rate as they occur in both the classes. It is interesting to see how these subsets change their frequency between the two classes by studying the growth rates. Next we define shadow patterns, which are special subsets of a boundary EP.

Definition 5. *All immediate subsets of a boundary EP are called shadow patterns.*

Shadow patterns can be used to measure the interestingness of boundary EPs. Given a boundary EP X , if the growth-rates of its shadow patterns approach $+\infty$, then the existence of this boundary EP is reasonable. This is because the

¹ Given a collection $\mathcal{C}$ of patterns and $A \in \mathcal{C}$, A is most general if there is no proper subset of A in $\mathcal{C}$. Similarly, A is most specific if there is no proper superset of A in $\mathcal{C}$.

possibility of X being a boundary EP is large. Otherwise if the growth-rates of the shadow patterns are on average around small numbers like 1 or 2, then the pattern X is *adversely interesting*. This is because the possibility of X being a boundary EP is small; the existence of this boundary EP is “unexpected”. This conflict may reveal some new insights into the correlation of the features.

2.4 Related Work on EPs

The general discussion of EP spaces has been thoroughly studied in our earlier work [12]. It has been proven that every EP space is a convex space. The efficient discovery of boundary EPs was a problem and it was solved by using border-based algorithms [3, 12]. Based on experience, the number of boundary EPs is usually large— from 100s to 1000s depending on datasets. So, the ranking and visualization of these patterns is an important issue. We propose some ideas here to sort and list boundary EPs.

The original idea of the concept of emerging patterns is proposed in [3]. General definition of EPs, its extension to spatial data and to time series data, and the mining of general EPs can be also found there [3]. This paper discusses two new types of patterns: plateau patterns and shadow patterns. They are closely related to boundary EPs. We study these three types of patterns together here.

The usefulness of EPs in classification has been previously investigated [4, 11]. We propose in this paper a new idea that only top-ranked boundary EPs are used in classification instead of using all boundary EPs. This new idea leads to a simple system without any loss of accuracy and can avoid the effect of possible noisy patterns.

3 The Convexity of P-spaces

Convexity is an important property of a certain type of large collections. It can be exploited to concisely represent those collections of large size. Next we give a definition of convex space. Then we prove that our P-spaces satisfy convexity.

Definition 6. A collection $\mathcal{C}$ of patterns is a convex space if, for any patterns X , Y , and Z , the conditions $X \subseteq Y \subseteq Z$ and $X, Z \in \mathcal{C}$ imply that $Y \in \mathcal{C}$.

If a collection is a convex space, it is said to hold *convexity*. More discussion about convexity can be found in [7].

Example 1. The patterns $\{a\}$, $\{a, b\}$, $\{a, c\}$, $\{a, d\}$, $\{a, b, c\}$, and $\{a, b, d\}$ form a convex space. The set $\mathcal{L}$ consisting of the most general elements in this space is $\{\{a\}\}$. The set $\mathcal{R}$ consisting of the most specific elements in this space is $\{\{a, b, c\}, \{a, b, d\}\}$. All the other elements can be considered to be “between” $\mathcal{L}$ and $\mathcal{R}$.

Theorem 1. *Given a set $\mathcal{D}_P$ of positive instances and a set $\mathcal{D}_N$ of negative instances, every P_n -space ($n \geq 1$) is a convex space.*

Proof. By definition, a P_n -space is the set of all plateau EPs of all boundary EPs with the same frequency of n in the same home class. Without loss of generality, suppose two patterns X and Z satisfy (i) $X \subseteq Z$; (ii) X and Z are plateau EPs having the occurrence of n in $\mathcal{D}_P$. Then, for any pattern Y satisfy $X \subseteq Y \subseteq Z$, it is a plateau EP with the same n occurrence in $\mathcal{D}_P$. This is because

1. X does not occur in $\mathcal{D}_N$. So, Y , a superset of X , does not occur in $\mathcal{D}_N$ either.
2. The pattern Z has n occurrences in $\mathcal{D}_P$. So, Y , a subset of Z , also has a non-zero frequency in $\mathcal{D}_P$.
3. The frequency of Y in $\mathcal{D}_P$ must be less than or equal to the frequency of X , but must be larger than or equal to the frequency of Z . As the frequency of both X and Z is n , the frequency of Y in $\mathcal{D}_P$ is also n .
4. X is a superset of a boundary EP, thus Y is a superset of some boundary EP as $X \subseteq Y$.

By the first two points, we can infer that Y is an EP of $\mathcal{D}_P$. From the third point, we know that Y 's occurrence in $\mathcal{D}_P$ is n . Therefore, with the forth point above, Y is a plateau EP. Then we have proven that every P_n -space is a convex space.

A plateau space can be bounded by two sets similar to the sets $\mathcal{L}$ and $\mathcal{R}$ as shown in example 1. The set $\mathcal{L}$ consists of the boundary EPs. These EPs are the most general elements of the P-space. Usually, features contained in the patterns in $\mathcal{R}$ are more numerous than the patterns in $\mathcal{L}$. This indicates that some feature groups can be expanded while keeping their significance.

The structure of an EP space can be understood in a way by decomposing the space into a series of P-spaces and a non P-space. This series of P-spaces can be sorted according to their frequency. Interestingly, one of them with the highest frequency is a version space [14,8] if the EPs have the full 100% frequency in their home class.

4 Our Discovered Patterns from Gene Expression Datasets

We next apply our methods to two public datasets. One contains gene expression levels of normal cells and cancer cells. The other contains gene expression levels of two main subtypes of a disease. We report our discovered patterns, including boundary EPs, P-spaces, and shadow patterns. We also explain these patterns in a biological sense.

Table 1. Two publicly accessible gene expression datasets

Dataset	Gene number	Training size	Classes
Leukemia	7129	27, 11	ALL, AML
Colon	2000	22, 40	Normal, Cancer

4.1 Data Description

The process of transcribing a gene’s DNA sequence into RNA is called *gene expression*. After translation, RNA becomes proteins consisting of amino-acid sequences. A gene’s expression level is the rough number of copies of that gene’s RNA produced in a cell.

Gene expression data, obtained by highly parallel experiments using technologies like oligonucleotide ‘chips’ [13], record expression levels of genes under specific experimental conditions. By conducting gene expression experiments, one hopes to find possible trends or regularities of every single gene under a series of conditions, or to identify genes whose expressions are good diagnostic indicators for a disease.

A leukemia dataset [6] and a colon tumor dataset [1] are used in this paper. The former contains a training set of 27 samples of acute lymphoblastic leukemia (ALL) and 11 samples of acute myeloblastic leukemia (AML), and a blind testing set of 20 ALL and 14 AML samples. (ALL and AML are two main subtypes of the leukemia disease.) The high-density oligonucleotide microarrays used 7129 probes of 6817 human gene. All these data are public available at <http://www.genome.wi.mit.edu/MPR>. The second dataset consists of 22 normal and 40 colon cancer tissues. The expression level of 2000 genes of these samples are recorded. The data is available at <http://microarray.princeton.edu/oncology/affydata/index.html>. We use Table 1 to summarize the data. A common characteristic of gene expression data is that the number of samples is not large and the number of features is high in comparison with commercial market data.

4.2 Gene Selection and Discretization

A major challenge in analysing gene expression data is the overwhelming number of features. How to extract informative genes and how to avoid noisy data effects are important issues. We use an entropy-based method [5,9] and the CFS (Correlation-based Feature Selection) algorithm [16] to perform feature selection and discretization.

The entropy-based discretization method ignores those features which contain a random distribution of values with different class labels. It finds those features which have big intervals containing almost the same class of points. The CFS method is a post-process of the discretization. Rather than scoring (and ranking) individual features, the method scores (and ranks) the worth of subsets of the discretized features [16].

Table 2. Four most discriminatory genes of the 7129 features. Each feature is partitioned into two intervals using the cut points in column 2. The item index is convenient for writing EPs

Features	Cut Point	Item Index
Zyxin	994	1, 2
FAH	1346	3, 4
CST3	1419.5	5, 6
Tropomyosin	83.5	7, 8

4.3 Patterns Derived from the Leukemia Data

The CFS method selects only one gene, Zyxin, from the total of 7129 features. The discretization method partitions this feature into two intervals using the cut point at 994. Then, we discovered two boundary EPs,

$$\{gene\_zyxin@(-\infty, 994)\} \text{ and } \{gene\_zyxin@[994, +\infty)\},$$

having a 100% occurrence in their home class.

Biologically, these two EPs say that if the expression of Zyxin in a cell is less than 994, then this cell is an ALL sample. Otherwise this cell is an AML sample. This rule regulates all 38 training samples without any exception. If this rule is applied to the 34 blind testing samples, we obtained only three misclassifications. This result is better than the accuracy of the system reported in [6].

Biological and technical noise sometimes happen in many stages such as in the production of DNA arrays, the preparation of samples, the extraction of expression levels, and may be from the impurity or mis-classification of tissues. To overcome these possible machine and human minor errors, we suggest to use more than one gene to strengthen our system as shown later.

We found four genes whose entropy values are significantly less than all the other 7127 features when partitioned by the discretization method. We used these four genes for our pattern discovery whose name, cut points, and item indexes are listed in Table 2.

We discovered a total of 6 boundary EPs, 3 each in the ALL and AML classes. Table 3 presents the boundary EPs together with their occurrence and the percentage of the occurrence in the whole class. The reference numbers contained in the patterns can be interpreted using the interval index in Table 2.

Biologically, the EP $\{5, 7\}$ as an example says that if the expression of CST3 is less than 1419.5 and the expression of Tropomyosin is less than 83.5 then this sample is ALL with 100% accuracy. So, all those genes involved in our boundary EPs are very good diagnostic indicators for classifying ALL and AML.

We discovered a P-space based on two boundary EPs of $\{5, 7\}$ and $\{1\}$. This P_{27} -space consists of five plateau EPs: $\{1\}$, $\{1, 7\}$, $\{1, 5\}$, $\{5, 7\}$, and $\{1, 5, 7\}$. The most specific plateau EP is $\{1, 5, 7\}$ and it still has a full occurrence of 27 in the ALL class.

Table 3. Three boundary EPs in the ALL class and three boundary EPs in the AML class

Boundary EPs	Occurrence in ALL (%)	Occurrence in AML (%)
{5, 7}	27 (100%)	0
{1}	27 (100%)	0
{3}	26 (96.3%)	0
{2}	0	11 (100%)
{8}	0	10 (90.9%)
{6}	0	10 (90.9%)

Table 4. Here only top 5 ranked boundary EPs in the normal class and in the cancerous class are listed. The meaning of the reference numbers contained in the patterns are not presented due to page limitation

Boundary EPs	Occurrence Normal (%)	Occurrence Cancer (%)
{2, 6, 7, 11, 21, 23, 31}	18 (81.8%)	0
{2, 6, 7, 21, 23, 25, 31}	18 (81.8%)	0
{2, 6, 7, 9, 15, 21, 31}	18 (81.8%)	0
{2, 6, 7, 9, 15, 23, 31}	18 (81.8%)	0
{2, 6, 7, 9, 21, 23, 31}	18 (81.8%)	0
{14, 34, 38}	0	30 (75.0%)
{18, 34, 38}	0	26 (65.0%)
{18, 32, 38, 40}	0	25 (62.5%)
{18, 32, 44}	0	25 (62.5%)
{20, 34}	0	25 (62.5%)

4.4 Patterns Derived from the Colon Tumor Data

This dataset is a bit more complex than the ALL/AML data. The CFS method selected 23 features from the 2000 as most important. All of the 23 features were partitioned into two intervals.

We discovered 371 boundary EPs in the normal cells class, and 131 boundary EPs in the cancer cells class. The total 502 patterns were ranked according to the these criteria:

1. Given two EPs X_i and X_j , if the frequency of X_i is larger than X_j , then X_i is prior to X_j in the list.
2. When the frequency of X_i is equal to X_j , if the cardinality of X_i is larger than X_j , then X_i is prior to X_j in the list.
3. If their frequency and cardinality are both identical, then X_i is prior to X_j when X_i is first produced.

Some top ranked boundary EPs are reported in Table 4.

Unlike the ALL/AML data, in the colon tumor dataset there does not exist single genes acting as arbitrator to separate normal and cancer cells clearly. Instead, gene groups are contrasting the two classes. Note that these boundary EPs, especially those having many conditions, are not obvious but novel to biol-

Table 5. Most general and most specific elements in a P_{18} -space in the normal class of the colon data

Most general and specific EPs Occurrence in Normal	
$\{2, 6, 7, 11, 21, 23, 31\}$	18
$\{2, 6, 7, 21, 23, 25, 31\}$	18
$\{2, 6, 7, 9, 15, 21, 31\}$	18
$\{2, 6, 7, 9, 15, 23, 31\}$	18
$\{2, 6, 7, 9, 21, 23, 31\}$	18
$\{2, 6, 9, 21, 23, 25, 31\}$	18
$\{2, 6, 7, 11, 15, 31\}$	18
$\{2, 6, 11, 15, 25, 31\}$	18
$\{2, 6, 15, 23, 25, 31\}$	18
$\{2, 6, 15, 21, 25, 31\}$	18
$\{2, 6, 7, 9, 11, 15, 21, 23, 25, 31\}$	18

Table 6. A boundary EPs and its three shadow patterns

Patterns	Occurrence in Normal	Occurrence in Cancer
$\{14, 34, 38\}$	0	30
$\{14, 34\}$	1	30
$\{14, 38\}$	7	38
$\{34, 38\}$	5	31

ogists and medical doctors. They may reveal some new protein interactions and may be used to find new pathways.

There are a total of ten boundary EPs having the same highest occurrence of 18 in the normal cells class. Based on these boundary EPs, we found a P_{18} -space in which the only most specific element is $Z = \{2, 6, 7, 9, 11, 15, 21, 23, 25, 31\}$. By convexity, any subsets of Z but superset of anyone of the ten boundary EPs have the occurrence of 18 in the normal class. Observe that there are approximately one hundred EPs in this P-space. While by convexity, we can concisely represent this space using only 11 EPs which are shown in Table 5.

From this P-space, it can be seen that significant gene groups (boundary EPs) can be expanded by adding some other genes without loss of significance, namely still keeping high occurrence in one class but absence in the other class. This may be useful in identifying a maximum length of a pathway.

We found a P_{30} -space in the cancerous class. The only most general EP in this space is $\{14, 34, 38\}$ and the only most specific EP is $\{14, 30, 34, 36, 38, 40, 41, 44, 45\}$. So a boundary EP can be extended by six more genes without a reduction in occurrence.

It is easy to find shadow patterns. Below, we report a boundary EP and its shadow patterns (see Table 6). These shadow patterns can also be used to illustrate the point that proper subsets of a boundary EP must occur in two classes at non-zero frequency.

5 Usefulness of EPs in Classification

In the previous section, we have found many simple EPs and rules which can well regulate gene expression data. Next we propose a new method, called PCL, to test the reliability and classification potential of the patterns by applying them to the 34 blind testing sample of the leukemia dataset [6] and by conducting a Leave-One-Out cross-validation (LOOCV) on the colon dataset.

5.1 Prediction by Collective Likelihood (PCL)

From the leukemia training data, we first discovered two boundary EPs which form a simple rule. So, there was no ambiguity in using the rule. However, a large number of EPs were found in the colon dataset. A testing sample may contain not only EPs from its own class, but it may also contain EPs from its counterpart class. This makes the prediction a bit more complicated. Naturally, a testing sample should contain many top-ranked EPs from its own class and contain a few low-ranked, preferably no, EPs from its opposite class. However, according to our observations, a testing sample can sometimes, though rarely, contain 1 to 20 top-ranked EPs from its counterpart class. To make reliable predictions, it is reasonable to use multiple highly frequent EPs of the home class to avoid the confusing signals from counterpart EPs.

Our method is described as follows: Given two training datasets $\mathcal{D}_P$ and $\mathcal{D}_N$ and a testing sample T , the first phase of our prediction method is to discover boundary EPs from $\mathcal{D}_P$ and $\mathcal{D}_N$. Denote the ranked EPs of $\mathcal{D}_P$ as,

$$TopEP\_P\_1, TopEP\_P\_2, \dots, TopEP\_P\_i,$$

in descending order of frequency. Similarly, denote the ranked boundary EPs of $\mathcal{D}_N$ as

$$TopEP\_N\_1, TopEP\_N\_2, \dots, TopEP\_N\_j$$

also in descending order of frequency. Suppose T contains the following EPs of $\mathcal{D}_P$:

$$TopEP\_P\_i_1, TopEP\_P\_i_2, \dots, TopEP\_P\_i_x,$$

where $i_1 < i_2 < \dots < i_x \leq i$, and the following EPs of $\mathcal{D}_N$:

$$TopEP\_N\_j_1, TopEP\_N\_j_2, \dots, TopEP\_N\_j_y,$$

where $j_1 < j_2 < \dots < j_y \leq j$.

The next step is to calculate two scores for predicting the class label of T . Suppose we use k ($k \ll i$ and $k \ll j$) top-ranked EPs of $\mathcal{D}_P$ and $\mathcal{D}_N$. Then we define the score of T in the $\mathcal{D}_P$ class as

$$score(T)_{\mathcal{D}_P} = \sum_{m=1}^k \frac{frequency(TopEP\_P\_i_m)}{frequency(TopEP\_P\_m)},$$

Table 7. By LOOCV on the colon dataset, our PCL’s error rate comparison with other methods

Methods	Error Rates
C4.5	20
NB	13
k -NN	28
SVM	24
Our PCL	13, 12, 10, 10, 10, 10
$(k = 5, 6, 7, 8, 9, 10)$	

and similarly the score in the $\mathcal{D}_N$ class as

$$score(T)_{\mathcal{D}_N} = \sum_{m=1}^k \frac{frequency(TopEP\_N\_j_m)}{frequency(TopEP\_N\_m)}.$$

If $score(T)_{\mathcal{D}_P} > score(T)_{\mathcal{D}_N}$, then T is predicted as the class of $\mathcal{D}_P$. Otherwise predicted as the class of $\mathcal{D}_N$. We use the size of $\mathcal{D}_P$ and $\mathcal{D}_N$ to break tie.

The spirit of our proposal is to measure how far the top k EPs contained in T are away from the top k EPs of a class. Assume $k = 1$, then $score(T)_{\mathcal{D}_P}$ indicates whether the number one EP contained in T is far from the most frequent EP of $\mathcal{D}_P$. If the score is the maximum value 1, then the “distance” is very close, namely the most common property of $\mathcal{D}_P$ is also present in this testing sample. With smaller scores, the distance becomes further. Thus the likelihood of T belonging to the class of $\mathcal{D}_P$ becomes weaker. Using more than one top-ranked EPs, we utilize a “collective” likelihood for more reliable predictions. We name this method PCL (prediction by collective likelihood).

5.2 Classification Results

Recall that we also have selected four genes in the leukemia data as the most important. Using PCL, we obtained a testing error rate of two mis-classifications. This result is one error less than the result obtained by using the sole Zyxin gene.

For the colon dataset, using our PCL, we can get a better LOOCV error rate than other classification methods such as C4.5 [15], Naive Bayes (NB) [10], k -NN, and support vector machine (SVM) [2]. We used the default settings of the Weka package [16] and exactly the gene selection preprocessing steps as ours to get the results. The result is summarized in Table 7.

5.3 Making Use of P-spaces for Classification: A Variant of PCL

Can the most specific elements of P-spaces be useful in classification? In PCL, we tried to replace the ranked boundary EPs with the most specific elements of all P-spaces in the colon dataset. The remaining process of PCL are not changed. By

LOOCV, we obtained an error rate of only six mis-classifications. This reduction is significant.

The reason for this good result is that the neighbourhood of the most specific elements of a P-space are all EPs in most cases, but there are many patterns in the neighbourhood of boundary EPs that are *not* EPs. Secondly, the conditions contained in the most specific elements of a P-space are usually much more than the boundary EPs. So, with more number of conditions, the chance for a testing sample to contain opposite EPs becomes smaller. Hence, the probability of being correctly classified becomes higher.

6 Recent Progress

In a collaboration with St. Jude Children's Research Hospital, our algorithm has been applied to a big gene expression dataset [17]. This dataset consists of the expression profile of 327 patients who suffered from Acute Lymphoblastic Leukemia (ALL). Each instance is represented by 12,558 features. The purpose is to establish a classification model to predict whether a new patient suffers from one of the six main subtypes of ALL. By our PCL, we achieved a testing error rate that is 71% better than C4.5, 50% better than Naive Bayes, 43% better than k -NN, and 33% better than SVM.

More than mere a prediction, importantly, our algorithm provides simple rules and patterns. These knowledge can greatly help medical doctors and biologists deeply understand why an instance is predicted as positive or negative.

7 Conclusion

We studied how to describe main differences between two classes of data using emerging patterns. We proposed methods to rank boundary EPs. Using boundary EPs, we defined two new types of patterns, plateau EPs and shadow patterns, and proved that all P-spaces satisfied convexity. Based on the idea of prediction by collective likelihood, we proposed a new classification method called PCL.

All these ideas and methods have been applied to three gene expression data. The discovered patterns are interesting, and may be useful in identifying new pathways and interactions between proteins. The PCL methods performed better than other classification models on the datasets used in this paper.

In future, we plan to define central points of a P-space and use the central patterns for classification. Also, we like to study shadow patterns and their relation with boundary EPs more deeply than in this paper.

Acknowledgments

We thank Huiqing Liu for providing the classification results of C4.5, NB, k -NN, and SVM. We also thank the reviewers for their useful comments.

References

1. Alon, U. and et al. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proceedings of National Academy of Sciences of the United States of American*, 96:6745–675, 1999. 325, 330
2. Burges, C. J. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2:121–167, 1998. 335
3. Guozhu Dong and Jinyan Li. Efficient mining of emerging patterns: Discovering trends and differences. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 43–52, San Diego, CA, 1999. ACM Press. 325, 326, 328
4. Guozhu Dong, Xiuzhen Zhang, Limsoon Wong, and Jinyan Li. CAEP: Classification by aggregating emerging patterns. In *Proceedings of the Second International Conference on Discovery Science, Tokyo, Japan*, pages 30–42. Springer-Verlag, December 1999. 328
5. Fayyad, U. M. and Irani, K. B. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, pages 1022–1029. Morgan Kaufmann, 1993. 330
6. Golub, T. R. and et al. Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science*, 286:531–537, October 1999. 325, 330, 331, 334
7. Carl A. Gunter, Teow-Hin Ngair, and Devika Subramanian. The common order-theoretic structure of version spaces and ATMS's. *Artificial Intelligence*, 95:357–407, 1997. 328
8. Hirsh, H. Generalizing version spaces. *Machine Learning*, 17:5–46, 1994. 329
9. Kohavi, R. and et al. MLC++: A machine learning library in C++. In *Tools with artificial intelligence*, pages 740 – 743, 1994. 330
10. Langley, P., Iba, W. and Thompson, K. An analysis of Bayesian classifier. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 223 – 228. AAAI Press, 1992. 335
11. Jinyan Li, Guozhu Dong, and Kotagiri Ramamohanarao. Making use of the most expressive jumping emerging patterns for classification. *Knowledge and Information Systems: An International Journal*, 3:131–145, 2001. 328
12. Jinyan Li, Kotagiri Ramamohanarao, and Guozhu Dong. The space of jumping emerging patterns and its incremental maintenance algorithms. In *Proceedings of the Seventeenth International Conference on Machine Learning, Stanford, CA, USA*, pages 551–558, San Francisco, June 2000. Morgan Kaufmann. 327, 328
13. Lockhart, T. J. and et al. Expression monitoring by hybridization to high-density oligonucleotide arrays. *Nature Biotechnology*, 14:1675–1680, 1996. 330
14. Mitchell, T. M. Generalization as search. *Artificial Intelligence*, 18:203–226, 1982. 329
15. Quinlan, J. R. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993. 335
16. Witten, H. and Frank, E. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementation*. Morgan Kaufmann, San Mateo, CA, 2000. 330, 335
17. Eng-Juh Yeoh and et. al. Classification, subtype discovery, and prediction of outcome in pediatric acute lymphoblastic leukemia by gene expression profiling. *Cancer Cell*, 1:133–143, 2002. 336

Rule Induction for Classification of Gene Expression Array Data

Per Liden¹, Lars Asker^{1,2}, and Henrik Boström^{1,2}

¹ Virtual Genetics Laboratory AB
Fogdevreten 2A, SE-171 77 Stockholm, Sweden
{per.liden,lars.asker,henrik.bostrom}@vglab.com
<http://www.vglab.com>

² Department of Computer and Systems Sciences
Stockholm University and Royal Institute of Technology
Forum 100, SE-164 40 Kista, Sweden
{asker,henke}@dsv.su.se

Abstract. Gene expression array technology has rapidly become a standard tool for biologists. Its use within areas such as diagnostics, toxicology, and genetics, calls for good methods for finding patterns and prediction models from the generated data. Rule induction is one promising candidate method due to several attractive properties such as high level of expressiveness and interpretability. In this work we investigate the use of rule induction methods for mining gene expression patterns from various cancer types. Three different rule induction methods are evaluated on two public tumor tissue data sets. The methods are shown to obtain as good prediction accuracy as the best current methods, at the same time allowing for straightforward interpretation of the prediction models. These models typically consist of small sets of simple rules, which associate a few genes and expression levels with specific types of cancer. We also show that information gain is a useful measure for ranked feature selection in this domain.

1 Introduction

Gene expression array technology has become a standard tool for studying patterns and dynamics of the genetic mechanisms of living cells. Many recent studies have highlighted its usefulness for studying cancer [1], [2], [3]. Gene expression profiling does not only provide an attractive alternative to current standard techniques such as histology, genotyping, and immunostaining for tumor classification, but also valuable insights into the molecular characteristics of specific cancer types. In clinical settings, diagnosis is of great importance for the treatment of cancer patients since the responsiveness for various drugs and prognostic outcome can vary between subtypes of cancers. Correct tumor classification will ideally optimize treatment, save time and resources, and avoid unnecessary clinical side effects.

To date, a number of methods have been applied to the problem of learning computers to classify cancer types based on gene expression measurements from microarrays. Alon et al used clustering as a means for classification in their original analysis of the colon cancer data set [2]. Subsequently, methods such as Support Vector Machines (SVMs) [4], [5], Naïve Bayesian Classification [6], Artificial Neural Networks (ANNs) [3], and decision trees [7] have been employed to address this task. Some of these studies indicate that besides creating accurate prediction models, an important goal is to find valuable information about the system components that are being used as input to these models.

Previous studies have shown that classification accuracy can be improved by reducing the number of features used as input to the machine learning method [3], [1]. The reason for this is most likely that the high level of correlation between the expression levels of many genes in the cell makes much of the information from one microarray redundant. The relevance of good feature ranking methods in this domain has also been discussed by Guyon and colleagues [8].

Rule induction methods have been studied for more than two decades within the field of machine learning. They include various techniques such as divide-and-conquer (recursive partitioning), that generates hierarchically organized rules (decision trees) [9], and separate-and-conquer (covering) that generates overlapping rules. These may either be treated as ordered (decision lists) [10] or unordered rule sets [11]. Common for all these methods is that they are very attractive with regard to the analysis of input feature importance. Since the rule induction process in itself takes redundancy of input parameters into account, and that the process will seek to use the most significant features first, superfluous features are commonly left outside the prediction model. In this study we investigate three different rule induction methods for classifying cancer types based on gene expression measurements from microarrays, together with a simple method for feature selection based on information gain ranking.

Two public datasets are used in this study. The first data set is the colon cancer study published by Alon and co-workers [2]. Here the task is to separate between tumor tissue and normal colon tissue (a two class problem). This data set has been extensively studied by others [2], [4], [5], [6]. The prediction task for the second data set [3] is to discriminate between four types of small round blue cell tumors (SRBCTs): neuroblastoma (NB), rhabdomyosarcoma (RMS), Burkitt's lymphoma (BL), and the Ewing family of tumors (EWS).

2 Learning Algorithms

The rule induction system used in this study, Virtual Predict 1.0 [12], extends several rule induction algorithms developed in Spectre 3.0 [13]. The three methods that are used in the experiments are briefly described below. All three methods use a technique for discretizing numerical features during the induction process based on finding split points that separate examples belonging to different classes [14]. The technique is further optimized with respect to efficiency by using a sampling scheme that randomly selects 10% of the possible split points for each feature. All methods use the m-estimate [15], with m set to 2, for calculating class probabilities.

2.1 Divide-and-Conquer Using the Minimum Description Length Principle (DAC-MDL)

Divide-and-conquer (DAC), also known as recursive partitioning, is a technique that generates hierarchically organized rule sets (decision trees). In this work, DAC is combined with the information gain criterion [9] for selecting branching features. Furthermore, the minimum description length (MDL) criterion [16], modified to handle numerical attributes effectively, is used to avoid over-fitting. This method, referred to as DAC MDL, is preferred instead of splitting the training data into one grow and one prune set. Splitting into grow and prune set for this data is more likely to result in highly variable rule sets due to the limited size of the data sets.

2.2 Boosting Decision Trees (Boosting 50)

Boosting is an ensemble learning method that uses a weight distribution over the training examples and iteratively re-adjusts the distribution after having generated each component classifier in the ensemble. This is done in a way so that the learning algorithm focuses on those examples that are classified incorrectly by the current ensemble. New examples are classified according to a weighted vote of the classifiers in the ensemble [17]. The base learning method used in this study is divide-and-conquer using information gain together with a randomly selected prune set corresponding to 33% of the total weight. The number of trees in the ensemble is set to 50. Thus, the method generates an ensemble consisting of 50 individual base classifiers.

2.3 Separate-and-Conquer for Unordered Rule Sets (Unordered SAC)

Finally, a method that employs a totally different search strategy is compared to the previous methods, namely separate-and-conquer (SAC), also known as covering. SAC iteratively finds one rule that covers a subset of the data instead of recursively partitioning the entire data set, cf., [18]. The examples covered by this rule are then subtracted from the entire data set. This strategy is combined with incremental reduced error pruning [19], where each clause immediately after its generation is pruned back to the best ancestor. The criterion for choosing the best ancestor is to select the most compressive rule using an MDL coding scheme similar to the one in [16] but adapted to the single-rule case. The method generates an unordered set of rules, in contrast to generating a decision list [10]. This means that rules are generated independently for each class, and any conflicts due to overlapping rules are resolved during classification by using the naïve Bayes inference rule (i.e., calculating class probabilities while assuming independent rule coverage).

2.4 Feature Selection Using Information Gain

Since the number of features in the two data sets in the current study is more than 25 times the number of data points, some dimensionality reduction scheme may prove useful for obtaining accurate models, in particular for the methods that generate single models. Although the use of the minimum description length criterion has shown to be

quite effective for generating models that are tolerant against noise and random correlations, a large number of irrelevant and redundant variables may cause the rules to be over-pruned due to the additional cost of investigating these superfluous variables. Commonly used dimensionality reduction methods include principal component analysis, multi-dimensional scaling, and feature selection. Since the two former classes of methods do not necessarily lead to dimensions that are suited for discriminating examples belonging to different classes, a feature selection method based on the discriminative power (as measured by information gain) is preferred. This method has the additional benefit of allowing for direct interpretation of the generated rules (i.e., there is no need for transforming the rules back to the original feature space). The following formula, which is a Laplace corrected version of the formula in [9], is used to measure the information content for a numeric feature f and threshold value t :

$$I_{f,t} = \sum_{i=1}^n -l_i \log_2 \frac{l_i + 1}{l + n} + \sum_{i=1}^n -r_i \log_2 \frac{r_i + 1}{r + n}$$

where n is the number of classes, l_i is the number of examples of class i in the first subset (i.e., examples with a value on f that is less than or equal to t), l is the total number of examples in the first subset, r_i denotes the number of examples of class i in the second subset (i.e., examples with a value on f that is greater than t), and r is the total number of examples in the second subset. It should be noted that the above formula is restricted to evaluating binary splits (i.e., two elements in the partition), which is sufficient when dealing with numeric features that are divided into two intervals. For each numeric feature, all split points obtained from the examples were evaluated, and the k most informative features (i.e., those resulting in the subsets with least information content) were kept, for some given k .

Other feature selection methods could be used as well, but the above was chosen because of its simplicity and expected suitability.

3 Experimental Evaluation

3.1 Colon Cancer Data Set

For a set of 62 samples, 40 tumor samples and 22 normal colon tissue samples, the gene expression levels of 6500 genes were measured using Affymetrix oligonucleotide arrays [2]. Of these genes, the 2000 with the highest minimal intensity were selected by the authors for further analysis. The raw data was normalized for global variance between arrays by dividing the intensities of all genes by the average intensity of the array and multiplying by 50.

Feature selection. This is a two-class dataset, for which feature selection was done in one iteration. In this case, the 128 most highly ranked genes according to the information gain measure were selected for further analysis.

Classification results. Leave-one-out cross-validation was performed (Figure 1a) with the 2, 4, 8, 16, 32, 64, and 128 most highly ranked features. A few points can be made

of the results of this analysis. The ensemble method Boosting 50, gave the best prediction accuracy using all 128 features, resulting in 7 misclassified examples. This accuracy does not significantly differ from other results reported for this data set. SVMs gave 6 errors [5], clustering gave 7 errors [2], [4], and Naïve Bayes classification gave 9 errors [6]. It is interesting to note that the boosting method works significantly better when applied to decision trees than decision stumps (i.e., one-level decision trees); 89% accuracy in our case vs. 73% for stumps, as evaluated by Ben-Dor and co-workers [4]. Zhang and co-workers report classification accuracy above 90% using a decision tree induction method similar to ours [7]. However, their analysis can be discussed from a methodological point of view, since the tree structure was induced using the entire data set, and the split-point values were the only parameters that were changed during the five-fold cross-validation for which this result was reported. This method thus takes advantage from a significant amount of information from the data it is going to be evaluated on, which is likely to result in an over-optimistic estimate.

Interestingly, the largest number of features resulted in best prediction accuracy for the ensemble method (Boosting 50). Figure 1a highlights a trend towards better classification with fewer attributes for the simple methods, and the opposite trend for the ensemble method, which is known to be more robust with respect to handling variance due to small sample sizes in relation to the number of features.

3.2 Small Round Blue Cell Tumor (SRBCT) Data Set

The expression levels for 6567 genes in 88 samples of both tissue biopsies and cell lines were measured using cDNA microarrays and 2308 of those genes were selected by a filtering step and divided into one training set (63 samples) and one test set (25 samples) [3]. Class labels were assigned by histological analysis. We have used the same division into test and training as in the original work. In the entire dataset, 29 examples were from the Ewing family of tumors (EWS) (of which 6 were test examples), 11 were Burkitt's lymphoma (BL) (3 test examples), 18 were neuroblastoma (NB) (6 test examples), and 25 were rhabdomyosarcoma (RMS) (5 test examples). The test set also included five non-tumor samples.

Feature selection. In order to select the best candidate features (genes) for this dataset, the information gain for each feature was calculated with respect to its usefulness for separating each of the four classes from the other three. The 32 top ranking features for each class were then selected, resulting in 125 unique genes out of a total of 128 selected (three genes occurred twice).

Classification results. The best classifier generated from the training set, Boosting 50, perfectly separates the 20 cancer samples in the test set. This separation is obtained using only the four attributes corresponding to the top ranked feature for each class. The same result is obtained for twelve and all selected features as well. Using the 96 features selected by Khan and co-workers [3], 100 % accuracy is obtained as well. One difference between our results and the ANN approach of Khan et al is the relative simplicity of the model generated here. The rule based prediction models that produce 100 % accuracy on test examples are typically based on about 200 rules, regardless of the number of features used. This means that every decision tree in the ensemble is on

average composed of four rules, and that the entire classifier can be manually inspected (although with some difficulty). This can be compared to the 3750 ANN models created by Khan and colleagues. The other two methods performed slightly worse. At their best, Unordered SAC misclassified two test examples (for 32 features), while DAC MDL misclassified three test examples (also for 32 features). On the other hand, they generated significantly smaller models, consisting of five rules each.

We also performed leave-one-out cross validation of the entire dataset (both training and test examples) using the 4, 8, 16, 32, 64, and 128 most highly ranked features. The top $n/4$ ranked features for each class were selected in every round, where n is the total number of features selected. Error free classification was obtained for Boosting 50 when all the 128 features were selected, while one example was misclassified for 16, 32, and 64 features resulting in 99% accuracy (Figure 1b).

The trend of obtaining better classification with fewer attributes for the simple methods, and the opposite trend for the ensemble method that we noticed in the other experiment, can be observed also here, although this data set has four classes instead, where each class has its own ranked set of features.

3.3 Inspecting the Rules

In the previous section it was shown that the employed rule induction methods are useful for constructing accurate prediction models. In addition to this, rules can also give valuable insights into the studied systems. As an illustration, seven easily interpretable rules are found when applying the unordered SAC method using the 16 highest-ranking features on the entire SRBCT data set (Table 1).

Table 1: Rules discovered by unordered SAC for SRBCT data set

Class	Rule	Coverage of examples			
		EWS	BL	NB	RMS
EWS	FVT1 > 1.35535	27	0	0	1
EWS	Caveolin 1 > 1.59365	26	0	0	1
BL	WASP > 0.61645	0	11	0	0
NB	AF1Q > 2.1795	0	0	17	0
NB	CSDA ≤ 0.69175	0	0	13	0
RMS	SGCA > 0.4218	0	1	0	24
RMS	IGF2 > 13.5508	0	0	0	4

The rules discovered for EWS involve two genes: Caveolin 1 and follicular lymphoma variant translocation 1 (FVT1). Caveolin 1 encodes a protein that is known to play an important role in signal transduction and lipid transport. It has been associated with prostate cancer [20] and adenocarcinoma of the colon [21]. FVT1 has been proposed to be associated with follicular lymphoma by its close localization with Bcl-2 [22]. The single rule for Burkitt's lymphoma (BL) shows how this cancer type can be singled out based on a high expression level for the gene encoding the Wiskott-Aldrich syndrome protein (WASP) only. Likewise, neuroblastoma (NB) is separated from all

the other tumor types by two independent rules involving the expression levels of the genes for AF1Q and cold shock domain protein A (CSDA). Specific expression of a fusion between the AF1Q gene and the mixed lineage leukemia (MLL) gene has been associated with leukemia [23], and this finding suggests an involvement in NB, possibly indicating that the fusion is present in NB as well. CSDA is a transcriptional regulator involved in stress response, and is believed to act as a repressor of human granulocyte-macrophage colony stimulating factor (GM-CSF) transcription [24]. Its down regulation may indicate an involvement in tumorigenesis in NB. Finally, RMS is separated from the other tumor types by the specific expression of sarcoglycan alpha (SGCA), a muscle specific protein associated with muscular dystrophy [25]. High expression of this gene is probably more indicative of the tissue origin of this tumor type than related to the molecular background of RMS. The second rule for RMS involves insulin-like growth factor II (IGF2), which is an already known oncogene associated with this cancer type [26]. Figure 2 shows a graphic representation of the coverage of all the rules.

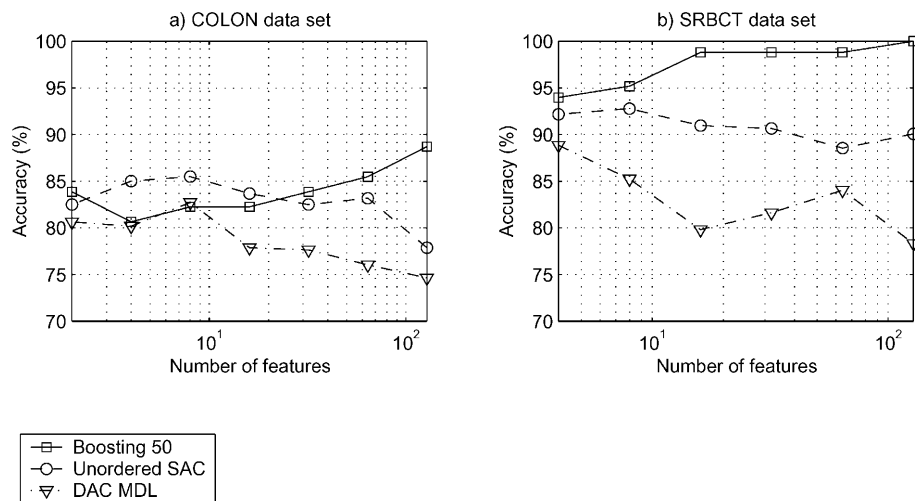


Fig. 1. Results from leave-one-out cross validation both data sets using DAC MDL, Unordered SAC, and Boosting 50. a) Results from the COLON data set using 2, 4, 8, 16, 32, 64, and 128 features. b) Results from the SRBCT data set using 4, 8, 16, 32, 64, and 128 features

4 Concluding Remarks

We have shown that rule induction methods are strong candidates for microarray analysis. One attractive property of this class of methods is that they do not only generate accurate prediction models, but also allow for straightforward interpretation of the reasons for the particular classification they make. Rule induction represents a whole class of methods, of which decision trees is perhaps the best known, but not necessarily the best-suited method for this particular type of task, as demonstrated in this study. Common for this class of methods is that they allow for a trade off between increased accuracy versus low complexity (i.e. high interpretability) of generated

models. We have evaluated three of these methods, DAC-MDL, SAC, and Boosting for two different tumor tissue classification tasks. The classification accuracy was shown to be on level with the best current methods while exhibiting a much higher level of interpretability. Moreover, as opposed to many other methods employed for microarray data classification, rule induction methods can be applied in a straightforward manner to multi-class problems, such as the SRBCT data set.

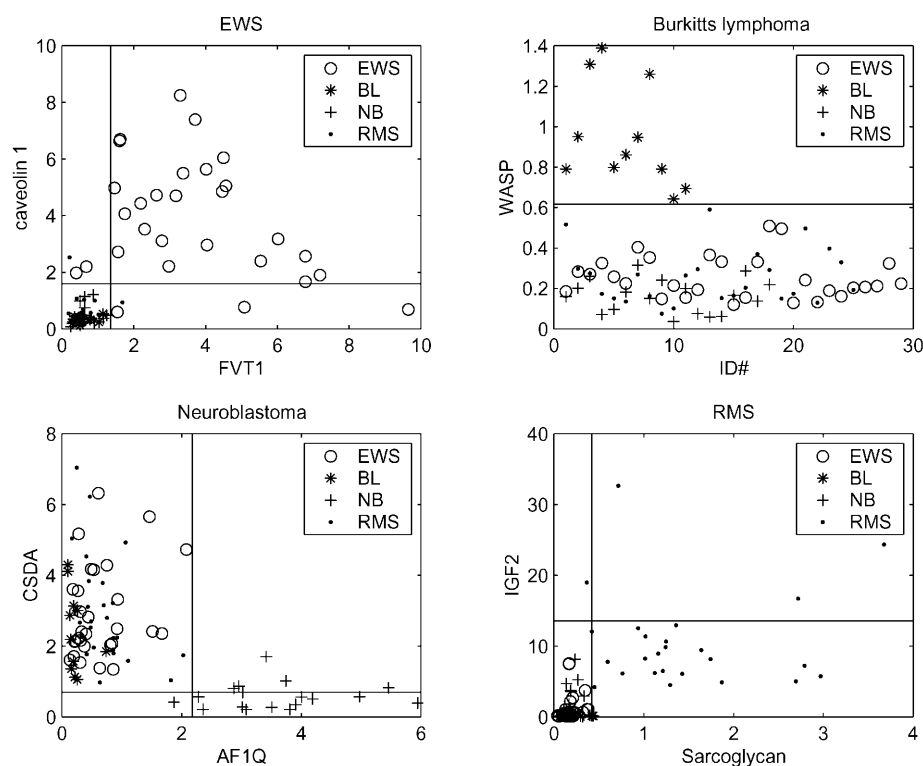


Fig. 2. Graphical representation of the seven rules discovered for the SRBCT data set. The lines mark thresholds for the expression levels of discovered genes. a) The two rules that separate EWS from all other cancer types. b) BL is perfectly separated from all other examples by one gene. c) NB is distinguished by high expression of AF1Q and low expression of CSDA. d) RMS is separated by high expression of sarcoglycan alpha and IGF2

From a histological point of view, the four tumor types represented in the SRBCT data set are rather similar. However, we found that the four classes can be distinguished quite easily due to a number of more or less obvious differences in their respective expression patterns. From a molecular genetics point of view, the cancer types are thus rather disparate. The extensive literature regarding cancer-associated genes has allowed us to verify relatedness between genes and cancer described by a small set of rules.

Inspection of classification rules derived from numerical attributes typically gives the impression of the rules being very specific. However, since most rule sets gener-

ated only employ one split point for every gene used, the rules can easily be translated into qualitative conditions, *i.e.* whether a particular gene is relatively up- or down-regulated, when distinguishing between different classes, such as tumor types.

One major goal of gene expression array analysis is to discover new and interesting pathways describing causal dependencies underlying characteristic cellular behavior. We believe that the methods described in this paper are useful tools can contribute to a complete understanding of these pathways. We also believe that this approach can be applicable to neighbouring areas of gene expression array classification where phenotypes are to be correlated with global gene expression patterns.

References

1. Golub, T. R., Slonim, D. K., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J. P., Coller, H., Loh, M. L., Downing, J. R., Caligiuri, M. A., Bloomfield, C. D. and Lander, E. S. (1999) Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science*, 286, 531-537
2. Alon, U., Barkai, N., Notterman, D. A., Gish, K., Ybarra, S., Mack, S. Y. D. and Levine, A. J. (1999) Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proc. Natl. Acad. Sci.*, **96**, 6745-6750.
3. Khan, J., Wei, J. S., Rign  , M., Saal, L. H., Ladanyi, M., Westermann, F., Berthold, F., Schwab, M., Antonescu, C. R., Peterson, C. and Meltzer, P. S. (2001) Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks. *Nature Medicine*, 7, 673-679.
4. Ben-Dor, A., Bruhn, L., Friedman, N., Nachman, I., Schummer, M. and Yakhini, Z. (2000) Tissue classification with gene expression profiles. In *Proceedings of the 4th International Conference on Computational Molecular Biology (RECOMB)* Universal Academy Press, Tokyo.
5. Furey, T. S., Cristianini, N., Duffy, N., Bednarski, D. W., Schumm, M. and Haussler, D. (2000) Support vector machine classification and validation of cancer tissue samples using microarray expression data. *Bioinformatics*, 16, 906-914.
6. Keller, A. D., Schummer, M., Hood, L. and Ruzzo, W. L. (2000) Bayesian Classification of DNA Array Expression Data. Technical Report, University of Washington.
7. Zhang, H., Yu, C. Y., Singer, B. and Xiong, M. (2001) Recursive partitioning for tumor classification with gene expression microarray data. *Proc. Natl. Acad. Sci.*, 98, 6730-6735
8. Guyon, I., Weston, J., Barnhill, S., and Vapnik, V. (2002) Gene Selection for Cancer Classification using Support Vector Machines. *Machine Learning*, 46 (1-3): 389-422.
9. Quinlan, J. R. (1986) Induction of decision trees. *Machine Learning*, 1, 81-106
10. Rivest, R. L. (1987) Learning Decision Lists. *Machine Learning*, 2, 229-246
11. Clark, P. and Niblett, T. (1989) The CN2 Induction Algorithm. *Machine Learning*, 3, 261-283

12. Boström, H. (2001) Virtual Predict User Manual. Virtual Genetics Laboratory AB, available from <http://www.vglab.com>
13. Boström, H. and Asker, L. (1999) Combining Divide-and-Conquer and Separate-and-Conquer for Efficient and Effective Rule Induction. Proc. of the Ninth International Workshop on Inductive Logic Programming, LNAI Series 1634, Springer, 33-43
14. Fayyad, U. and Irani, K. (1992) On the Handling of Continuous Valued Attributes in Decision Tree Generation. Machine Learning, 8, 87-102
15. Cestnik, B. and Bratko, I. (1991) On estimating probabilities in tree pruning. Proc. of the Fifth European Working Session on Learning, Springer, 151-163
16. Quinlan and Rivest (1989) □Inferring Decision Trees Using the Minimum Description Length Principle□ Information and Computation 80(3) (1989) 227-248
17. Freund, Y. and Schapire, R. E. (1996) Experiments with a new boosting algorithm. Machine Learning: Proceedings of the Thirteenth International Conference, 148-156
18. Boström, H. (1995) Covering vs. Divide-and-Conquer for Top-Down Induction of Logic Programs. Proc. of the Fourteenth International Joint Conference on Artificial Intelligence, Morgan Kaufmann 1194-1200
19. Cohen, W. W. (1995) Fast Effective Rule Induction. Machine Learning: Proc. of the 12th International Conference, Morgan Kaufmann, 115-123
20. Tahir, S. A., Yang, G., Ebara, S., Timme, T. L., Satoh, T., Li, L., Goltsov, A., Ittmann, M., Morrisett, J. D. and Thompson, T. C. (2001) Secreted caveolin-1 stimulates cell survival/clonal growth and contributes to metastasis in androgen-insensitive prostate cancer. Cancer Res., 61, 3882-3885
21. Fine, S. W., Lisanti, M. P., Galbiati, F. and Li, M. (2001) Elevated expression of caveolin-1 in adenocarcinoma of the colon. Am. J. Clin. Pathol., 115, 719-724
22. Rimokh, R., Gadoux, M., Berthias, M. F., Berger, F., Garoscio, M., Delage, G., Germain, D. and Magaud, J. P. (1993) FVT-1, a novel human transcription unit affected by variant translocation t(2;18)(p11;q21) of follicular lymphoma. Blood, 81, 136-142
23. Busson-Le Coniat, M., Salomon-Nguyen, F., Hillion, J., Bernard, O. A. and Berger, R. (1999) MLL-AF1q fusion resulting from t(1;11) in acute leukemia. Leukemia, 13, 302-6
24. Coles, L. S., Diamond, P., Occhiodoro, F., Vadas, M. A. and Shannon, M. F. (1996) Cold shock domain proteins repress transcription from the GM-CSF promoter. Nucleic Acids Res., 24, 2311-2317
25. Duclos, F., Straub, V., Moore, S. A., Venzke, D. P., Hrstka, R. F., Crosbie, R. H., Durbeej, M., Lebakken, C. S., Ettinger, A. J., van der Meulen, J., Holt, K. H., Lim, L. E., Sanes, J. R., Davidson, B. L., Faulkner, J. A., Williamson, R. and Campbell, K. P. (1998) Progressive muscular dystrophy in alpha-sarcoglycan-deficient mice. J. Cell. Biol., 142, 1461-1471
26. El-Badry, O. M., Minniti, C., Kohn, E. C., Houghton, P. J., Daughaday, W. H. and Helman, L. J. (1990) Insulin-like growth factor II acts as an autocrine growth and motility factor in human rhabdomyosarcoma tumors. Cell Growth Differ., 1, 325-331

Clustering Ontology-Based Metadata in the Semantic Web

Alexander Maedche and Valentin Zacharias

FZI Research Center for Information Technologies at the
University of Karlsruhe, Research Group WIM
D-76131 Karlsruhe, Germany
{maedche,zach}@fzi.de
<http://www.fzi.de/wim>

Abstract. The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation. Recently, different applications based on this vision have been designed, e.g. in the fields of knowledge management, community web portals, e-learning, multimedia retrieval, etc. It is obvious that the complex metadata descriptions generated on the basis of pre-defined ontologies serve as perfect input data for machine learning techniques. In this paper we propose an approach for clustering ontology-based metadata. Main contributions of this paper are the definition of a set of similarity measures for comparing ontology-based metadata and an application study using these measures within a hierarchical clustering algorithm.

1 Introduction

The Web in its' current form is an impressive success with a growing number of users and information sources. However, the heavy burden of accessing, extracting, interpreting and maintaining information is left to the human user. Recently, Tim Berners-Lee, the inventor of the WWW, coined the vision of a Semantic Web¹ in which background knowledge on the meaning of Web resources is stored through the use of machine-processable metadata. The Semantic Web should bring structure to the content of Web pages, being an extension of the current Web, in which information is given a well-defined meaning. Recently, different applications based on this Semantic Web vision have been designed, including scenarios such as knowledge management, information integration, community web portals, e-learning, multimedia retrieval, etc. The Semantic Web relies heavily on formal ontologies that provide shared conceptualizations of specific domains and on metadata defined according these ontologies enabling comprehensive and transportable machine understanding.

Our approach relies on a set of similarity measures that allow to compute similarities between ontology-based metadata along different dimensions. The

¹ <http://www.w3.org/2001/sw/>

similarity measures serve as input to hierarchical clustering algorithm. The similarity measures and the overall clustering approach have been applied on real world data, namely the CIA world fact book². In the context of this empirical evaluation and application study we have obtained promising results.

Organization. Section 2 introduces ontologies and metadata in the context of the Semantic Web. Section 3 focuses on three different similarity measuring dimensions for ontology-based metadata. Section 4 provides insights into our empirical evaluation and application study and the results we obtained when applying our clustering technique on Semantic Web data. Before we conclude and outline the next steps within our work, we give an overview on related work in Section 5.

2 Ontologies and Metadata in the Semantic Web

As introduced earlier the term "Semantic Web" encompasses efforts to build a new WWW architecture that enhances content with formal semantics. This will enable automated agents to reason about Web content, and carry out more intelligent tasks on behalf of the user. Figure 1 illustrates the relation between "ontology", "metadata" and "Web documents". It depicts a small part of the CIA world fact book ontology. Furthermore, it shows two Web pages, viz. the CIA fact book pages about the country Argentina and the home page of the United Nations, respectively, with semantic annotations given in an XML serialization of RDF-based metadata descriptions³. For the country and the organization there are metadata definitions denoted by corresponding uniform resource identifiers (URIs) ([HTTP://WWW.CIA.ORG/COUNTRY#AG](http://www.cia.org/country#AG) and [HTTP://WWW.UN.ORG#ORG](http://www.un.org#ORG)). The URIs are typed with the concepts COUNTRY and ORGANIZATION. In addition, there is a relationship instance between the country and organisation: Argentina *ISMEMBEROF* United Nations.

In the following we introduce a ontology and metadata model. We here only present the part of our overall model that is actually used within our ontology-based metadata clustering approach⁴. The model that is introduced in the following builds the core backbone for the definition of similarity measures.

Ontologies. In its classical sense ontology is a philosophical discipline, a branch of philosophy that deals with the nature and the organization of being. In its most prevalent use an ontology refers to an engineering artifact, describing a formal, shared conceptualization of a particular domain of interest [4].

Definition 1 (Ontology Structure). *An ontology structure is a 6-tuple $\mathcal{O} := \{\mathcal{C}, \mathcal{P}, \mathcal{A}, \mathcal{H}^{\mathcal{C}}, prop, att\}$, consisting of two disjoint sets $\mathcal{C}$ and $\mathcal{P}$ whose elements*

² <http://www.cia.gov/cia/publications/factbook/>

³ The Resource Description Format (RDF) is a W3C Recommendation for metadata representation, <http://www.w3c.org/RDF>.

⁴ A more detailed definition is available in [7].

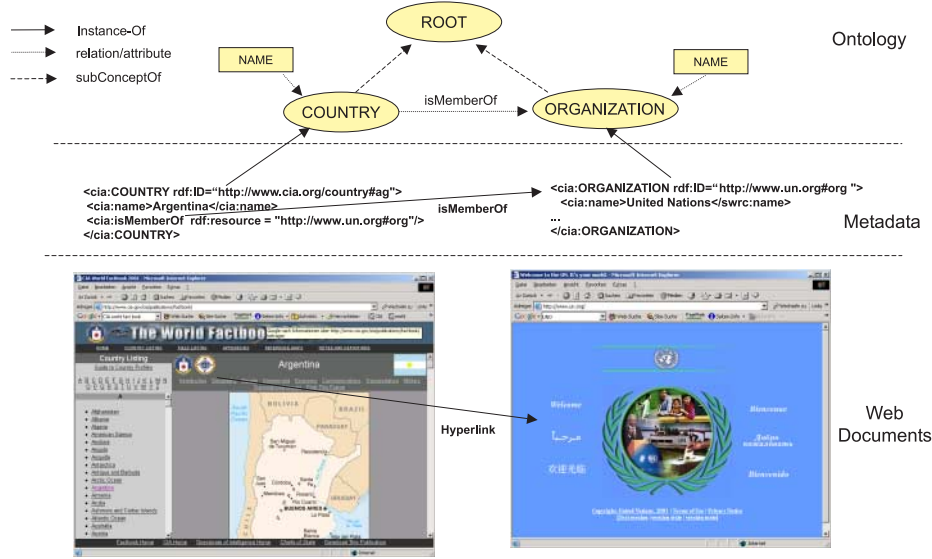


Fig. 1. Ontology, metadata and Web documents

are called concepts and relation identifiers, respectively, a **concept hierarchy** $\mathcal{H}^C$: $\mathcal{H}^C$ is a directed, transitive relation $\mathcal{H}^C \subseteq \mathcal{C} \times \mathcal{C}$ which is also called concept taxonomy. $\mathcal{H}^C(C_1, C_2)$ means that C_1 is a sub-concept of C_2 , a **function** $\text{prop} : \mathcal{P} \rightarrow \mathcal{C} \times \mathcal{C}$, that relates concepts non-taxonomically (The function $\text{dom} : \mathcal{P} \rightarrow \mathcal{C}$ with $\text{dom}(P) := \Pi_1(\text{rel}(P))$ gives the domain of P , and $\text{range} : \mathcal{P} \rightarrow \mathcal{C}$ with $\text{range}(P) := \Pi_2(\text{rel}(P))$ give its range. For $\text{prop}(P) = (C_1, C_2)$ one may also write $P(C_1, C_2)$). A specific kind of relations are attributes $\mathcal{A}$. The **function** $\text{att} : \mathcal{A} \rightarrow \mathcal{C}$ relates concepts with literal values (this means $\text{range}(\mathcal{A}) := \text{STRING}$)

Example. Let us consider a short example of an instantiated ontology structure as depicted in Figure 2. Here on the basis of $\mathcal{C} := \{ \text{COUNTRY}, \text{RELIGION}, \text{RELIGION} \}$, $\mathcal{P} := \{ \text{BELIEVE}, \text{SPEAK}, \text{BORDERS} \}$, $\mathcal{A} := \{ \text{POPGRW} \}$ the relations $\text{BELIEVE}(\text{COUNTRY}, \text{RELIGION})$, $\text{SPEAK}(\text{COUNTRY}, \text{LANGUAGE})$, $\text{BORDERS}(\text{COUNTRY}, \text{COUNTRY})$ with its domain/range restrictions and the attribute $\text{POPGRW}(\text{COUNTRY})$ are defined.

Ontology-Based Metadata. We consider the term metadata as synonym to instances of ontologies and define a so-called metadata structure as following:

Definition 2 (Metadata Structure). A metadata structure is a 6-tupel $\text{MD} := \{ \mathcal{O}, \mathcal{I}, \mathcal{L}, \text{inst}, \text{instr}, \text{instl} \}$, that consists of an ontology $\mathcal{O}$, a set $\mathcal{I}$ whose elements are called instance identifiers (correspondingly C , P and I are disjoint), a set of literal values $\mathcal{L}$, a function $\text{inst} : \mathcal{C} \rightarrow 2^{\mathcal{I}}$ called **concept instantiation** (For $\text{inst}(C) = I$ one may also write $C(I)$), and a function

$instr : \mathcal{P} \rightarrow 2^{\mathcal{I} \times \mathcal{I}}$ called **relation instantiation** (For $inst(P) = \{I_1, I_2\}$ one may also write $P(I_1, I_2)$). The **attribute instantiation** is described via the function $instl : \mathcal{P} \rightarrow 2^{\mathcal{I} \times \mathcal{L}}$ relates instances with literal values.

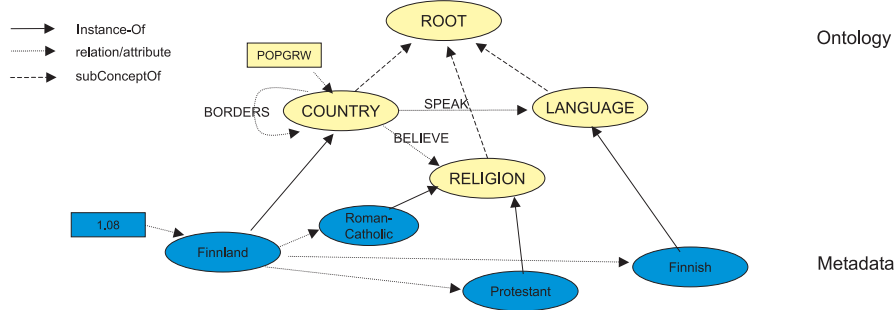


Fig. 2. Example ontology and metadata

Example. Here, the following metadata statements according to the ontology are defined. Let $\mathcal{I} := \{\text{FINNLAND}, \text{ROMAN-CATHOLIC}, \text{PROTESTANT}, \text{FINNISH}\}$. $inst$ is applied as follows: $inst(\text{FINNLAND}) = \text{COUNTRY}$, $inst(\text{ROMAN-CATHOLIC}) = \text{RELIGION}$, $inst(\text{PROTESTANT}) = \text{RELIGION}$, $inst(\text{FINNISH}) = \text{LANGUAGE}$. Furthermore, we define relations between the instances and an attribute for the country instance. This is done as follows: We define $\text{BELIEVE}(\text{FINNLAND}, \text{ROMAN-CATHOLIC})$, $\text{BELIEVE}(\text{FINNLAND}, \text{PROTESTANT})$, $\text{SPEAK}(\text{FINNLAND}, \text{FINNISH})$ and $\text{POPGRW}(\text{FINNLAND}, "1.08")$.

3 Measuring Similarity on Ontology-Based Metadata

As mentioned earlier, clustering of objects requires some kind of similarity measure that is computed between the objects. In our specific case the objects are described via ontology-based metadata that serve as input for measuring similarities. Our approach is based on similarities using the instantiated ontology structure and the instantiated metadata structure as introduced earlier in parallel. Within the overall similarity computation approach, we distinguish the following three dimensions:

- **Taxonomy similarity:** Computes the similarity between two instances on the basis of their corresponding concepts and their position in $\mathcal{H}^C$.
- **Relation similarity:** Compute the similarity between two instances on the basis of their relations to other objects.
- **Attribute similarity:** Computes the similarity between two instances on the basis of their attributes and attribute values.

Taxonomy Similarity. The taxonomic similarity computed between metadata instances relies on the concepts with their position in the concept taxonomy $\mathcal{H}^C$. The so-called upwards cotopy (SC) [7] is the underlying measure to compute the semantic distance in a concept hierarchy.

Definition 3 (Upwards Cotopy (UC)).

$$UC(C_i, \mathcal{H}^C) := \{C_j \in \mathcal{C} \mid \mathcal{H}^C(C_i, C_j) \vee C_j = C_i\}.$$

The semantic characteristics of $\mathcal{H}^C$ are utilized: The attention is restricted to super-concepts of a given concept C_i and the reflexive relationship of C_i to itself. Based on the definition of the upwards cotopy (UC) the concept match (CM) is then defined:

Definition 4 (Concept Match).

$$CM(C_1, C_2) := \frac{|(UC(C_1, \mathcal{H}^C) \cap (UC(C_2, \mathcal{H}^C)))|}{|(UC(C_1, \mathcal{H}^C)) \cup (UC(C_2, \mathcal{H}^C))|}.$$

Example. Figure 3 depicts the example scenario for computing CM graphically. The upwards cotopy $UC(\text{CHRISTIANISM}, \mathcal{H}^C)$ is given by $UC(\{\{\text{CHRISTIANISM}\}\}, \mathcal{H}^C) = \{\text{CHRISTIANISM}, \text{RELIGION}, \text{ROOT}\}$. The upwards cotopy $UC(\{\{\text{MUSLIM}\}\}, \mathcal{H}^C)$ is computed by $UC(\{\{\text{MUSLIM}\}\}, \mathcal{H}^C) = \{\text{MUSLIM}, \text{RELIGION}, \text{ROOT}\}$. Based on the upwards cotopy one can compute the concept match CM between two given specific concepts. The concept match CM between MUSLIM and CHRISTIANISM is given as $\frac{1}{2}$.

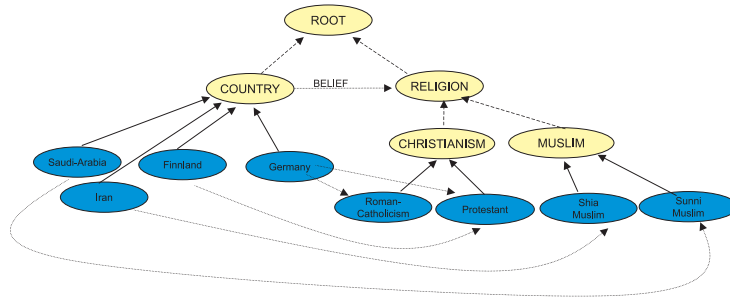


Fig. 3. Example for computing similarities

Definition 5 (Taxonomy Similarity).

$$TS(I_1, I_2) = \begin{cases} 1 & \text{if } I_1 = I_2 \\ \frac{CM(C(I_1), C(I_2))}{2} & \text{otherwise} \end{cases}$$

The taxonomy similarity between SHIA MUSLIM to PROTESTANT results in $\frac{1}{4}$.

Relation similarity. Our algorithm is based on the assumption that if two instances have the same relation to a third instance, they are more likely similar than two instances that have relations to totally different instances. Thus, the similarity of two instances depends on the similarity of the instances they have relations to. The similarity of the referred instances is once again calculated using taxonomic similarity. For example, assuming we are given two concepts COUNTRY and RELIGION and a relation BELIEVE(COUNTRY, RELIGION). The algorithm will infer that specific countries believing in catholicism and protestantism are more similar than either of these two compared to hinduism because more countries have both catholics and protestants than a combination of either of these and hindis.

After this overview, let's get to the nitty gritty of really defining the similarity on relations. We are comparing two instances I_1 and I_2 , $I_1, I_2 \in \mathcal{I}$. From the definition of the ontology we know that there is a set of relations P_1 that allow instance I_1 either as domain, as range or both (Likewise there is a set P_2 for I_2). Only the intersection $P_{CO} = P_1 \cap P_2$ will be of interest for relation similarity because differences between P_1 and P_2 are determined by the taxonomic relations, which are already taken into account by the taxonomic similarity. The set P_{CO} of relations is differentiated between relations allowing I_1 and I_2 as range - P_{CO-I} , and those that allow I_1 and I_2 as domain - P_{CO-O} .

Definition 6 (Incoming P_{CO-I} and Outgoing P_{CO-O} Relations).

Given $\mathcal{O} := \{\mathcal{C}, \mathcal{P}, \mathcal{A}, \mathcal{H}^{\mathcal{C}, \mathcal{P}}, prop, att\}$ and instances I_1 and I_2 let:

$$\begin{aligned} H^{trans} &:= \left\{ (a, b) : (\exists a_1 \dots a_n \in C : H^C(a, a_1) \dots H^C(a_n, b)) \right\} \\ P_{CO-I_i}(I_i) &:= \{ R : R \in \mathcal{P} \wedge ((C(I_i), range(R)) \in H^{trans}) \} \\ P_{CO-O_i}(I_i) &:= \{ R : R \in \mathcal{P} \wedge ((C(I_i), domain(R)) \in H^{trans}) \} \\ P_{CO-I}(I_i, I_j) &:= P_{CO-I_i}(I_i) \cap P_{CO-I_j}(I_j) \\ P_{CO-O}(I_i, I_j) &:= P_{CO-O_i}(I_i) \cap P_{CO-O_j}(I_j) \end{aligned}$$

In the following we will only look at P_{CO-O} , but everything applies to P_{CO-I} as well. Before we continue we have to note an interesting aspect: For a given ontology with a relation P_x there is a minimum similarity greater than zero between any two instances that are source or target of an instance relation - $MinSim_{s(P_x)}$ and $MinSim_{t(P_x)}$ ⁵. Ignoring this will increase the similarity of two instances with relations to the most different instances when compared to two instances that simply don't define this relation. This is especially troublesome when dealing with missing values. For each relation $P_n \in P_{CO-O}$ and each instance I_i there exists a set of instance relations $P_n(I_i, I_x)$. We will call the set of instances I_x the associated instances A_s .

Definition 7 (Associated Instances).

$$A_s(P, I) := \{ I_x : I_x \in \mathcal{I} \wedge P(I, I_x) \}$$

⁵ Range and domain specify a concept and any two instances of this concept or one of its sub-concepts will have a taxonomic similarity bigger than zero

The task of comparing the instances I_1 and I_2 with respect to relation P_n boils down to comparing $A_s(P_n, I_1)$ with $A_s(P_n, I_2)$. This is done as follows:

Definition 8 (Similarity for One Relation).

$$OR(I_1, I_2, P) = \begin{cases} MinSim_t(P) & \text{if } A_s(P, I_1) = \emptyset \vee A_s(P, I_2) = \emptyset \\ \left(\frac{\sum_{(a \in A_s(P, I_1))} \max\{sim(a, b) | b \in A_s(P, I_2)\}}{|A_s(P, I_1)|} \right) & \text{if } |A_s(P, I_1)| \geq |A_s(P, I_2)| \\ \left(\frac{\sum_{(a \in A_s(P, I_2))} \max\{sim(a, b) | b \in A_s(P, I_1)\}}{|A_s(P, I_2)|} \right) & \text{otherwise} \end{cases}$$

Finally, the results for all $P_n \in P_{co-O}$ and $P_n \in P_{co-I}$ are combined by calculating their arithmetic mean.

Definition 9 (Relational Similarity).

$$RS(I_1, I_2) := \frac{\sum_{p \in P_{co-I}} OR(I_1, I_2, p) + \sum_{p \in P_{co-O}} OR(I_1, I_2, p)}{|P_{co-I}| + |P_{co-O}|}$$

The last problem that remains is the recursive nature of process of calculating similarities that may lead to infinite cycles, but it can be easily solved by imposing a maximum depth for the recursion. After reaching this maximum depth the arithmetic mean of taxonomic and attribute similarity is returned.

Example. Assuming based on Figure 3 we compare FINNLAND and GERMANY, we see that the set of common relations only contains the BELIEF relation. As the next step we compare the sets of instances associated with GERMANY and FINNLAND through the belief relation - that's {ROMAN-CATHOLICISM, PROTESTANT} for GERMANY and PROTESTANT for FINNLAND. The similarity function for PROTESTANT compared with PROTESTANT returns one because they are equal, but the similarity of PROTESTANT compared with ROMAN-CATHOLICISM once again depends on their relational similarity. If we assume the the maximum depth of recursion is set to one, the relational similarity between ROMAN-CATHOLICISM and PROTESTANT is 0.5⁶. So finally the relational similarity between FINNLAND and GERMANY in this example is 0.75.

Attribute Similarity. Attribute similarity focuses on similar attribute values to determine the similarity between two instances. As attributes are very similar to relations⁷, most of what is said for relations also applies here.

Definition 10 (Compared Attributes for Two Instances).

$$PAi(I_i) := \{A : A \in \mathcal{A}\}$$

$$PA(I_i, I_j) := PAi(I_i) \cap PAi(I_j)$$

⁶ The set of associated instances for PROTESTANT contains FINNLAND and GERMANY, the set for ROMAN-CATHOLICISM just GERMANY.

⁷ In RDF attributes are actually relations with a range of literal.

Definition 11 (Attribute Values).

$$A_s(A, I_i) := \{L_x : L_x \in \mathcal{L} \wedge A(I_i, L_x)\}$$

Only the members of the sets A_s defined earlier are not instances but literals and we need a new similarity method to compare literals. Because attributes can be names, date of birth, population of a country, income etc. comparing them in a senseful way is very difficult. We decided to try to parse the attribute values as a known data type (so far only date or number)⁸ and to do the comparison on the parsed values. If it's not possible to parse all values of a specific attribute, we ignore this attribute. But even if numbers are compared, translating a numeric difference to a similarity value $[0, 1]$ can be difficult. For example comparing the attribute population of a country a difference of 4 should yield a similarity value very close to 1, but comparing the attribute “average number of children per woman” the same numeric difference value should result in a similarity value close to 0. To take this into account, we first find the maximum difference between values of this attribute and then calculate the similarity as $1 - (\text{Difference} / \text{max Difference})$.

Definition 12 (Literal Similarity).

$$slsim(\mathcal{A}, \mathcal{A}) \rightarrow [0, 1]$$

$$mlsim := \max \{slsim(A_1, A_2) : A_1 \in \mathcal{A} \wedge A_2 \in \mathcal{A}\}$$

$$lsim(A_i, A_j, A) := \frac{slsim(A_i, A_j)}{mlsim(A)}$$

And last but not least, unlike for relations the minimal similarity when comparing attributes is always zero.

Definition 13 (Similarity for One Attribute).

$$OA(I_1, I_2, A) := \begin{cases} 0 & \text{if } A_s(A, I_1) = \emptyset \vee A_s(A, I_2) = \emptyset \\ \frac{\sum_{(a \in A_s(A, I_1))} \max\{lsim(a, b, A) \mid b \in A_s(A, I_2)\}}{|A_s(A, I_1)|} & \text{if } |A_s(A, I_1)| \geq |A_s(A, I_2)| \\ \frac{\sum_{(a \in A_s(A, I_2))} \max\{lsim(a, b, A) \mid b \in A_s(A, I_1)\}}{|A_s(A, I_2)|} & \text{otherwise} \end{cases}$$

Definition 14 (Attribute Similarity).

$$AS(I_1, I_2) := \frac{\sum_{a \in P_{A(I_1, I_2)}} OA(I_1, I_2, a)}{|P_{A(I_1, I_2)}|}$$

⁸ For simple string data types one may use a notion of string similarity: The *edit distance* formulated by Levenshtein [6] is a well-established method for weighting the difference between two strings. It measures the minimum number of token insertions, deletions, and substitutions required to transform one string into another using a dynamic programming algorithm. For example, the edit distance, ed , between the two lexical entries “TopHotel” and “Top_Hotel” equals 1, $ed(\text{“TopHotel”}, \text{“Top\_Hotel”}) = 1$, because one insertion operation changes the string “TopHotel” into “Top_Hotel”.

Combined Measure. The combined measure uses the three dimensions introduced above in a common measure. This done by calculating the weighted arithmetic mean of attribute, relation and semantic similarity.

Definition 15 (Similarity Measure).

$$\text{sim}(I_i, I_j) := \frac{t \times TS(I_i, I_j) + r \times RS(I_i, I_j) + a \times AS(I_i, I_j)}{t + r + a}$$

The weights may be adjusted according to the given data set the measures should be applied, e.g. within our empirical evaluation we used a weight of 2 for relation similarity, because most of the overall information of the ontology and the associated metadata was contained in the relations.

Hierarchical Clustering. Based on the similarity measures introduced above we may now apply a clustering technique. Hierarchical clustering algorithms are preferable for concept-based learning. They produce hierarchies of clusters, and therefore contain more information than non-hierarchical algorithms. [8] describes the bottom-up algorithm we use within our approach. It starts with a separate cluster for each object. In each step, the two most similar clusters are determined, and merged into a new cluster. The algorithm terminates when one large cluster containing all objects has been formed.

4 Empirical Evaluation

We have empirically evaluated our approach for clustering ontology-based metadata based on the different similarity measures and the clustering algorithm introduced above. We used the well-known CIA world fact book data set as input⁹ available in the form of a MONDIAL database¹⁰. Due to a lack of currently available ontology-based metadata on the Web, we converted a subset of MONDIAL in RDF and modeled a corresponding RDF-Schema for the databases (on the basis of the ER model also provided by MONDIAL). Our subset of the MONDIAL database contained the concepts COUNTRY, LANGUAGE, ETHNIC-GROUP, RELIGION and CONTINENT. Relations contained where

- SPEAK(COUNTRY, LANGUAGE),
- BELONG(COUNTRY, ETHNIC-GROUP),
- BELIEVE(COUNTRY, RELIGION),
- BORDERS(COUNTRY, COUNTRY) and
- ENCOMPASSES(COUNTRY, CONTINENT).

We also converted the attributes infant mortality and population growth of the concept COUNTRY. As there is no pre-classification of countries, we decided

⁹ <http://www.cia.gov/cia/publications/factbook/>

¹⁰ <http://www.informatik.uni-freiburg.de/~may/Mondial/>

to empirically evaluate the cluster against the country clusters we know and use in our daily live (like european countries, scandinavian countries, arabic countries etc). Sadly there is no further taxonomic information for the concepts RELIGION, ETHNIC-GROUP or LANGUAGE available within the data set. For our experiments we used the already introduced bottom-up clustering algorithm with a single linkage computation strategy using cosine measure.

Using only relation similarity. Using only the relations of countries for measuring similarities we got clusters resembling many real world country clusters, like the european countries, the former soviet republics in the caucasus or such small cluster like {AUSTRIA, GERMANY}. A particular interesting example is the cluster of scandinavian countries depicted in Figure 4 because our data nowhere contains a value like "scandinavian language" or a ethnic group "scandinavian".¹¹ Figure 5 shows another interesting cluster of countries that we know as the

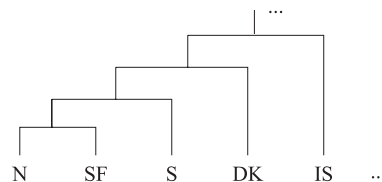


Fig. 4. Example clustering result – scandinavian countries

Middle East¹². The politically interested reader will immediately recognize that Israel is missing. This can be easily explained by observing that Israel, while geographically in the middle east is in terms of language, religion and ethnic group a very different country. More troublesome is that Oman is missing too and this can be only explained by turning to the data set used to calculate the similarities, where we see that Oman is missing many values, for example any relation to language or ethnic group.

Using only attribute similarity. When using only attributes of countries for measuring similarities we had to restrict the clustering to infant mortality and population growth. As infant mortality and population growth are good indicators for wealth of a country, we got cluster like industrialized countries or very poor countries.

¹¹ The meaning of the acronyms in the picture is: N:Norway, SF: Finnland, S: Sweden, DK: Denmark and IS:Island.

¹² The meaning of the acronyms used in the picture is: Q:Qatar, KWT: Kuwait, UAE: United Arab Emirates, SA: Saudi Arabia, JOR: Jordan, RL: Lebanon, IRQ: Iraq, SYR: Syria, YE, Yemen.

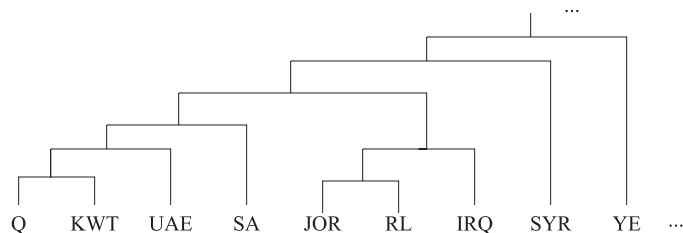


Fig. 5. Example clustering result – middle east

Combining relation and attribute similarity. At first surprisingly the clusters generated with the combination of attribute and relation similarity closely resemble the clusters generated only with relation similarity. But after checking the attribute values of the countries it actually increased our confidence in the algorithm, because countries that are geographically close together, and are similar in terms of ethnic group, religion and language are almost always also similar in terms of population growth and infant mortality. In the few cases where this was not the case the countries were rated far apart, for example Saudi Arabia and Iraq lost its position in the core middle east cluster depicted because of their high infant mortality¹³.

Summarization of results. Due to the lack of pre-classified countries and due to the subjectivity of clustering in general, we had to restrict our evaluation procedure to an empirical evaluation of the cluster we obtained against the country clusters we know and use in our daily live. It has been seen that using our attribute and relation similarity measures combined with a hierarchical clustering algorithm results in reasonable clusters of countries taking into account the very different aspects a country may be described and classified.

5 Related Work

One work closely related to ours was done by Bisson [1]. In [1] it is argued that object-based representation systems should use the notion of similarity instead of the subsumption criterion for classification and categorization. The similarity between attributes is obtained by calculating the similarity between the values for common attributes (taking upper and lower bound for this attribute into account) and combining them. For a symmetrical similarity measure they are combined by dividing the weighted sum of the similarity values for the common attributes by the weights of all attribute that occur in one of the compared

¹³ It may be surprising for such a rich country, but according to the CIA world fact book the infant mortality rate in Saudi Arabia (51 death per 1000 live born children) much closer resembles that of sanctioned Iraq (60) than that of much poorer countries like Syria (33) or Lebanon (28)

individuals. For an asymmetrical similarity measure the sum is divided using just the weights for the attributes that occur in the first argument individual, thereby allowing to calculate the degree of inclusion between first and second argument. The similarity for relations is calculated by using the similarity of the individuals that are connected through this relations. The resulting similarity measures are then again combined in the above described symmetrical or asymmetrical way. Compared to the algorithm proposed here the approach proposed by Bisson does not take ontological background knowledge into account.

Similar to our approach a distance-based clustering is introduced in [3] that used RIBL (Relational Instance-Based Learning) for distance computations. RIBL as introduced in [5] is an adaption of a propositional instance-based learner to a first order representation. It uses distance weighted k-nearest neighbor learning to classify test cases. In order to calculate the distance between examples RIBL computes for each example a conjunction of literals describing the objects that are represented by the arguments of the example fact. Given an example fact RIBL first collects all facts from the knowledge base containing at least one of the arguments also contained in the example fact. Depending on a parameter set by the user, the system may then continue to collect all facts that contain at least one of the arguments contained in the earlier selected facts (this goes on until a specified depth is reached). After selecting these facts the algorithm then goes on to calculate the similarity between the examples in a manner similar to the one used by Bisson or described in this paper: The similarity of the objects depends on the similarity of their attribute values and on the similarity of the objects related to them. The calculation of the similarity value is augmented by predicate and attribute weight estimation based on classification feedback¹⁴. But like Bisson's approach RIBL does not use ontological background knowledge¹⁵.

In the context of Semantic Web research, an approach for clustering RDF statements to obtain and refine an ontology has been introduced by [2]. The authors present a method for learning concept hierarchies by systematically generating the most specific generalization of all possible sets of resources - in essence building a subsumption hierarchy using both the intension and extension of newly formed concepts. If an ontology is already present, its information is used to find generalizations - for example generalizing "type of Max is Cat" and "type of Moritz is Dog" to "type of Max, Moritz is Mammal". Unlike the authors of [2] we deliberately chose to use a distance and not a subsumption based clustering because - as for example [2] points out - subsumption based criteria are not

¹⁴ Weight estimation was not used in [3]

¹⁵ It may seem obvious that it is possible to include ontological background information as facts in the knowledge base, but the results would not be comparable to our approach. Assuming we are comparing u_1 , u_2 and have the facts $\text{instance\_of}(u_1, c_1)$, $\text{instance\_of}(u_2, c_2)$. Comparing u_1 and u_2 with respect to instance_of would lead to comparing c_1 and c_2 which in turn lets the algorithm select all facts containing c_1 and c_2 - containing all instances of c_1 and c_2 and their description. Assuming a single root concept and a high depth parameter sooner or later all facts will be selected - resulting not only in a long runtime but also in a very low impact of the taxonomic relations

well equipped to deal with incomplete or incoherent information (something we expect to be very common within the Semantic Web).

6 Conclusion

In this paper we have presented an approach towards mining Semantic Web data, focusing on clustering objects described by ontology-based metadata. Our method has been empirically evaluated on the basis of the CIA world fact book data set that was easily to convert into ontology-based metadata. The results have shown that our clustering method is able to detect commonly known clusters of countries like scandinavian countries or middle east countries.

In the future much work remains to be done. Our empirical evaluation could not be formalized due to the lack of available pre-classifications. The actual problem is that there are no ontological background knowledge. Therefore, we will model country clusters within the CIA world fact book ontology and experiment to which degree the algorithm is able to discover these country clusters. These data set may serve as a future reference data set when experimenting with our Semantic Web mining techniques.

Acknowledgments

The research presented in this paper has been partially funded by Daimler-Chrysler AG, Woerth in the HRMore project. We thank Steffen Staab for providing useful input for defining the taxonomic similarity measure. Furthermore, we thank our student Peter Horn who did the implementation work for our empirical evaluation study.

References

1. G. Bisson. Why and how to define a similarity measure for object based representation systems, 1995. 358
2. A. Delteil, C. Faron-Zucker, and R. Dieng. Learning ontologies from RDF annotations. In A. Maedche, S. Staab, C. Nedellec, and E. Hovy, editors, *Proceedings of IJCAI-01 Workshop on Ontology Learning OL-2001, Seattle, August 2001*, Menlo Park, 2001. AAAI Press. 359
3. W. Emde and D. Wettschereck. Relational instance-based learning. *Proceedings of the 13th International Conference on Machine Learning*, 1996, 1996. 359
4. T. R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 6(2):199–221, 1993. 349
5. M. Kirsten and S. Wrobel. Relational distance-based clustering. pages 261–270. *Proceedings of ILP-98, LNAI 1449*, Springer, 1998, 1998. 359
6. I. V. Levenshtein. Binary Codes capable of correcting deletions, insertions, and reversals. *Cybernetics and Control Theory*, 10(8):707–710, 1966. 355
7. A. Maedche, S. Staab, N. Stojanovic, R. Studer, and Y. Sure. *SEmantic PortAL – The SEAL approach*. to appear in: *Creating the Semantic Web*. D. Fensel et al., MIT Press, MA, Cambridge, 2001. 349, 352
8. C. D. Manning and H. Schuetze. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, Massachusetts, 1999. 356

Iteratively Selecting Feature Subsets for Mining from High-Dimensional Databases

Hiroshi Mamitsuka

Institute for Chemical Research, Kyoto University
Gokasho, Uji, 611-0011, Japan
`mami@kuicr.kyoto-u.ac.jp`

Abstract. We propose a new data mining method that is effective for mining from extremely high-dimensional databases. Our proposed method iteratively selects a subset of features from a database and builds a hypothesis with the subset. Our selection of a feature subset has two steps, i.e. selecting a subset of instances from the database, to which predictions by multiple hypotheses previously obtained are most unreliable, and then selecting a subset of features, the distribution of whose values in the selected instances varies the most from that in all instances of the database. We empirically evaluate the effectiveness of the proposed method by comparing its performance with those of two other methods, including Xing et al.’s one of the latest feature subset selection methods. The evaluation was performed on a real-world data set with approximately 140,000 features. Our results show that the performance of the proposed method exceeds those of the other methods, both in terms of the final predictive accuracy and the precision attained at a recall given by Xing et al.’s method. We have also examined the effect of noise in the data and found that the advantage of the proposed method becomes more pronounced for larger noise levels.

1 Introduction

As the fields to which machine learning or data mining techniques are applied increase, the types of data sets dealt with have also been increasing. In particular, the growth of the size of the data sets in real world applications has been extremely pronounced. In this paper, among the large-scale data sets, we focus on mining from high-dimensional data sets, i.e. data sets with a large number (say 1,000,000) features (attributes), to which a single usual induction algorithm cannot be applied on a normal hardware. Our goal is to efficiently find prediction rules by mining from such a very high-dimensional data set. This type of data set, for example, appears in the process of drug design (or drug discovery). That is, each record in the data set corresponds to a chemical compound and has both an enormous number of features characterizing it and its label of drugability, toxicity etc..

So far, the only way to learn from this type of extremely high-dimensional data sets is to reduce the number of features of the data set by selecting a feature

subset. There are two main techniques for feature subset selection, i.e. the filter and wrapper methods [3,5,6]. In both methods, there are two different approaches in terms of incremental or decremental selection of features. More concretely, the incremental approach starts with a set of no features and adds features one by one to the set, and the decremental approach starts with a set having all features and reduces the number of the features in the set. The incremental approach has to depend on the initial set of features, and thus, the decrementally selected features can be generally said to be more reliable than the incrementally selected ones. Actually, the recently proposed feature subset selection methods, e.g. [4,8,12], belong to the decremental approach. Furthermore, an inductive algorithm used in the decremental wrapper approach has to be fed with the whole of a given high-dimensional data set at the initial process. Therefore, the decremental filter approach can be considered the most practical for the large-size data set considered here.

The method we propose here for a high-dimensional data set does not select a feature subset once, but iteratively selects a feature subset. Our method is closely related to an approach called ‘sequential multi-subset learning with a model-guided instance selection’, which is named in a comprehensive survey of [9] on the methods for up-scaling inductive algorithms. A method categorized in the approach repeats the following two steps: selecting a small subset of a large database, using previously obtained multiple predictive hypotheses, and training a component induction algorithm with the subset to obtain a new hypothesis. Our new method, named Qifs (Query-learning-based iterative feature-subset selection), follows the repetition of the two steps, but selects a subset of features from a given data set, instead of the subset of instances selected in the approach. Selecting a feature subset in Qifs consists of two steps. First, Qifs selects a subset of instances based on the idea of a method of query learning called ‘Query by Committee’ [11]. Using the idea, it predicts a label of each instance of the data set with existing hypotheses and selects the instances to which the predictions are distributed (or split) most evenly. Then, for each feature of the data set, Qifs focuses on the two distributions of feature values, i.e. the feature value distribution of the original all instances and that of the instances selected by the idea of query learning. Qifs selects the features in each of which the feature value distribution of the selected instances varies the most from that of the original all instances. Note that Qifs has the scalability for the number of features, as a method categorized in the above approach does for the size of data instances.

The Query by Committee algorithm selects the instances with maximum uncertainty of predicting label values so that the information gain in instance selection is maximized. In our scenario, the selected features are expected to be more strongly needed than others in terms of the information gain, since the distributions of their values in the selected instances differ the most from those in the original all instances, among all given features.

The purpose of this paper is to empirically evaluate the performance of Qifs using a data set used in KDD Cup 2001 which consists of approximately 140,000 binary features. In our experiments, we compared our new method with two

other methods, including the latest feature subset selection method, proposed by [12]. We used two different component algorithms, C4.5 [10] and a support vector machine (SVM) [2] to test the performance for each of the three methods. Our evaluation was done by five-fold cross validation in all of our experiments.

We first used the original data set for the five-fold cross validation, and found that the performance of Qifs exceeds those of the other two methods in terms of final prediction accuracy. In order to better understand the conditions under which Qifs performs well, we varied the noise level of the original data set. That is, the noise was generated by randomly flipping the binary feature values and the noise level was controlled by varying the ratio of the number of flipped features to the number of all features. We evaluated the performance of the three methods, varying the noise level of the data set so that it was either ten or twenty percent. It was found that for larger noise levels, the significance level by which Qifs out-performed the other two methods in terms of final prediction accuracy became larger. Furthermore, we measured precision and recall (both of which are frequently used in information retrieval literature) of the three methods for the noisy data sets. In terms of the precision and recall, the difference between the precision value of Qifs and those of the other two methods at a same recall value also became larger, for larger noise levels.

All of these experiments show that for an extremely high-dimensional data set, our proposed method is more robust against noise than other currently-used methods for high-dimensional data sets and that our method will be a powerful tool for application to real-world very high-dimensional data sets.

2 The Methods

2.1 Proposed Method

Here, we propose a new method for mining from a very high-dimensional database. The method iteratively selects a feature subset and trains an arbitrary component algorithm as a subroutine with the subset. The method works roughly as follows. (See the pseudocode shown in Fig. 1.)

At the initialization, the method processes the following two different steps. For each feature, it calculates the distribution (number) of feature values in a given database (line 1 of Initialization). To obtain the first hypothesis, it uses an arbitrary algorithm of feature subset selection to obtain a feature subset and applies an arbitrary component learning algorithm to it. (line 2 of Initialization). At each iteration, it first calculates the ‘margin’ of each instance, that is, the difference between the number of votes by the past hypotheses for the most ‘popular’ label, and that for the second most popular label (line 1 of Step 1). It selects N (say a half of all) instances having the least values of margin (line 2 of Step 1). Then, for each feature, it calculates the distribution of feature values in the selected instances and examines the ‘difference’ between the distribution of the selected instances and the previously calculated distribution of the whole given database (line 1 of Step 2). It selects a small (say 500) subset

Input: Number of iterations: T

Component learning algorithm: A

Component feature subset selection algorithm: B

Set of features in a given data set: F

Set of instances in a given data set: S

Number of examples selected at each iteration: N

Training examples with a feature subset at the i -th iteration: E_i

Number of features in E_i : Q

Initialization: 1. For all $z \in F$, calculate the distribution n_z of the feature values.

2. Select Q features $\langle z_1^+, \dots, z_Q^+ \rangle$ by running B on the database and obtain training instances $E_1 = \langle z_1^+, \dots, z_Q^+, y \rangle$ from database.

3. Run A on E_1 and obtain hypothesis h_1 .

For $i = 1, \dots, T$

Step1: 1. For all $x \in S$, calculate ‘margin’ $m(x)$ using past hypotheses $h_1, \dots, h_i$

$$m(x) = \max_y |\{t \leq i : h_t(x) = y\}| - \max_{y \neq y_{\max}(x)} |\{t \leq i : h_t(x) = y\}|$$

where $y_{\max}(x) = \arg \max_y |\{t \leq i : h_t(x) = y\}|$

2. Select N instances having the smallest $m(x)$.

Step 2: 1. For all $z \in F$, calculate distribution n_z^* of the feature values in the selected instances, and calculate ‘difference’ d_z between the two distributions, n_z and n_z^* .

2. Select Q features $\langle z_1^+, \dots, z_Q^+ \rangle$ having the largest d_z and let $E_{i+1} = \langle z_1^+, \dots, z_Q^+, y \rangle$.

3. Run A on E_{i+1} and obtain hypothesis h_{i+1} .

End For

Output: Output final hypothesis given by:

$$h_{fin}(x) = \arg \max_{y \in Y} |\{t \leq T : h_t(x) = y\}|$$

Fig. 1. Algorithm: Query-learning-based iterative feature-subset selection (Qifs)

(Q) of features, whose calculated differences are the largest (line 2 of Step 2) and applies a component inductive algorithm to it to obtain a new hypothesis (line 3 of Step 2). The final hypothesis is defined by the majority vote over all the hypotheses obtained in the above process ¹.

The method first selects the instances that cannot be reliably predicted, and then selects features whose values in the selected instances differ the most from those of all the given instances. Since the method uses the technique of query learning in selecting the instances, we call our method ‘Query-learning-based iterative feature-subset selection’, **Qifs** for short.

The first step of Qifs, i.e. selecting instances, is almost the same as our previously proposed method, called QbagS [7]. QbagS is a method for mining from databases with a large number of instances and is also categorized in the approach named ‘sequential multi-subset learning with a model-guided instance selection’. QbagS first randomly chooses a relatively large number of instances as selectable candidates from the database and then out of the candidates, selects instances to which predictions of previously obtained multiple hypotheses distributed (or split) most evenly, to build a new hypothesis. In [7], we have already shown that for very large and noisy datasets, QbagS out-performed Iv-

¹ Note that the total number of selected features equals to $T \times Q$.

otes, one of the latest error-driven approaches, namely the methods which use label information and select examples on which the current hypotheses make a mistake.

In the second step of Qifs, a simple possible candidate for the difference between the distribution of the selected instances and that of all instances is the square distance, if a feature z is a discrete attribute:

$$d_z = \sum_i \left| \frac{n_z(i)}{\sum_i n_z(i)} - \frac{n_z^+(i)}{\sum_i n_z^+(i)} \right|^2, \quad (1)$$

where d_z is the difference given to a feature z , $n_z(i)$ is the number of all instances in which the value of feature z is i , and $n_z^+(i)$ is the number of the selected instances in which the value of feature z is i . We used the difference given by Eq. (1), in our experiments.

Note that if a feature z is a continuous attribute, we need to modify it to a discrete one to apply a distance (including the square distance described above) to it. There are actually a number of methods to discretize a continuous attribute, e.g. unconditional mixture modeling performed in [12].

2.2 Xing et al.'s Feature Subset Selection Method

Here, we briefly review Xing et al.'s feature selection method [12], to which we compare the performance of our method². We can say that the method is one of the most recent feature subset selection methods, which belong to the decremental filter approach. The method consists of two steps, i.e. information gain ranking and Markov blanket filtering. The method first calculates the information gain for all given features and selects those which have a high information gain. Then, the selected features are reduced one by one using the method of Markov blanket filtering proposed by [4]. We can interpret the two steps roughly as follows: The information gain ranking of the method selects a set of features, each of which is strongly relevant to the label of a given database. Then, out of the obtained set, the Markov blanket filtering one by one removes the features for which a similar feature is contained in the set. Our implementation follows [12] exactly.

3 Empirical Evaluation

We empirically evaluate the performance of our method with those of two other methods, i.e. random iterative feature-subset sampling (hereafter called Rand for short) and Xing et al.'s feature subset selection method (hereafter called Fss for short). Random iterative feature-subset sampling is a strategy which, as done in Qifs, repeats the two steps of sampling a feature subset from a given database and applying a component learning algorithm to it, but it samples a

² We also use Xing et al.'s method as our component feature-subset selection method, i.e. B in Fig. 1

Table 1. Data summary

Data set	# classes	# (binary) features	# training samples	# test samples
Thrombin	2	139,351	1560	390

feature randomly when a feature subset is sampled. Thus, random feature-subset sampling does not use the previously obtained hypotheses and simply repeats random sampling and hypothesis building.

In our experiments, we used C4.5 [10] and a support vector machine (SVM)³ as a component learning algorithm. We used them with no particular options and equal conditions for all three methods, i.e. Qifs, Rand and Fss.

We need to evaluate our method with a data set which has a large number (say 100,000 or more) of features and to which we cannot apply a single usual inductive algorithm on a normal hardware. It is, however, difficult to find such a publicly available large-scale data set; the only one we found is a data set used in KDD Cup 2001⁴. The KDD Cup data set is a real-world data set used for discovering drugs, i.e. small organic molecules which bind to a protein. The data set is a table in which each record, corresponding to a chemical molecule, has a number of binary (0 or 1) features characterizing it and a binary class value of binding to a protein called ‘thrombin’. A class value of ‘A’ for active (binding) and ‘I’ for inactive (non-binding) is given to each chemical compound. The sizes of the originally given training and test data sets are 1909 and 636, respectively. Out of them, we obtain a total of 1950 records (compounds) by mixing them, while removing the records in which all features are zero. Of the 1950, 190 are active compounds. That is, the percentage of the compounds binding to thrombin is 9.74%. We call the data set ‘Thrombin data set’.

In evaluating the Thrombin data set, we compare the ‘final accuracy’ obtained by each of the methods. By final accuracy, we mean an accuracy level large enough that the predictive performance appears to be saturating⁵. We also used the measures of precision and recall, standard performance measures in the field of information retrieval. Note that ‘recall’ is defined as the probability of correct prediction given that the actual label is ‘A’, and ‘precision’ is defined as the probability of correct prediction given that the predicted label is ‘A’.

In all of the experiments, the evaluation was done by five-fold cross validation. That is, we split the data set into five blocks of roughly equal size, and in each trial four out of these five blocks were used as training data, and the last block was reserved for test data. The results (learning curves, final accuracy, precision and recall) were then averaged over the five runs.

³ To be precise, we used SVM^{light} [2].

⁴ <http://www.cs.wisc.edu/~dpage/kddcup2001/>

⁵ To be exact, we compared the predictive accuracy obtained at a certain point with that obtained 1,000 seconds prior to that point. If the difference between them is less than 0.05%, we considered that the predictive accuracy at that point is saturated.

Table 2. Summary of parameter settings in our experiments

Methods	# selected features per iteration(Q)	# selected samples per iteration(N)
Qifs	500	780
Rand	500	-

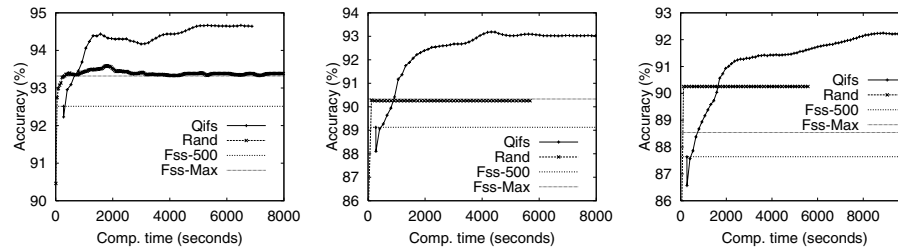


Fig. 2. Learning curves of Qifs and Rand using C4.5 as a component algorithm, with the prediction accuracies of Fss shown for reference, when the noise level is (a) zero, (b) ten and (c) twenty percent

The properties of the Thrombin data set used in our five-fold cross validation are shown in Table 1.

The parameters of Qifs and Rand that were used in our experiments are shown in Table 2. In all of our experiments, we run Fss as follows: We first reduce the number of features to 1,000 by choosing the top features as ranked by information gain, and then we reduce the number one by one to 100 by Markov blanket filtering. In the latter process, we check the prediction accuracy (on separate test data) of a feature set whenever it is obtained, and thus, we obtain a total of 901 subsets and prediction accuracies.

3.1 Cross-Validation on Thrombin Data Set

We show the results of the cross-validation on the Thrombin data set in the form of learning curves in Figures 2 and 3⁶.

Figures 2 (a) and 3 (a) show the learning curves using C4.5 and an SVM as a component learning algorithm, respectively. Note that, in these curves, the average prediction accuracy (on separate test data) is plotted against the total computation time, including disk access time. In both Figures 2 and 3, we also

⁶ The number of iterations of Qifs till the average prediction accuracy is saturated varies widely among the cases of Figures 2 and 3. It ranges from approximately fifty to a couple of hundreds.

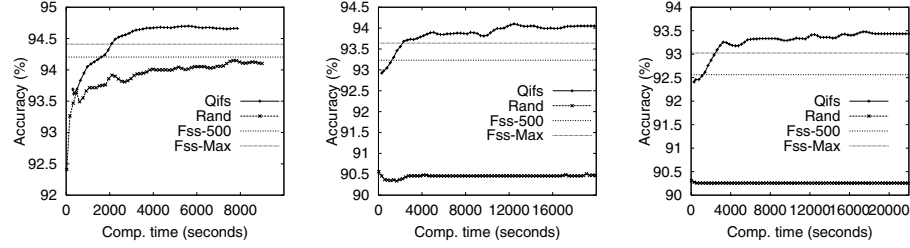


Fig. 3. Learning curves of Qifs and Rand using an SVM as a component algorithm, with the prediction accuracies of Fss shown for reference, when the noise level is (a) zero, (b) ten and (c) twenty percent

add two types of prediction accuracies for Fss for reference⁷. One, shown as Fss-500, is the accuracy obtained when the number of selected features reaches 500, which is the same number as that of the selected features at each iteration in both Qifs and Rand, and the other, shown as Fss-Max, is the highest accuracy attained while reducing the number of features from 1,000 to 100. Note that in a practical situation, we cannot obtain the accuracy given by the Fss-Max for unknown test data, and thus, the performance of our method should be compared with that of not Fss-Max but Fss-500.

In terms of the ‘final prediction accuracy’ results, Qifs out-performed both Rand and Fss. These results are summarized in Table 3, as the case of the noise level of zero percent. The accuracies reached by the three methods for the data set and the ‘ t ’ values of the mean difference significance (pairwise) test for the respective cases are given in the table. The t values are calculated using the following formula :

$$t = \frac{|ave(D)|}{\sqrt{\frac{var(D)}{n}}},$$

where we let D denote the difference between the accuracies of two methods for each data set in our cross-validation, $ave(X)$ the average of X , $var(X)$ the variance of X , and n the number of data sets (five in our case). For the case that $n = 5$, if t is greater than 4.604 then it is more than 99 per cent statistically significant that one achieves higher accuracy than the other.

As is shown in Table 3, for the Thrombin data set, the t values range from 1.15 to 4.78. We can statistically see that the performance of Qifs is slightly (insignificantly) better than those of Rand and Fss.

In order to check the performance of our method in more realistic conditions, we add a kind of noise to the Thrombin data set, varying the noise level. More concretely, we randomly reversed binary feature values of the data set, while

⁷ We show the results of Fss in this form, because a set of 1,000 features is obtained all at once in the first step (information gain ranking) of Fss and thus learning curves cannot be obtained from the process of feature subset selection.

Table 3. Average final accuracies of Qifs and Rand, average accuracies of Fss-500 and Fss-Max, and the t values calculated between Qifs and Rand and between Qifs and Fss-500

Noise level(%)	Component algorithm	Final accuracy (%)				t	t
		Qifs	Rand	Fss-500	Fss-Max	(vs. Rand)	(vs. Fss-500)
0	C4.5	94.67	93.38	92.51	93.33	4.56	4.78
	SVM	94.66	94.12	94.41	94.21	2.55	1.15
10	C4.5	93.03	90.26	89.13	90.31	8.47	11.28
	SVM	94.05	90.46	93.64	93.23	6.05	2.34
20	C4.5	92.21	90.26	87.64	88.51	7.77	7.42
	SVM	93.44	90.26	92.56	93.03	11.10	6.11

keeping the percentage of the number of the reversed features at a certain level, i.e. ten or twenty percent. Figures 2 (b) and (c) show the learning curves of the ten and twenty percent noise level, respectively, using C4.5 as a component algorithm. Figures 3 (b) and (c) also show the learning curves of the two noise levels, using an SVM as a component algorithm. Here, too, in terms of the final prediction accuracy results, Qifs performed better than both Rand and Fss. The final prediction accuracies and the t -values of the mean difference significance test for the cases of the noise levels of ten and twenty percent are also summarized in Table 3.

When the noise level is ten or twenty percent, Qifs did significantly better than Rand or Fss in seven out of the eight cases, in terms of the t values shown in Table 3. We can see, in these results, that for higher noise level, the significance of the difference in the predictive performance between Qifs and the other two methods becomes more pronounced. This result can be visualized by the graph shown in Figure 6 (a). The figure shows how the t values of the mean difference significance test vary as the noise level is changed from zero to twenty percent. We also found from the results that the difference between the performance of Qifs and those of Rand and Fss obtained by using C4.5 as a component learning algorithm, is larger than that obtained by using an SVM.

The precision-recall curves for Qifs, Rand and Fss are shown in Figures 4, using C4.5 as a component learning algorithm. Note that in Fss, prediction is done by a single induction algorithm and only a single pair of recall and precision values is obtained. For Qifs and Rand, the precision-recall curves shown in Figure 4 are those attained after approximately (a) 6,000 (b) 2,000 and (c) 5,000 seconds of computation time. The curves in Figure 5 are those attained after approximately (a) 7,000 (b) 18,000 and (c) 20,000 seconds of computation time using an SVM as a component algorithm.

As shown in the figure, for larger noise levels, the gap between the precision value of Qifs at a certain recall value and those of the other two methods at the same recall value is larger. In particular, as shown in Figure 4 (c), when the noise level reaches twenty percent, the precision of Qifs is approximately 40 percent

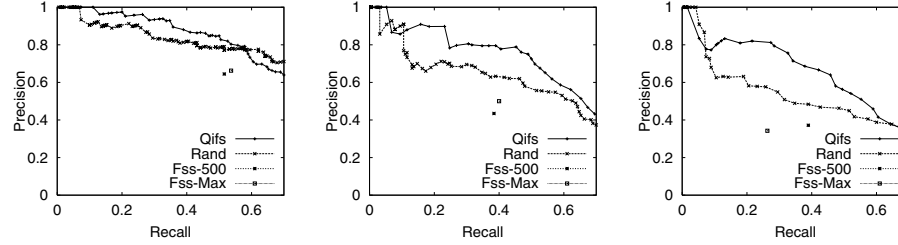


Fig. 4. Precision-recall curves of Qifs, Rand and Fss using C4.5 as a component algorithm, when the noise level is (a) zero (b) ten and (c) twenty percent

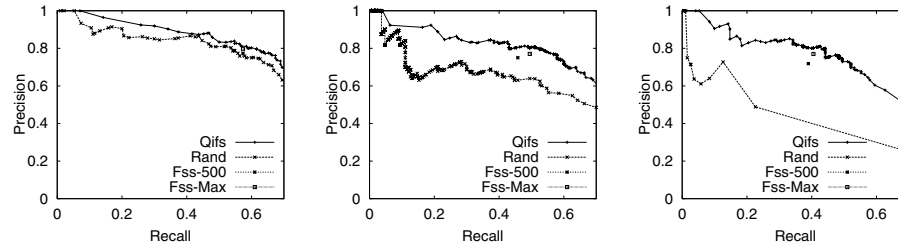


Fig. 5. Precision-recall curves of Qifs, Rand and Fss using an SVM as a component algorithm, when the noise level is (a) zero (b) ten and (c) twenty percent

better than that of Fss-500 at an equal recall value given by Fss-500. One more item of note is that the performance of Rand is better than that of Fss, in the case of using C4.5 as a component learning algorithm. This shows that there is a case in which multiple hypotheses built by sets of randomly selected features achieve a better predictive performance than the single hypothesis built by a set of features carefully selected from all given features.

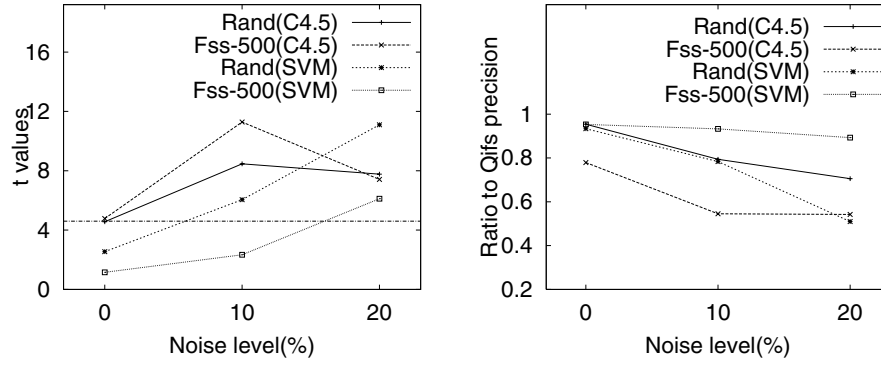
The results of precision-recall curves for all three methods when using either C4.5 or an SVM as a component learning algorithm, are summarized in Table 4. The table shows, for each noise level and component algorithm, the precision values of the three methods at a recall value given by Fss-500. This result can be visualized by the graph shown in Figure 6 (b). The figure shows how the ratio of precision values of Rand or Fss-500 to those of Qifs varies as the noise level is changed from zero to twenty percent.

4 Concluding Remarks

We have proposed a new method for data mining that targets the mining from very *high-dimensional, noisy* data sets. Though the number of features of the data set used here is 140,000, we have shown that the performance of our pro-

Table 4. Precision values of Qifs, Rand and Fss-500, corresponding to recall values given by Fss-500

algorithm	level(%)	Qifs	Rand	Fss-500	(Fss-500)
C4.5	0	0.828(1.0)	0.790(0.954)	0.645(0.779)	0.516
	10	0.796(1.0)	0.632(0.794)	0.434(0.545)	0.384
	20	0.686(1.0)	0.484(0.706)	0.372(0.542)	0.389
SVM	0	0.811(1.0)	0.757(0.933)	0.773(0.953)	0.574
	10	0.804(1.0)	0.630(0.784)	0.75(0.933)	0.458
	20	0.804(1.0)	0.41(0.510)	0.718(0.893)	0.389

**Fig. 6.** (a) t values obtained when varying the noise level and (b) ratio of precision of Rand/Fss to that of Qifs obtained when varying the noise level

posed method is clearly better than that of one of the latest feature subset selection method. The advantage of our method would become more pronounced for more high-dimensional and noisy data sets. The key property of our method which contributes to this advantage is its iterative feature-subset sampling strategy, based on the idea of query learning.

We may compare Qifs with another method which iteratively selects feature subsets using Fss. More concretely, it repeats as follows: it first randomly picks a subset of a given data set, then runs Fss for the subset and obtains a hypothesis with the final feature subset and a learning algorithm. Final prediction is done by the majority votes of the obtained hypotheses. This comparison may be a possible interesting future work.

For mining from a large database which has a large number of both features and instances, we can modify our method to a selective sampling method, in which we use only the instances obtained by the first step of our current method to build a new hypothesis. That is, the new method iteratively selects a subset of both instances and features from the large-scale database. It would also

be interesting to investigate under what conditions (noise level and number of features and instances) it works better than other methods, if such a type of database is available.

Acknowledgements

The author would like to thank Naoki Abe of IBM for discussions related to the topics of this paper and anonymous reviewers for helpful comments.

References

1. Breiman, L.: Pasting Small Votes for Classification in Large Databases and On-line. *Machine Learning* **36** (1999) 85–103
2. Joachims, T. Making Large-scale SVM Learning Practical. In: Scholkopf, B., Burges, C., Smola, A. (eds.): *Advances in Kernel Methods - Support Vector Learning*, B. MIT Press, Cambridge (1999) 363, 366
3. Kohavi, R., John, G. H.: Wrappers for Feature Subset Selection. *Artificial Intelligence* **97** (1997) 273–324 362
4. Koller, D., Sahami, M.: Toward Optimal Feature Selection. In: Saitta, L. (eds.): *Proceedings of the Thirteenth International Conference on Machine Learning*. Morgan Kaufmann, Bari, Italy (1996) 284–292 362, 365
5. Kononenko, I., Hong, S. J.: Attribute Selection for Modelling. *Future Generation Computer Systems* **13** (1997) 181–195 362
6. Liu, H., Motoda, H.: *Feature Selection for Knowledge Discovery Data Mining*. Kluwer Academic Publishers, Boston (1998) 362
7. Mamitsuka, H., Abe, N.: Efficient Mining from Large Databases by Query Learning. In: Langley, P. (eds.): *Proceedings of the Seventeenth International Conference on Machine Learning*. Morgan Kaufmann, Stanford Univ., CA (2000) 575–582 364
8. Ng, A.: On Feature Selection: Learning with Exponentially Many Irrelevant Features as Training Examples. In: Shavlik, J. (eds.): *Proceedings of the Fifteenth International Conference on Machine Learning*. Morgan Kaufmann, Madison, WI (1998) 404–412 362
9. Provost, F., Kolluri, V.: A Survey of Methods for Scaling up Inductive Algorithms. *Knowledge Discovery and Data Mining* **3** (1999) 131–169 362
10. Quinlan, J. R.: *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Francisco (1993) 363, 366
11. Seung, H. S., Oppen, M., Sompolinsky, H.: Query by Committee. In: Haussler, D. (eds.): *Proceedings of the Fifth International Conference on Computational Learning Theory*. Morgan Kaufmann, NY (1992) 287–294 362
12. Xing, E. P., Jordan, M. I., Karp, R. M.: Feature Selection for High-dimensional Genomic Microarray Data In: Brodley, C. E., Danyluk, A. P. (eds.): *Proceedings of the Eighteenth International Conference on Machine Learning*. Morgan Kaufmann, Madison, WI (2001) 601–608

SVM Classification Using Sequences of Phonemes and Syllables

Gerhard Paaß¹, Edda Leopold¹, Martha Larson²,
Jörg Kindermann¹, and Stefan Eickeler²

¹ Fraunhofer Institute for Autonomous Intelligent Systems (AIS)
53754 St. Augustin, Germany

² Fraunhofer Institute for Media Communication (IMK)
53754 St. Augustin, Germany

Abstract. In this paper we use SVMs to classify spoken and written documents. We show that classification accuracy for written material is improved by the utilization of strings of sub-word units with dramatic gains for small topic categories. The classification of spoken documents for large categories using sub-word units is only slightly worse than for written material, with a larger drop for small topic categories. Finally it is possible, without loss, to train SVMs on syllables generated from written material and use them to classify audio documents. Our results confirm the strong promise that SVMs hold for robust audio document classification, and suggest that SVMs can compensate for speech recognition error to an extent that allows a significant degree of topic independence to be introduced into the system.

1 Introduction

Support Vector Machines (SVM) have proven to be fast and effective classifiers for text documents [6]. Since SVMs also have the advantage of being able to effectively exploit otherwise indiscernible regularities in high dimensional data, they represent an obvious candidate for spoken document classification, offering the potential to effectively circumvent the error-prone speech-to-text conversion. If optimizing spoken document classification performance is not entirely dependent on minimizing word error rate from the speech recognition component, room becomes available to adjust the interface between the speech recognizer and the document classifier.

We are interested in making the spoken document classification system as a whole speaker and topic independent. We present the results of experiments which applied SVMs to a real-life scenario, classifying radio documents from the program Kalenderblatt of the Deutsche Welle radio station. One striking result was that SVMs trained on written texts can be used to classify spoken documents.

2 SVM and Text Document Classification

Instead of restricting the number of features, support vector machines use a refined structure, which does not necessarily depend on the dimensionality of the input space. In the *bag-of-words-representation* the number of occurrences in a document is recorded for each word. A typical text corpus can contain more than 100,000 different words with each text document covering only a small fraction. Joachims [6] showed that SVMs classify text documents into topic categories with better performance than the currently best-performing conventional methods. Similar results were achieved by Dumais et al. [2] and Drucker et al. [1].

Previous experiments [9] have demonstrated that the choice of kernel for text document classification has a minimal effect on classifier performance, and that choosing the appropriate input text features is essential. We assume that this extends to spoken documents and chose basic kernels for these experiments, focusing on identifying appropriate input features. Recently a new family of kernel functions — the so called string kernels — has emerged in the SVM literature. They were independently introduced by Watkins [13] and Haussler [5]. In contrast to usual kernel functions these kernels do not merely calculate the inner product of two vectors in a feature space. They are instead defined on discrete structures like sequences of signs. String kernels have been applied successfully to problems in the field of bio-informatics [10] as well as to the classification of written text [11].

To facilitate classification with sub-word units one can generate n -grams which may take the role of words in conventional SVM text classification described above [Leo02][Joa98][Dum98]. Lodhi et al. [11] used subsequences of characters occurring in a text to represent them in a string kernel. The kernel is an inner product in the feature space consisting of all subsequences of length k , i.e. ordered sequences of k characters occurring in the text though not necessarily contiguously. The subsequences are weighted by an exponentially decaying factor of their full length in text, hence emphasizing those sequences which are close to contiguous. In contrast to our approach they use no classification dependent selection or weighting of features.

We use subsequences of linguistic units — phonemes, syllables or words — occurring in the text as inputs to a standard SVM. We only use contiguous sequences not exceeding a given length. Our approach is equivalent to a special case of the string kernel. Since the focus of this paper is on the representation of spoken documents we go beyond the original string kernel approach insofar as we investigate building strings from different basic units. We employ the word “ n -gram” to refer to sequences of linguistic units and reserve the expression “kernel” for traditional SVM-kernels. The kernels that we use in the subsequent experiments are the linear kernel, the polynomial of degree 2 and the Gaussian RBF-kernel. Our experiments consist of 1-of- n classification tasks. Each document is classified into the class which yields the highest SVM-score.

3 Sub-word Unit Speech Recognition

Continuous speech recognition systems (CSR) integrates two separately trained models each capturing a different level of language regularities. The *acoustic model* generates phoneme hypotheses from the acoustic signal whereas the *language model* constrains phoneme sequences admissible in the language. Errors made by a CSR can be roughly attributed to one or the other of these models. The acoustic models are responsible for errors occurring when phonemes in the input audio deviate in pronunciation from those present in the training data or when other unexpected acoustics, such as coughing or background noise from traffic or music intervene. The language model is responsible for errors due to words occurring in the audio input that either were missing or inappropriately distributed in the training data. Missing words are called OOV (Out of Vocabulary) and are a source of error even if the language model includes a 100,000 word vocabulary. A language model which is based on sub-word units like syllables rather than on words helps eliminate OOV error and makes possible independence from domain specific vocabularies. A syllable based language model, however, introduces extra noise on the syllable level because of errors due to combinations that would not have been part of the search space in a CSR with a word based language model. So there is a trade off between generality and accuracy of a CSR.

4 Combining SVMs and Speech Recognition

The representation of documents blends two worlds. From the linguistic point of view, texts consist of words which bear individual meaning and combine to form larger structures. From an algorithmic point of view, a text is a series of features which can be modeled by making certain approximations concerning their dependencies and assuming an underlying statistical distribution.

When it comes to the classification of *spoken documents*, the question of the appropriate text features becomes difficult, because the interplay between the speech processing system and the classification algorithm has to be considered as well. Our assumption is that it is desirable to give the SVM fine-grained classes of linguistic elements, and have it learn generalizations over them, rather than try to guess which larger linguistic classes might carry the information which would best assist the classifier in its decision.

Large vocabulary CSR systems were originally developed to perform pure transcription and they were optimized to output word for word the speech of the user. Under such a scenario, a substitution of the word 'an' for 'and' — a very difficult discrimination for the recognizer — would be counted as an error. Under a spoken document classification scenario, the effects of the substitution would be undetectable. If recognizer output is to be optimized for spoken document classification instead of transcription, non-orthographic units become an interesting alternative to words.

The potential of sub-word units to enhance the domain independence of a spoken document retrieval system is well documented in the literature. One of

the first systems to experiment with sub-word units, used vectors composed of sub-word phoneme sequences delimited by vowels. In this system the sub-word units, although shorter, perform marginally better than words. At the lowest sub-word level experimenters have used acoustic feature vectors and phonemes. In [4] N-gram topic models are built using such features, and incoming speech documents are classified as to which topic model most likely generated them. A concise overview of the literature on sub-words in speech document retrieval is given in [8].

For spoken document classification we decided that syllables and phoneme strings provide the best potential as text features. Since SVMs are able to deal with high dimensional inputs, we are not obliged to limit the number of input features. The idea is that short-ranged features such as short phoneme strings or syllables, will allow SVM to exploit patterns in the recognition error and indirectly access underlying topic features. Long-ranged features such as longer phoneme strings and syllable bi- and tri-grams will allow the SVM access to features with a higher discriminative value, since they are long enough to be semantically very specific.

5 The Data

In order to evaluate a system for spoken document classification, a large audio document collection annotated with classes is required. It is also necessary to have a parallel text document collection consisting of literal transcriptions of all the audio documents. Classification of this text document collection provides a baseline for the spoken document system.

The Deutsche Welle *Kalenderblatt* data set consists of 952 radio programs and the parallel transcriptions from the Deutsche Welle *Kalenderblatt* web-page <http://www.kalenderblatt.de>. Although the transcriptions are not perfect, they are accurate enough to provide a text classification baseline for spoken document classification experiments. The transcriptions were decomposed into syllables for the syllable experiments and phonemes for the phoneme based experiments using the transcription module of the BOSSII system [7].

Each program is about 5 minutes long and contains 600 running words. The programs were written by about 200 different authors and are read by about 10 different radio reporters and are liberally interspersed with the voices of people interviewed. This diversity makes the Deutsche Welle *Kalenderblatt* an appealing resource since it represents a real world task. The challenge of processing these documents is further compounded by the fact that they are interspersed with interviews, music and other background sound effects.

In order to train and to evaluate the classifier we needed topic class annotations for all of the documents in the data set. We chose as our list of topics the International Press Telecommunications Council (IPTC) subject reference system. Annotating the data set with topic classes was not straightforward, since which topic class a given document belongs to is a matter of human opinion. The

Fig. 1. Agreement of human Annotators in classifying the Kalenderblatt Documents

DW Kalenderblatt data from year	top choice of both annotators the same	one annotator choosing top others less	complete disagreement between annotators
1999	67 %	22 %	11 %
2000	74 %	17 %	9 %
2001	70 %	10 %	20 %

agreement of the human annotators about the class of documents represents an upper bound for the performance of the SVM classification system (table 1).

6 Experiment: Design and Setup

In the standard bag-of-words approach texts are represented by their type-frequency-vectors. Here we examine the usefulness of type-frequency-vectors constructed from the following linguistic units: word-forms of the written text, syllables derived from written text using BOSSII, phonemes derived from written text using BOSSII, syllables obtained from spoken documents by CSR, and phonemes obtained from spoken documents by CSR.

To derive syllables and phonemes from *written text* we use the transcription module of the second version of the Bonn Open Source Synthesis System (BOSSII) developed by the Institut für Kommunikationsforschung und Phonetik of Bonn University to transform written German words into strings of phonemes that represent their pronunciations and their syllable decompositions. A more detailed description of this system is given in [7].

In order to obtain Phonemes and Syllables the *spoken documents* We used the simplest acoustic models — monophone models — which have been trained on a minimal amount of generic audio data and have not been adapted to any of the speakers in the corpus. Additionally, we train a simple bigram model as the language model for the speech recognition using data from a completely different domain. We use syllables as the basic unit for recognition. System tests showed that the syllable recognition accuracy rate hovers around 30% for this configuration. Phonemes of the spoken documents are drawn from the syllable transcripts by splitting the syllables into their component phonemes parts.

As there is a large number of possible n -grams in the text we used statistical test to eliminate unimportant ones. First we required that each term must occur at least twice in the corpus. In addition we check the hypothesis that there is a statistical relation between the document class and the occurrence of a term w_k . Let $f(w_k, y)$ denote the number of documents of class y containing term w_k and let N_1 and N_{-1} be the number of documents of class 1 or -1 respectively. Then we obtain the table

	number of documents where ...	
	class $y = 1$	class $y = -1$
w_k in document	$f(w_k, y = 1)$	$f(w_k, y = -1)$
w_k not in document	$N_1 - f(w_k, y = 1)$	$N_{-1} - f(w_k, y = -1)$

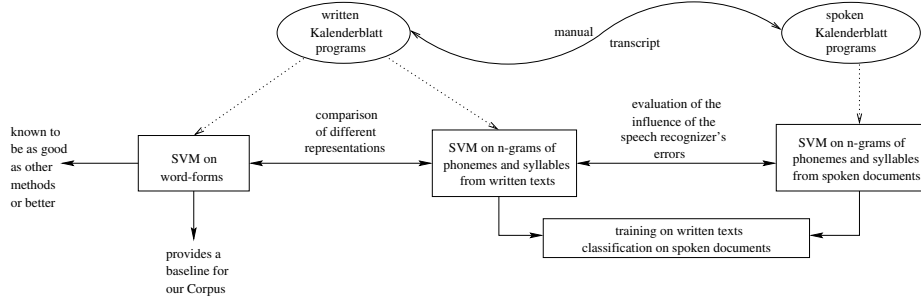


Fig. 2. The logical structure of our experimental design

If the rows and columns were independent then we would have $f(w_k, y = 1) = N * p(w_k) * p(y = 1)$ where $p(w_k)$ is the probability that w_k occurs in a document and $p(y = \pm 1)$ is the probability that $y = \pm 1$. We may check by a significance test if the first table originates from the distribution which obeys the independence assumption. We use a Bayesian version of the likelihood ratio test assuming a Dirichlet prior distribution. The procedure is discussed in [3]. The resulting test statistic is used to perform a preliminary selection of promising input terms to reduce the number of many thousand inputs. In the experiments different threshold values for the test statistic are evaluated.

We consider the task of deciding if a previously unknown text belongs to a given category or not. Let c_{targ} and e_{targ} respectively denote the number of correctly and incorrectly classified documents of the target category $y = 1$ and let e_{alt} and let c_{alt} be the same figures for the alternative class $y = -1$. We use the *precision* $prec = c_{targ} / c_{targ} + e_{alt}$ and the *recall* $rec = c_{targ} / c_{targ} + e_{targ}$ to describe the result of an experiment. In a specific situation a decision maker has to define a loss function and quantify the cost of misclassifying a target document as well as a document of the alternative class. The *F*-measure is a compromise between both cases [12]

$$F_{val} = \frac{2}{\frac{1}{prec} + \frac{1}{rec}}, \quad (1)$$

If recall is equal to precision then F_{val} is also equal to precision and recall.

7 Experiments with the Kalenderblatt Corpus

Our corpus poses a quite difficult categorization task. This can be seen from figure 1 where the discrepancies of the different human annotators are shown. If we assume that one annotator provides the “correct” classification then precision and recall of the other annotator is about 70%. As the final classification was defined by a human this is some upper limit of the possible accuracy that can be achieved. In our experiments we compare the properties of three different representational aspects and their effect on classification performance: (1)

Representation of a document by words. (2) Representation by simple terms or n -grams of terms, where ‘non-significant’ n -grams are eliminated. (3) Terms generated from the written representation or terms produced by CSR. As all representations are available for the same documents this allows to compare the relative merits of the representations. The setup is shown in figure 2.

We used five-fold cross-validation to get large enough training sets. As the F -value seems to be more stable because it is not affected by the tradeoff between recall and precision we use it as our main comparison figure. We utilized the *SVM^{light}* package developed by Joachims [6]. We performed experiments with respect to two topic categories: ‘politics’ of about 230 documents and ‘science’ with about 100 documents. This should give an impression of the possible range of results. Experiments with smaller categories led to unsatisfactory results. In preliminary experiments RBF-kernels turned out to be unstable with fluctuating results. We therefore concentrated on linear kernels.

7.1 Experiments with Written Material

We observed as a general tendency that precision increases and recall decreases with the size on n -grams. This can be explained by the fact, that longer linguistic sign-aggregates have a more specific meaning than shorter ones.

As can be seen in the upper part of table 1 topic classification using simple words starts with an F -value of 67.6% and 60.5% for ‘politics’ and ‘science’ respectively.

For both classes the syllables yield better results than words. For ‘politics’ syllables reach an F -value of 71.4% which is 3.8% better than the best word figure. There is a gain by using n -grams instead of single syllables which nevertheless reach an F -value of 70.1%. Longer n -grams ($n = 5, 6$) reduce accuracy. This can be explained by their low frequency of occurrence.

For the smaller category ‘science’ there is a dramatic performance increase to an $F_{val} = 73.1\%$ compared to an optimal 60.5% for words. Here n -grams perform at least 8.7% worse than simple terms, perhaps as they are more affected by the relatively large noise in estimating syllable counts. The good performance of syllables again may be explained by more stable estimates of their frequencies in each class. It is interesting that in the larger ‘politics’ class n -grams work better in contrast to the smaller ‘science’ class.

The best results are achieved for phonemes. For ‘politics’ there is no significant difference F -values compared to syllables, whereas for the small category ‘science’ there is again a marked increase to an F -value of 76.9% which is 3.8% larger than for syllables. The average length of German syllables is 4 to 5 phonemes, so phoneme trigrams in average are shorter and consequently more frequent than syllables. This explains the high F -value of phoneme trigram in the small category. Note that for both categories we get about the same accuracy which seems to be close to the possible upper limit as discussed above.

The effect of the significance threshold for n -gram selection can be demonstrated for bigrams, where the levels of 0.1 and 4 were used. The selection of

features according to their significance is able to support the SVMs capability to control model complexity independently of input dimension.

Table 1. Classification results on spoken and written material. Linear kernels and ten-fold cross-validation are applied

linguistic units	source	<i>n</i> -gram		politics			science		
		degree	thresh.	prec.	recall	F_{val}	prec.	recall	F_{val}
words	written	1	0.1	65.5	69.1	67.3	69.1	53.8	60.5
		1	4.0	66.1	69.1	67.6	71.6	55.8	62.7
		2	0.1	69.9	62.3	65.9	76.8	41.3	53.8
		2	4.0	69.5	63.2	66.2	85.2	44.2	58.2
		3	0.1	71.1	63.6	67.1	80.0	38.5	51.9
		3	4.0	71.5	60.5	65.5	84.9	43.3	57.3
syllables	written	1	0.1	63.0	80.5	70.7	70.5	76.0	73.1
		1	4.0	58.7	78.2	67.1	68.1	77.9	72.6
		2	0.1	69.4	72.3	70.8	78.1	54.8	64.4
		2	4.0	66.5	72.3	69.3	72.8	56.7	63.8
		3	0.1	71.2	70.9	71.1	78.7	46.2	58.2
		3	4.0	68.7	67.7	68.2	75.7	51.0	60.9
		4	0.1	71.9	70.9	71.4	80.0	46.2	58.5
		4	4.0	70.2	66.4	68.2	79.0	47.1	59.0
phonemes	written	5	4.0	71.1	65.0	67.9	79.3	44.2	56.8
		6	4.0	70.6	64.5	67.5	79.3	44.2	56.8
		2	0.1	55.2	84.5	66.8	59.5	90.4	71.8
		2	4.0	57.3	85.9	68.7	59.0	88.5	70.8
		3	0.1	60.6	79.5	68.8	72.8	72.1	72.5
		3	4.0	60.0	79.1	68.2	74.1	79.8	76.9
		4	0.1	65.9	76.4	70.7	81.2	66.3	73.0
		4	4.0	63.9	78.2	70.3	76.3	68.3	72.1
syllables	spoken	5	4.0	65.0	75.0	69.6	77.9	57.7	66.3
		6	4.0	68.6	73.6	71.1	80.6	51.9	63.2
		1	0.1	58.2	75.9	65.9	39.6	36.5	38.0
		1	4.0	57.6	75.5	65.4	40.2	45.2	42.5
		2	0.1	71.8	48.6	58.0	80.0	3.9	7.3
		2	4.0	69.0	52.7	59.8	60.0	5.8	10.5
		3	4.0	75.2	34.5	47.4	33.3	1.0	1.9
		4	4.0	76.5	29.5	42.6	33.3	1.0	1.9
phonemes	spoken	5	4.0	77.5	28.2	41.3	33.3	1.0	1.9
		6	4.0	77.5	28.2	41.3	33.3	1.0	1.9
		2	0.1	43.5	84.1	57.4	28.7	65.4	39.9
		2	4.0	47.9	79.5	59.8	30.2	59.6	40.1
		3	4.0	58.4	71.4	64.2	42.6	27.9	33.7
		4	4.0	64.8	61.8	63.3	63.2	11.5	19.5
		5	4.0	67.5	49.1	56.8	80.0	3.9	7.3
		6	4.0	73.4	41.4	52.9	50.0	1.0	1.9

Table 2. SVM classification of spoken documents when trained on written material. Only the topic category ‘politics’ is considered. Linear kernels are applied and ten-fold cross-validation is performed

linguistic units	n -gram		results for politics		
	degree	thresh.	prec.	recall	F_{val}
syllables	1	0.1	57.5	57.7	57.6
	1	4.0	55.6	54.1	54.8
	2	0.1	72.5	33.6	46.0
	2	4.0	64.5	53.6	58.6
	3	0.1	79.1	30.9	44.4
	3	4.0	71.2	42.7	53.4
	4	0.1	79.3	31.4	45.0
	4	4.0	74.3	38.2	50.5
phonemes	5	4.0	74.5	34.5	47.2
	6	4.0	74.5	33.2	45.9
	2	0.1	48.8	82.3	61.3
	2	4.0	55.5	69.1	61.5
	3	0.1	57.6	77.7	66.2
	3	4.0	59.5	74.1	66.0
	4	0.1	65.4	60.9	63.1
	4	4.0	59.8	69.5	64.3
	5	4.0	60.8	70.5	65.3
	6	4.0	62.2	67.3	64.6

7.2 Experiments with Spoken Documents

As discussed above the language model of the speech recognizer was trained on a text corpus which is different from the spoken documents to be recognized. Only 35% of the syllables produced by CSR were correct. With the experiments we can investigate if there are enough regularities left in the output of the CSR such that a classification by the SVM is possible. This also depends on the extent of systematic errors introduced by the CSR.

Again we performed experiments with respect to the two topic categories ‘politics’ and ‘science’. In the next section we evaluate the results for spoken documents and compare them to the results for written material. As before the SVM was trained on the output of the CSR and used to classify the documents in the test set. The results are shown in the lower part of table 1.

For ‘politics’ simple syllables have an F -value of 65.9%. This is only 5% worse than for the written material. The effect of errors introduced by the CSR is relatively low. There is a sharp performance drop for higher order n -grams with $n > 3$. A possible explanation is the fact that the language model of the CSR is based on bigrams of syllables.

For ‘science’ classifiers using syllables for spoken documents yield only an F -value of 42.5% and perform far worse than for written documents (73.1%).

Table 3. Optimal F -values for experiments discussed in this paper

topic category	data used for		optimal F -values		
	training	test	words	syllables	phonemes
‘politics’	written	written	67.6	71.4	71.1
‘politics’	spoken	spoken	—	65.9	64.2
‘politics’	written	spoken	—	58.6	66.2
‘science’	written	written	60.5	73.1	76.9
‘science’	spoken	spoken	—	42.5	40.1

Probably the errors introduced by CSR together with the small size of the class lead to this result.

Surprisingly phonemes yield for the topic category ‘politics’ on spoken documents an F -value of 64.2% which is nearly as good as the results for syllables. This result is achieved for 3-grams. For the small category ‘science’ phonemes yield 40.1% which is about 1.5% worse than the result for syllables.

7.3 Classification of Spoken Documents with Models Trained on Written Material

To get insight into the regularities of the errors of the speech recognizer we trained the SVM on synthetic syllables and phonemes generated for the written documents by BOSSII and applied these models to the output of the CSR.

The results for this experiment are shown in table 2. Whereas models trained on phonemes arrive at an F -value of 45.0% the syllables get up to 63.4%. This is nearly as much as the maximum F -value of 65.9% resulting from a model directly trained on the CSR output. This means that — at least in this setting — topic classification models may be trained without loss on synthetically generated syllables instead of genuine syllables obtained from a CSR.

We suppose that in spite of the low recognition rate of the speech recognizer the spoken and written dataset correspond to each other in terms of those syllables which consist the most important features for the classification procedure. One may argue, that those syllables are pronounced more distinctively which makes them better recognizable.

8 Discussion and Conclusions

The main results of this paper are summarized in table 3.

- On written text the utilization of n -grams of sub-word units like syllables and phonemes improve the classification performance compared to the use of words. The improvement is dramatic for small document classes.
- If the output of a continuous speech recognition system (CSR) is used for training and testing there is a drop of performance, which is relatively small

for larger classes and substantial for small classes. On the basis of syllable n -grams the SVM can compensate errors of a low-performance speech recognizer.

- In our setup it is possible to train syllable classifiers on written material and apply them to spoken documents. This is important since written material is far easier to obtain in larger quantities than annotated spoken documents.

An interesting result is that a spoken document classifier can be trained on written texts. This means that no *spoken* documents are needed for training a spoken document classifier. One can instead rely on *written* documents which are much easier to obtain.

The advantage of using for example syllables instead of words as input for the classification algorithm is that the syllables occurring in a given language can be well-represented by a finite inventory, typically containing several thousand forms. The inventory of words, in contrast, is infinite due to the productivity of word formation processes (derivation, borrowing, and coinage).

The results were obtained using a CSR with a simple speaker-independent acoustic model and a domain-independent statistical language model of syllable bigrams, insuring that recognizer performance is not specific to the experimental domain. Both models were trained on a different corpus which shows that the CSR may be applied to new corpora without the need to retrain. Syllables help circumvent the need for a domain-specific vocabularies and allows transfer to new domains to occur virtually unhindered by OOV considerations. The syllable model serves to control the complexity of the system, by keeping the inventory of text features to a bare minimum.

It is difficult to judge the significance of results. Since the tables demonstrate the F -value shows a stable behavior for different experimental setups, we think that the tendencies we have discovered are substantial. Future experiments will seek to further substantiate our results by evaluating topic categories additional to the two focused as well as investigating different kernels.

Acknowledgment

This study is part of the project Pi-AVida which is funded by the German ministry for research and technology (BMFT) (proj. nr. 107). We thank the Institute for Communication and Phonetics of the University of Bonn for contributing the BOSSII system and we thank Thorsten Joachims (Cornell University) who provided the SVM-implementation *SVM^{light}*.

References

1. Drucker, H., Wu, D., Vapnik, V. Support vector machines for spam categorization. *IEEE Transactions on Neural Networks*, 10 (5): 10048-1054, 1999. 374
2. Dumais, S., Platt, J., Heckerman, D., Sahami, M. (1998): Inductive learning algorithms and representations for text categorization. In: *7th International Conference on Information and Knowledge Management*, 1998. 374

3. Gelman, A., Carlin J. B., Stern, H. S., Rubin, D. B.: Bayesian Data Analysis. Chapman, Hall, London, 1995. 378
4. Glavitsch, U., Schäuble, P. (1992): A System for Retrieving Speech Documents, SIGIR 1992. 376
5. Haussler, David (1999): Convolution Kernels on Discrete Structures, UCSL-CRL-99-10. 374
6. Joachims, T. (1998). Text categorization with support vector machines: learning with many relevant features. *Proc. ECML '98*, (pp. 137–142). 373, 374, 379
7. Klabbers, E., Stöber, K., Veldhuis, R. Wagner, P., Breuer, S.: Speech synthesis development made easy: The Bonn Open Synthesis System, EUROSPEECH 2001. 376, 377
8. Larson, M.: Sub-word-based language models for speech recognition: implications for spoken document retrieval, *Proc. Workshop on Language Modeling and IR*. Pittsburgh 2001. 376
9. Leopold, E., Kindermann, J.: Text Categorization with Support Vector Machines. How to Represent Texts in Input Space? *Machine Learning*, 46, 2002, 423–444. 374
10. Leslie, Christa, Eskin, Eleazar, Noble, William Stafford (2002): The Spectrum Kernel: A String Kernel SVM Protein Classification. To appear: Pacific Symposium on Biocomputing. 374
11. Lodhi, Huma, Shawe-Taylor, John, Cristianini, Nello & Watkins, Chris (2001) Text classification using kernels, *NIPS 2001*, pp. 563-569. MIT Press. 374
12. Manning, Christopher D., Schütze (2000): *Foundations of Statistical Natural Language Processing*, MIT Press. 378
13. Watkins, Chris (1998): *Dynamic alignment Kernels*. Technical report, Royal Holloway, University of London. CSD-TR-98-11. 374

A Novel Web Text Mining Method Using the Discrete Cosine Transform

Laurence A.F. Park, Marimuthu Palaniswami, and Kotagiri Ramamohanarao

ARC Special Research Centre for Ultra-Broadband Information Networks
Department of Electrical & Electronic Engineering
The University of Melbourne
Parkville, Victoria, Australia 3010
lapark@ee.mu.oz.au
<http://www.ee.mu.oz.au/cubin>

Abstract. Fourier Domain Scoring (FDS) has been shown to give a 60% improvement in precision over the existing vector space methods, but its index requires a large storage space. We propose a new Web text mining method using the discrete cosine transform (DCT) to extract useful information from text documents and to provide improved document ranking, without having to store excessive data. While the new method preserves the performance of the FDS method, it gives a 40% improvement in precision over the established text mining methods when using only 20% of the storage space required by FDS.

1 Introduction

Text mining has been one of the great challenges to the knowledge discovery community and since the introduction of the Web, its importance has sky rocketed. The easiest way to search on the Web is to supply a set of query terms related to the information you want to find. A search engine will then proceed and try to find information that is associated to the query terms given. To classify a text based document using current vector space similarity measures, a search engine will compare the number of times the query terms appear, these results are then weighted and a relevance score is given to that document. Zobel and Moffat [7] showed that the most precise weighting scheme of the vector space measures is the BD-ACI-BCA method. This works very well on the TREC data set (where the queries are about 80 terms long), but as we have seen by results given by Web search engines, counting the words is sometimes not enough.

In [3] we showed how to utilise the spatial information in a document using Fourier Domain Scoring (FDS) to obtain more precise results. FDS not only records the number of times each word appears in the document, but also the positions of the words into entities called word signals. The Fourier transform is then applied to these word signals to obtain magnitude and phase information. This extra information can then be used to compare against other words.

Experiments [4] have shown that FDS gives similar results to the BD-ACI-BCA for long queries (containing about 80 terms), and a vast improvement of 60% greater precision for short queries (containing 1 to 5 terms).

We have shown through experimentation that using the Fourier transform on the word signals gave excellent results. However, the problem is that it requires more disk space to store the index (containing the extra spatial information) relative to the vector space methods and it requires more calculations to build the index and score the documents. With the intent to study the impact of reducing storage cost, we experimented in [4] using only a few frequency components but found the results were of a poorer quality. Therefore we propose a new method of document scoring using another transform which will give similar results to the Fourier transform, but not require as much information to obtain them.

When examining the properties of transforms, it is useful to note that one that keeps appearing in a wide range of disciplines is the Discrete Cosine Transform (DCT). The DCT decomposes a signal into the sum of cosine waves of different frequencies. Ahmed *et al.* [1] had first proposed the DCT to approximate the Karhunen-Loève transform. They showed that the DCT could easily be calculated by using the fast Fourier transform and that it also gave a close approximation to the KLT, which is used for signal compression. Some areas which have taken advantage of the compression property of the DCT are image (used in JPEG compression) and video compression (used in MPEG compression).

This paper is organised as follows. Section 2 will give a brief introduction to the Karhunen-Loève Transform, and its properties. Section 3 will introduce the discrete cosine transform and show its association to the KLT. Section 4 will explain the methods used in the document ranking experiments. Section 5 will outline the experiments performed and discuss some results. Finally, section 6 contains the conclusion.

2 Karhunen-Loève Transform

The Karhunen-Loève transform [2] (KLT, also known as Principle Component Analysis) adjusts the basis of a random signal, in a way as to diagonalise its covariance matrix. This is an important transform in signal compression, since it is able produce a transformed signal in which every element is linearly independent of each other, and will also order the basis functions in terms of importance to allow for easy least squares estimations.

The KLT is of the form:

$$\tilde{y} = T\tilde{x} \quad (1)$$

where $\tilde{x}$ is the input vector, $T = [t_0 \ t_1 \ \dots \ t_{N-1}]^T$ is the transformation matrix containing the basis vectors t_n , and $\tilde{y}$ is the transformed vector. The basis vectors t_n are found by solving:

$$\text{cov}(X)t_n = \lambda_n t_n \quad (2)$$

where $\text{cov}(X)$ is the covariance matrix of the matrix X consisting of input vectors $\tilde{x}$ and λ_n is a constant. We can see that equation 2 is the eigenvalue problem. Therefore the basis vectors t_n are the eigenvectors of the covariance matrix of X .

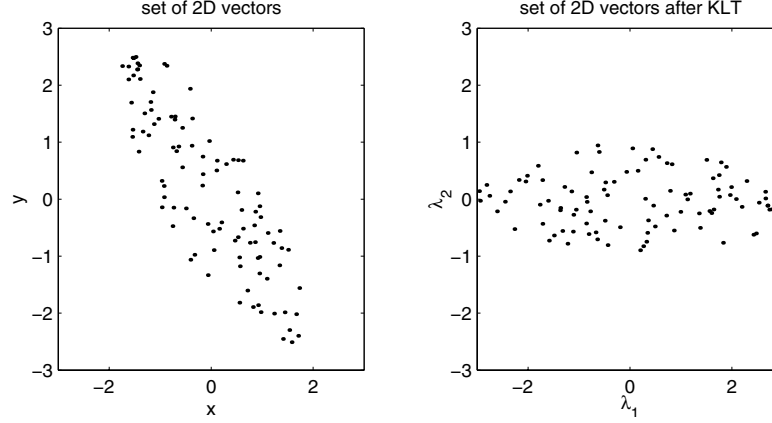


Fig. 1. Example of the Karhunen-Loève transform. The top plot displays 100 randomly generated points. The bottom plot shows the same points after performing the KLT

An example of the KLT can be seen in figure 1. We can see that the points have been mapped to a space where the x-axis is the dimension of greatest variance. The y-axis is the dimension of second greatest variance, which is also orthogonal to the x-axis. In this two dimensional case, the y axis is also the dimension of least variance.

To perform the KLT, we must take the whole data set into consideration. For large data sets, this can be computationally expensive. Many experiments have been performed to find the best estimate of the KLT which requires less calculations. It was shown in Ahmed *et al.* [1] that the DCT is a good approximation to the KLT for first order stationary Markov processes. A first order Markov process is defined as a random process $\{\dots, X_{n-2}, X_{n-1}, X_n, X_{n+1}, X_{n+2}, \dots\}$ such that:

$$\Pr(X_n = x_n | X_{n-1} = x_{n-1}, X_{n-2} = x_{n-2}, \dots) = \Pr(X_n = x_n | X_{n-1} = x_{n-1})$$

for each n . A stationary Markov process implies that the conditional probability $\Pr(X_n = x_n | X_{n-1} = x_{n-1})$ is independent of n . The signals we will be observing are word signals found in the FDS method. A weighted word signal $(\tilde{w}_{d,t})$ consists of the positions of term t in document d . Therefore, if we consider the weighted word count $w_{d,t,b}$ as a state and the bin position as the time (n), we can treat the word signal as a random process. Due to the nature of the English language, we will assume it is safe to identify a word signal as a first order stationary Markov process. The probability of term t appearing, after taking into account its previous appearances, should be similar to the probability when only taking into account its last appearance, independent of the bin position. Therefore by applying the DCT to a word signal, we are approximating the KLT of the word signal.

3 The Discrete Cosine Transform

The Discrete Cosine Transform (DCT), like the Fourier Transform, converts a sequence of samples to the frequency domain. But unlike the Fourier transform, the basis is made up of cosine waves, therefore each basis wave has a set phase. The DCT is of the form:

$$\tilde{X} = \text{DCT}(\tilde{x}) \quad X_k = \sum_{b=0}^{B-1} x_b \cos \frac{(2b+1)k\pi}{2B} \quad (3)$$

where $\tilde{X} = [X_0 \ X_1 \ \dots \ X_{B-1}]$ and $\tilde{x} = [x_0 \ x_1 \ \dots \ x_{B-1}]$. Therefore, a real positive signal (as in a word signal) maps to a real signal after the performing cosine transform. The DCT was introduced to solve the problems of pattern recognition and Wiener filtering [1].

To obtain significant features in a data set, a transform is usually applied, the features are selected and then the inverse transform is applied. The most significant features are the ones with the greatest variance. As shown, the KLT transforms a set of vectors, so that each component of the vector represents the direction of greatest variance in decreasing order. Therefore KLT is optimal for this task. We have seen that the DCT is a good approximation to the KLT for first order stationary Markov processes. Therefore the DCT should be a good choice for transforming to a space for easy feature selection (as in JPEG and MPEG).

4 Cosine Domain Scoring Method

In a recent paper on the FDS [4] method, we explained steps of the scoring method and proposed different ways to perform each step. The steps of performing the Cosine Domain Scoring (CDS) method on a document are:

1. Extract query term word signals $\tilde{f}_{d,t}$
2. Perform preweighting ($\tilde{f}_{d,t} \rightarrow \tilde{w}_{d,t}$)
3. Perform DCT ($\tilde{\eta}_{d,t} = \text{DCT}(\tilde{w}_{d,t})$)
4. Combine word spectrums ($\tilde{\eta}_{d,t} \rightarrow \tilde{s}_d$)
5. Combine word spectrum components ($\tilde{s}_d \rightarrow s_d$)

In this section we will look into the steps which differ from the FDS method.

4.1 Prewighting

When querying using a vector space method, weights are always applied to the term counts from documents to emphasise the significance of a term. The TBF×IDF and PTF×IDF weighting schemes [4] are both variants of the TF×IDF [6] which have been adjusted to suit the use of word signals. These are defined as:

$$\text{TBF} : w_{d,t,b} = 1 + \log_e f_{d,t,b} \quad (4)$$

where $f_{d,t,b}$ and $w_{d,t,b}$ are the count and weight of term t in spatial bin b of document d respectively.

$$\text{PTF} : w_{d,t,b} = (1 + \log_e f_{d,t}) \left(\frac{f_{d,t,b}}{f_{d,t}} \right) \quad (5)$$

where $f_{d,t}$ is the count of term t in document d . The preweighting of CDS will consist of one of these two methods or a variant of the BD-ACI-BCA weighting. The variant takes into account the word signals by replacing $w_{d,t}$ with:

$$w_{d,t,b} = r_{d,t,b} = 1 + \log_e f_{d,t,b} \quad (6)$$

The same values of W_d and W_q are used.

4.2 Combination of Word Spectrums

Once the DCT has been performed, we must combine all of the query word spectrums into one. In this experiment, this was done in two ways. The first called magnitude, the second called magnitude×selective phase precision. The combined word spectrum is defined as:

$$\tilde{s}_d = [s_{d,0} \ s_{d,1} \ \dots \ s_{d,B-1}] \quad s_{d,b} = \Phi_{d,b} \sum_{t \in T} H_{d,t,b}$$

where B is the number of spatial bins chosen, $H_{d,t,b}$ is the magnitude of the b th frequency component of the t th query term in the d th document and $\Phi_{d,b}$ is the phase precision of the b th frequency component in the d th document. The magnitude and phase precision values are extracted from the frequency components in the following way:

$$\eta_{d,t,b} = H_{d,t,b} \exp(i\theta_{d,t,b}) \quad (7)$$

where $\eta_{d,t,b}$ is the b th frequency component of the t th query term in the d th document. The phase vector is defined as follows:

$$\phi_{d,t,b} = \frac{\eta_{d,t,b}}{|\eta_{d,t,b}|} = \exp(i\theta_{d,t,b})$$

The DCT does not produce complex values when applied to a real signal, so we can either ignore the phase or treat the sign of the component as the phase. If we ignore the phase, this implies that we let $\Phi_{d,b} = 1$ for all d and b , we call this method *magnitude*. In the case where we do not ignore the phase, $\eta_{d,t,b}$ is real and so $\theta_{d,t,b}$ must be of the form πn , where n is an integer. This implies that we will have only $\phi_{d,t,b} \in \{-1, 1\}$. The selective phase precision equation [4] can be simplified to:

$$\begin{aligned} \text{Selective phase precision} &:= \bar{\Phi}_{d,b} = \left| \frac{\sum_{t \in T: H_{d,t,b} \neq 0} \phi_{d,t,b}}{\#(T)} \right| \\ &= \left| \frac{\sum_{t \in T} \text{sgn}_0(\eta_{d,t,b})}{\#(T)} \right| \end{aligned} \quad (8)$$

where T is the set of query terms, $\#(T)$ is the cardinality of the set T , and

$$\text{sgn}_x(y) = \begin{cases} 1 & \text{if } y \geq 0 \\ x & \text{if } y = 0 \\ -1 & \text{if } y < 0 \end{cases}$$

4.3 Combination of Spectral Components

After combining the word spectrums into one single score spectrum, we are left with B elements to combine in some way to produce a score. If using the Fourier transform, only the first $B/2 + 1$ elements are used since the rest are the complex conjugate of these. If using the DCT, all elements need to be considered, there is no dependence on any of these elements. Methods that will be considered are:

- Sum all components
- Sum first b components

where $0 < b < B$. By summing all of the components we will be able to utilise all of the information obtained from the DCT of the word spectrums. The second method (Sum first b components) will be considered due to the closeness of the DCT to the KLT. When the KLT is performed on a signal, we are adjusting the basis of the signals space such that the dimensions are ordered in terms of importance. If we consider only the first b components, we will be making a least squares approximation of the spectral vector for b dimensions. Therefore by performing the DCT and taking the first b components, we should have a close approximation to the B dimensional vector in the b dimensional space.

5 Experiments

The experiments were split into three groups. The first consisted of a general comparison of CDS methods using the already classified TREC documents and queries. The second compared the best CDS method with FDS 3.4.1¹ [4] method using short queries to simulate the Web environment. The third examined the ability to reduce the dimension of the word signals after performing the DCT. All experiments used the AP-2 document set from TREC, which is a collection of news paper articles from the Associated Press in the year 1988. The number of bins per word signal was set to eight. Case folding, stop word removal and stemming were performed before the index was created. The “staggered” form of the cosine transform is used since this is the standard for data compression and processing [5]. Each experiment compares the CDS method with the existing FDS, and the current best vector space method BD-ACI-BCA. The experiments are explained in more detail in the following sections.

¹ FDS 3.4.1 uses TBF×IDF preweighting, DFT, selective phase precision and adds all components

Table 1. Methods performed in experiment A

Method	Weighting	Combine word spectrums	Combine spectral components
CDS 1.1	none	magnitude	add all components
CDS 2.1	TBF $\times$ IDF	magnitude	add all components
CDS 3.1	PTF $\times$ IDF	magnitude	add all components
CDS 4.1	BD-ACI-BCA	magnitude	add all components
CDS 1.2	none	magnitude $\times$ selective phase precision	add all components
CDS 2.2	TBF $\times$ IDF	magnitude $\times$ selective phase precision	add all components
CDS 3.2	PTF $\times$ IDF	magnitude $\times$ selective phase precision	add all components
CDS 4.2	BD-ACI-BCA	magnitude $\times$ selective phase precision	add all components

5.1 Experiment A : Method Selection

To get an idea of the performance of each of the DCT methods, we will use the standard queries and relevance lists supplied by TREC. The queries applied were those from the TREC-1,2 and 3 conferences (queries 51 to 200). Each query is on average 80 words long. In [4], we have seen that when queries with t terms are given, where $t \gg$ number of words per bin, then the performance of FDS will approach the performance of a vector space measure. Due to the similarity between the FDS and CDS methods, this can also be said for CDS.

Therefore, in this experiment, we are looking for a method which will give similar (or better) performance than the BD-ACI-BCA method *This experiment is not to simulate the environment of the Web, but to examine the relative performance of each of the CDS methods using a standard document set and queries.* The methods used are displayed in table 1. The results can be seen in table 2 and figure 2. We can see that the methods CDS 2.2 and 4.1 perform well relative to the other CDS methods. Method CDS 2.2 gives a precision close to the FDS 3.4.1 method and BD-ACI-BCA. This gives a good indication that the DCT can be used in place of the DFT.

5.2 Experiment B : Web Queries

To simulate the Web environment, we will perform experiments using short queries (containing 1 to 5 words). The short queries were created by taking the title of the TREC queries used in experiment A. Due to this shortening of each query, the specifics of each query was also relaxed. Therefore the document relevance lists had to be recompiled. To create the new relevance lists, the top twenty documents classified by each method were collected and judged relative to each query. Only the top twenty were observed to emulate the level of patience of the typical Web search engine user. The methods compared are the CDS 2.2 (considered the best method from experiment A for low levels of recall), FDS 3.4.1 and BD-ACI-BCA. The results can be viewed in figure 3 and table 3. We can see that both FDS methods produce very similar results and show a 60%

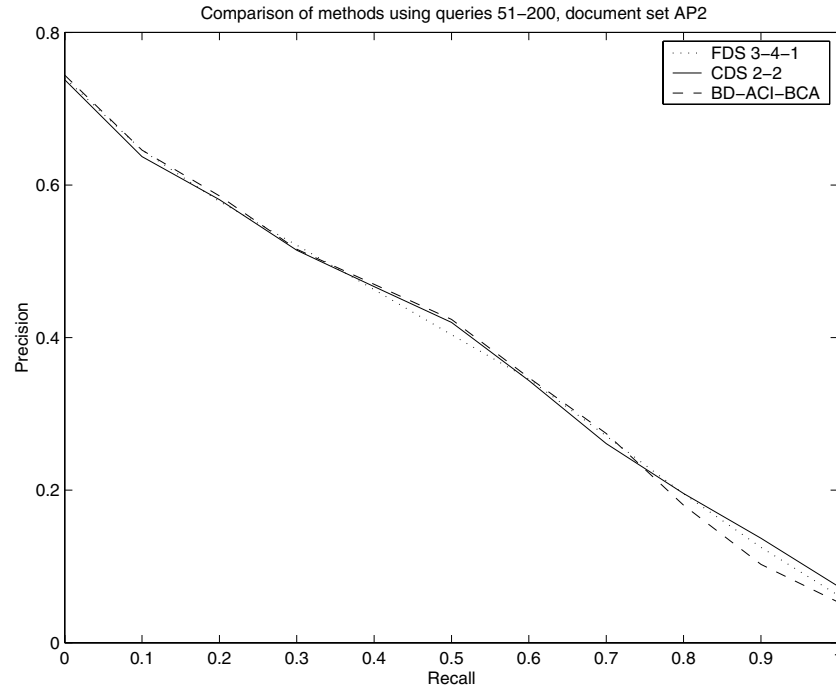


Fig. 2. Precision-recall plot for CDS 2.2, FDS 3.4.1 and BD-ACI-BCA using long query form of queries 51 to 200 and document set AP2

Table 2. Comparison of CDS methods using data set AP2 with the long form of queries 51 to 200. The largest CDS values per column are shown in *italics*

Method	Precision at Recall					Average Precision	R-Precision
	0%	10%	20%	30%	40%		
BD-ACI-BCA	0.7441	0.6458	0.5858	0.5159	0.4698	0.3792	0.4039
FDS 3.4.1	0.7404	0.6457	0.5783	0.5211	0.4628	0.3816	0.4015
CDS 1.1	0.6826	0.5631	0.4801	0.4049	0.3490	0.2817	0.3149
CDS 2.1	0.6889	0.5967	0.5383	0.4659	0.4252	0.3451	0.3603
CDS 3.1	0.6418	0.5320	0.4521	0.3797	0.3440	0.2859	0.3183
CDS 4.1	0.7326	<i>0.6462</i>	<i>0.5800</i>	0.5143	0.4511	0.3707	0.3938
CDS 1.2	0.6926	0.5772	0.5012	0.4331	0.3676	0.3047	0.3338
CDS 2.2	0.7343	0.6420	0.5767	<i>0.5228</i>	<i>0.4648</i>	<i>0.3808</i>	<i>0.4026</i>
CDS 3.2	0.7093	0.6031	0.5282	0.4569	0.4058	0.3428	0.3607
CDS 4.2	<i>0.7438</i>	0.6298	0.5672	0.5010	0.4419	0.3619	0.3804

improvement over BD-ACI-BCA. For some queries CDS 2.2 performs slightly better than FDS 3.4.1, for some it performs slightly less. From these results, we can see that we would get approximately the same results whether using CDS 2.2 or FDS 3.4.1.

Table 3. This table shows the short queries applied to the AP2 document set. We can see that the CDS and FDS methods give more relevant documents out of the top 20 returned by each method

Query term	Relevant documents in top 20		
	BD-ACI-BCA	CDS 2.2	FDS 3.4.1
Airbus Subsidies	10	14	14
Satellite Launch Contracts	7	13	12
Rail Strikes	8	18	17
Weather Related Fatalities	6	10	11
Information Retrieval Systems	3	6	7
Attempts to Revive the SALT II Treaty	8	14	12
Bank Failures	16	18	18
U.S. Army Acquisition of Advanced Weapons Systems	2	3	4
International Military Equipment Sales	6	10	11
Fiber Optics Equipment Manufacturers	5	8	8
Total	71	114	114

5.3 Experiment C : Reduction of Dimension

FDS requires $B+2$ elements to be stored per word signal ($\frac{B}{2} + 1$ elements for both magnitude and phase). CDS uses the DCT which produces real values, therefore only B elements need to be stored. This is still a large amount of data when we consider that the vector space methods only require that one element is to be stored. This is where the dimension reduction properties of the DCT are useful.

It is safe to assume that the CDS word signals are first order stationary Markov processes and hence the DCT is a good approximation of the KLT. Therefore, we should be able to perform a reduction of dimensionality and still obtain results comparable to those without the reduction. By performing the reduction, we do not have to store as much in the index and we do not have to perform as many calculations. Although the reduction may cause a degradation in the quality of results, it should be graceful due to the DCT approximation of the KLT. We performed experiments on the reduced data using both the long and short queries. The results can be seen in tables 4 and 5 respectively. We can see in both cases that the precision is reduced only by a small margin when the number of elements stored are reduced. Reducing the number of components has little effect on the precision of the top 20 documents for these ten short queries.

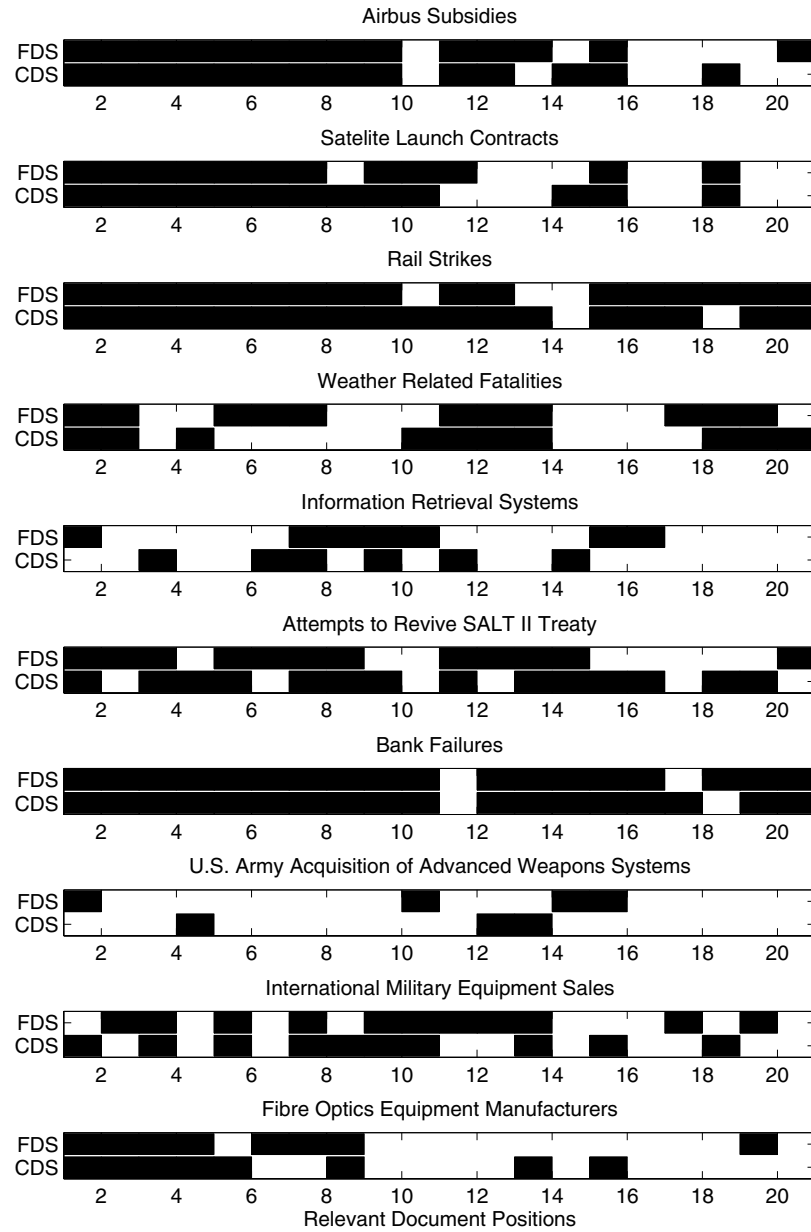


Fig. 3. This set of charts shows the positions of relevant documents from the queries in table 3. The documents are ranked with FDS 3.4.1 and CDS 2.2. A relevant document is identified by a black block. Both methods provide similar results

Table 4. Reduction of dimension results for long queries using CDS 2.2. The largest value in each column is shown in italics. The comp column refers to the number of components added to obtain the document score

Comp.	Precision at Recall					Average Precision	R-Precision
	0%	10%	20%	30%	40%		
1	0.7082	0.6204	0.5568	0.5002	0.4433	0.3492	0.3728
2	0.7139	0.6197	0.5570	0.5052	0.4443	0.3572	0.3770
3	0.7221	0.6201	0.5616	0.5103	0.4429	0.3625	0.3749
4	0.7222	0.6200	0.5580	0.5117	0.4516	0.3664	0.3807
5	0.7270	0.6239	0.5688	0.5176	0.4564	0.3743	0.3961
6	<i>0.7414</i>	0.6408	0.5698	<i>0.5258</i>	0.4557	0.3769	0.3955
7	0.7385	<i>0.6421</i>	0.5695	<i>0.5258</i>	0.4597	0.3788	0.3978
8	0.7343	0.6420	<i>0.5767</i>	0.5228	<i>0.4648</i>	<i>0.3808</i>	<i>0.4026</i>

In some cases we can see that by choosing a smaller number of components, we obtain a higher precision. If we use only 2 components (20% of size of FDS) we still obtain a 40% improvement in precision over BD-ACI-BCA for short queries.

Table 5. Short queries applied to the AP2 document set using the reduced dimension CDS 2.2 method. The column Dx refers to the CDS 2.2 method using the first x components

Query term	Number of Relevant documents in top 20							
	D1	D2	D3	D4	D5	D6	D7	D8
Airbus Subsidies	10	11	12	12	12	13	14	14
Satellite Launch Contracts	13	14	14	14	14	14	14	13
Rail Strikes	16	17	17	18	18	18	19	18
Weather Related Fatalities	10	10	10	9	9	8	9	10
Information Retrieval Systems	6	6	7	6	7	7	7	6
Attempts to Revive the SALT II Treaty	4	5	7	9	10	12	12	14
Bank Failures	19	19	17	19	19	17	18	18
U.S. Army Acquisition of Advanced Weapons Systems	1	3	4	4	4	4	4	3
International Military Equipment Sales	8	9	8	9	9	11	11	10
Fiber Optics Equipment Manufacturers	5	8	8	8	8	8	8	8
Total	92	102	104	108	110	112	116	114

6 Conclusion

We have introduced the new method called Cosine Domain Scoring (CDS) which uses the Discrete Cosine Transform to perform document ranking. Since each

word signal can be classified as a first order stationary Markov process, the results further illustrate the fact that the DCT is a close approximation to the Karhunen-Loève transform (KLT).

Results were given for three different experiments. The first experiment showed that CDS 2.2 produced the most precise results for long queries out of the CDS methods given. The second showed that using CDS resulted in comparable results to those of FDS. The third experiment displayed that by reducing the dimension of the transformed word signals, we not only reduce the number of calculations and space needed to store the index, but we also produce results with approximately the same precision. The experiment showed that if only 2 components were used, we obtain precision 40% higher than that of BD-ACI-BCA and require only 20% of the storage needed by FDS.

From these experiments, we have concluded that replacing the DFT with the DCT gives us similar results by only using a fraction of the components. The DCT's relationship to the KLT has allowed us to obtain a deeper understanding of the components produced by the transform. This allows us to give results just as good as FDS, requiring fewer calculations, and allowing us to store the index in a more compact manner.

Acknowledgements

We would like to thank the ARC Special Research Centre for Ultra-Broadband Information Networks for their support and funding of this research.

References

1. N. Ahmed, T. Natarajan, and K. R. Rao. Discrete cosine transform. *IEEE Transactions on Computers*, 23:90–93, January 1974. 386, 387, 388
2. Okan Ersoy. *Fourier-Related Transforms, Fast Algorithms and Applications*. Prentice-Hall, Upper Saddle River, NJ 07458, 1997. 386
3. Laurence A. F. Park, Marimuthu Palaniswami, and Ramamohanarao Kotagiri. Internet document filtering using fourier domain scoring. In Luc de Raedt and Arno Siebes, editors, *Principles of Data Mining and Knowledge Discovery*, number 2168 in Lecture Notes in Artificial Intelligence, pages 362–373. Springer-Verlag, September 2001. 385
4. Laurence A. F. Park, Kotagiri Ramamohanarao, and Marimuthu Palaniswami. Fourier domain scoring : A novel document ranking method. *IEEE Transactions on Knowledge and Data Engineering*, Submitted February 2002. http://www.ee.mu.oz.au/pgrad/lapark/fds_compare3.pdf. 385, 386, 388, 389, 390, 391
5. William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C, The art of scientific computing*. Cambridge University Press, 2nd edition, 1997. 390
6. Ian H. Witten, Alistair Moffat, and Timothy C. Bell. *Managing gigabytes : compressing and indexing documents and images*. Morgan Kaufmann Publishers, 1999. 388

7. Justin Zobel and Alistair Moffat. Exploring the similarity space. In *ACM SIGIR Forum*, volume 32, pages 18–34, Spring 1998. 385

A Scalable Constant-Memory Sampling Algorithm for Pattern Discovery in Large Databases

Tobias Scheffer¹ and Stefan Wrobel^{2,3}

¹ University of Magdeburg, FIN/IWS
P.O. Box 4120, 39016 Magdeburg, Germany
`scheffer@iws.cs.uni-magdeburg.de`

² FhG AiS, Schloß Birlinghoven
53754 Sankt Augustin, Germany

³ University of Bonn, Informatik III
Römerstr. 164, 53117 Bonn, Germany
`wrobel@ais.fhg.de`

Abstract. Many data mining tasks can be seen as an instance of the problem of finding the most interesting (according to some utility function) patterns in a large database. In recent years, significant progress has been achieved in scaling algorithms for this task to very large databases through the use of sequential sampling techniques. However, except for sampling-based greedy algorithms which cannot give absolute quality guarantees, the scalability of existing approaches to this problem is only with respect to the *data*, not with respect to the *size of the pattern space*: it is universally assumed that the entire hypothesis space fits in main memory. In this paper, we describe how this class of algorithms can be extended to hypothesis spaces that do not fit in memory while maintaining the algorithms' precise $\varepsilon - \delta$ quality guarantees. We present a constant memory algorithm for this task and prove that it possesses the required properties. In an empirical comparison, we compare variable memory and constant memory sampling.

1 Introduction

In many machine learning settings, an agent has to find a hypothesis which maximizes a given utility criterion. This criterion can be as simple as classification accuracy, or it can be a combination of generality and accuracy of, for instance, an association rule. The utility of a hypothesis can only be estimated based on data; it cannot be determined exactly (this would generally require processing very large, or even infinite amounts of data). Algorithms can still give stochastic guarantees on the optimality of the returned hypotheses, but guarantees that hold for *all possible* problems usually requires impractically large samples.

Past work on algorithms with stochastic guarantees has pursued two approaches — either processing a fixed amount of data and making the *guarantee* dependent on the observed empirical utility values (*e.g.*, [4,11]), or demanding

a certain fixed quality and making the number of *examples* dependent on the observed utility values [17,12,5,3] (this is often referred to as *sequential sampling*). The GSS algorithm [14,15] generalizes other sequential sampling algorithms by working for arbitrary utility functions that are to be maximized, as long as it is possible to estimate the utility with bounded error.

“General purpose” sampling algorithms like GSS suffer from the necessity of representing the hypothesis space explicitly in main memory. Clearly, this is only feasible for the smallest hypothesis spaces. In this paper, we present a sampling algorithm that has a constant memory usage, independently of the size of the hypothesis space that is to be searched.

The paper is organized as follows. In Section 2, we discuss related research. We define the problem setting in Section 3. For the reader’s convenience, we briefly recall the GSS sampling algorithm in Section 4 before we present our constant-memory algorithm in Section 5 and discuss the algorithm’s properties. We discuss experimental results on a purchase transactions database in Section 6; Section 7 concludes.

2 Prior Work

While many practical learning algorithms heuristically try to limit the risk of returning a sub-optimal hypothesis, it is clearly desirable to arrive at learning algorithms that can give precise guarantees about the quality of their solutions. If the learning algorithm is not allowed to look at any data before specifying the guarantee or fixing the required sample size (“data-independent”), we arrive at impractically large bounds as they arise, for instance, when applying PAC learning (*e.g.*, [6]) in a data-independent way. Researchers have therefore turned to algorithms that are allowed to look at (parts of) the data first.

We can then ask two questions. Knowing that our sample will be of size m , we can ask about the quality guarantee that results. On the other hand, knowing that we would like a particular quality guarantee, we can ask how large a sample we need to draw to ensure that guarantee. The former question has been addressed for predictive learning in work on self-bounding learning algorithms [4] and shell decomposition bounds [7,11].

For our purposes here, the latter question is more interesting. We assume that samples can be requested incrementally from an oracle (“incremental learning”). We can then dynamically adjust the required sample size based on the characteristics of the data that have already been seen; this idea has originally been referred to as sequential analysis [2,17]. Note that even when a (very large) database is given, it is useful to assume that examples are drawn incrementally from this database, potentially allowing termination before processing the entire database (referred to as *sampling* in KDD; [16]).

For predictive learning, the idea of sequential analysis has been developed into the Hoeffding race algorithm [12]. It processes examples incrementally, updates the utility values simultaneously, and outputs (or discards) hypotheses as soon as it becomes very likely that some hypothesis is near-optimal (or very

poor, respectively). The incremental greedy learning algorithm PALO [5] has been reported to require many times fewer examples than the worst-case bounds suggest.

In a KDD context, similar improvements have been achieved with the sequential algorithm of [3]. The GSS algorithm [14] sequentially samples large databases and maximizes arbitrary utility functions. For the special case of decision trees, the algorithm of [8] samples a database and finds a hypothesis that is very similar to the hypothesis that C4.5 would have found after looking at all available data.

3 Problem Setting

In many cases, it is more natural for a user to ask for the n best solutions instead of the single best or all hypotheses above a threshold. For instance, a user might find a small number of the most interesting patterns in a database, as is the case for association rule [1] or subgroup discovery [10,18].

We thus arrive at the following problem statement and quality guarantee.

Definition 1. (Approximate n -best hypotheses problem) *Let D be a distribution on instances, H a set of hypotheses, $f : H \rightarrow \mathbb{R}^{\geq 0}$ a function that assigns a utility value to each hypothesis and n a number of desired solutions. Then let δ , $0 < \delta \leq 1$, be a user-specified confidence, and $\varepsilon \in \mathbb{R}^+$ a user-specified maximal error. The approximate n -best hypotheses problem is to find a set $G \subseteq H$ of size n such that*

with confidence $1 - \delta$, there is no $h' \in H : h' \notin G$ and $f(h', D) > f_{\min} + \varepsilon$, where $f_{\min} := \min_{h \in G} f(h, D)$.

Previous sampling algorithms assume that all hypotheses can be represented explicitly in main memory along with the statistics of each hypothesis (e.g., [12,3,14,15]). Clearly, this is only possible for very small hypothesis spaces. In this paper, we only assume that there exists a generator function that enumerates all hypotheses in the hypothesis space. However, only a constant number of hypotheses and their statistics can be kept in main memory. Such generator functions exist for all practically relevant hypothesis spaces (it is easy to come up with an algorithm that generates all decision trees, or all association rules).

Most previous work has focused on the particular class of *instance-averaging* utility functions where the utility of a hypothesis h is the average of utilities defined locally for each instance. While prediction error clearly is an instance-averaging utility function, popular utility functions for other learning or discovery tasks often combine the generality of hypotheses with distributional properties in a way that cannot be expressed as average over the data records [10].

A popular example of such a discovery task is *subgroup discovery* [10]. Subgroups characterize subsets of database records within which the average value of the target attributes differs from the global average value, without actually conjecturing a value of that attribute. For instance, a subgroup might characterize a population which is particularly likely (or unlikely) to buy a certain

product. The generality of a subgroup is the fraction of all database records that belong to that subgroup. The term *statistical unusualness* refers to the difference between the default probability p_0 (the target attribute taking value one in the whole database) and the probability p of a target value of one within the subgroup. Usually, subgroups are desired to be both general (large g) and statistically unusual (large $|p - p_0|$). There are many possible utility functions [10] for subgroup discovery, none of which can be expressed as the average (over all instances) of an instance utility function.

Like [14], in order to avoid unduly restricting our algorithm, we will not make syntactic assumptions about f . In particular, we will not assume that f is based on averages of instance properties. Instead, we only assume that it is possible to determine a two-sided *confidence interval* f that bounds the possible difference between true utility and estimated utility (on a sample) with a certain confidence. Finding such confidence intervals is straightforward for classification accuracy, and is also possible for all but one of the popular utility functions from association rule and subgroup discovery [14].

Definition 2 (Utility confidence interval). *Let f be a utility function, let $h \in H$ be a hypotheses. Let $f(h)$ denote the true utility of h on the instance distribution D , $\hat{f}(h, Q_m)$ its estimated quality computed based on a sample Q_m of size m , drawn iid from the distribution D . Then $E : \mathcal{N} \times \mathbb{R} \rightarrow \mathbb{R}$ is a utility confidence bound for f iff for any δ , $0 < \delta < 1$,*

$$Pr_{Q_m} [|\hat{f}(h, Q_m) - f(h)| \leq E(m, \delta)] \geq 1 - \delta \quad (1)$$

We sometimes write the confidence interval for a specific hypothesis h as $E_h(m, \delta)$. Thus, we allow the confidence interval to depend on characteristics of h , such as the variance of one or more random variables that the utility of h depends on.

4 Sequential Sampling

In this section, we summarize the generalized sequential sampling algorithm of [14] for the reader's convenience.

The algorithm (Table 1), combines sequential sampling with the popular “loop reversal” technique found in many KDD algorithms. In step 3b, we collect data incrementally and apply these to all remaining hypotheses simultaneously (step 3c). This strategy allows the algorithm to be easily implemented on top of database systems (assuming they are capable of drawing samples), and enables us to terminate earlier.

After the statistics of each remaining hypothesis have been updated, the algorithm checks all remaining hypotheses and (step 3(e)i) outputs those where it can be sufficiently certain that the number of better hypotheses is no larger than the number of hypotheses still to be found (so they can all become solutions), or (Step 3(e)ii) discards those hypotheses where it can be sufficiently certain that the number of better other hypotheses is at least the number of hypotheses still

Table 1. Generic sequential sampling algorithm for the n -best hypotheses problem

Algorithm GSS. **Input:** n (number of desired hypotheses), H (hypothesis space), ε and δ (approximation and confidence parameters). **Output:** n approximately best hypotheses (with confidence $1 - \delta$).

1. **Let** $n_1 = n$ (the number of hypotheses that we still need to find) and **Let** $H_1 = H$ (the set of hypotheses that have, so far, neither been discarded nor accepted). **Let** $Q_0 = \emptyset$ (no sample drawn yet). **Let** $i = 1$ (loop counter).
 2. **Let** M be the smallest number such that $E(M, \frac{\delta}{2|H|}) \leq \frac{\varepsilon}{2}$.
 3. **Repeat until** $n_i = 0$ **Or** $|H_{i+1}| = n_i$ **Or** $E(i, \frac{\delta}{2|H_i|}) \leq \frac{\varepsilon}{2}$
 - (a) **Let** $H_{i+1} = H_i$.
 - (b) Query a random item of the database q_i .
 - (c) Update the empirical utility $\hat{f}$ of the hypotheses in the cache; update the sample size m_i .
 - (d) **Let** H_i^* be the n_i hypotheses from H_i which maximize the empirical utility $\hat{f}$.
 - (e) **For** $h \in H_i$ **While** $n_i > 0$ **And** $|H_i| > n_i$
 - i. **If** $h \in H_i^*$ (h appears good) **And** $\hat{f}(h, Q_i) \geq E_h(i, \frac{\delta}{2M|H_i|}) + \max_{h_k \in H_i \setminus H_i^*} \left\{ \hat{f}(h_k, Q_i) + E_{h_k}(i, \frac{\delta}{2M|H_i|}) \right\} - \varepsilon$ **Then Output** hypothesis h and then **Delete** h from H_{i+1} and **Let** $n_{i+1} = n_i - 1$. **Let** H_i^* be the new set of empirically best hypotheses.
 - ii. **Else If** $\hat{f}(h, Q_i) \leq \min_{h_k \in H_i^*} \left\{ \hat{f}(h_k, Q_i) - E_{h_k}(i, \frac{\delta}{2M|H_i|}) \right\} - E_h(i, \frac{\delta}{2M|H_i|})$ (h appears poor) **Then Delete** h from H_{i+1} . **Let** H_i^* be the new set of empirically best hypotheses.
 - (f) **Increment** i .
 4. **Output** the n_i hypotheses from H_i which have the highest empirical utility.
-

to be found (so it can be sure the current hypothesis does not need to be in the solutions). When the algorithm has gathered enough information to distinguish the good hypotheses that remain to be found from the bad ones with sufficient probability, it exits in step 3. Indeed it can be shown that this strategy leads to a total error probability less than δ as required [14].

In order to implement the algorithm for a given interestingness function a confidence bound $E(m, \delta)$ is required that satisfies Equation 1 for that specific f . In Table 2 we present a list of confidence intervals. We ask the reader to refer to [15] for a detailed treatment. All confidence intervals are based on normal approximation rather than the loose Chernoff or Hoeffding bound. z refers to the inverse normal distribution.

The simplest form of a utility function is the average, over all example queries, of some instance utility function $f_{inst}(h, q_i)$. The utility is then defined as $f(h) = \int f_{inst}(h, q_i) D(q_i) dq_i$ (the average over the instance distribution) and the estimated utility is $\hat{f}(h, Q_m) = \frac{1}{m} \sum_{i=1}^m f_{inst}(h, q_i)$ (average over the exam-

Table 2. Utility functions and the corresponding utility confidence bounds

$f(h)$	$E(m, \delta)$
instance-averaging	$E(m, \delta) = -\frac{z_{1-\frac{\delta}{2}}^\Lambda}{2\sqrt{m}}; \quad E_h(m, \delta) = -z_{1-\frac{\delta}{2}} s_h$
$\frac{g(p-p_0)}{g p-p_0 }$ $\frac{g\frac{1}{c}\sum_{i=1}^c p_i - p_{0_i} }{g\frac{1}{c}\sum_{i=1}^c p_i - p_{0_i} }$	$E(m, \delta) = \frac{z_{1-\frac{\delta}{4}}}{\sqrt{m}} + \frac{(z_{1-\frac{\delta}{4}})^2}{4m}$ $E_h(m, \delta) = z_{1-\frac{\delta}{4}}(s_g + s_p + z_{1-\frac{\delta}{4}} s_g s_p)$
$\frac{g^2(p-p_0)}{g^2 p-p_0 }$ $\frac{g^2\frac{1}{c}\sum_{i=1}^c p_i - p_{0_i} }{g^2\frac{1}{c}\sum_{i=1}^c p_i - p_{0_i} }$	$E(m, \delta) = \frac{3}{2\sqrt{m}}z_{1-\frac{\delta}{2}} + \frac{m+\sqrt{m}}{4m\sqrt{m}}(z_{1-\frac{\delta}{2}})^2 + \frac{1}{8m\sqrt{m}}(z_{1-\frac{\delta}{2}})^3$ $E_h(m, \delta) = (2s_g + s_p)z_{1-\frac{\delta}{2}} + (s_g^2 + 2s_g s_p)(z_{1-\frac{\delta}{2}})^2 + s_p s_g^2 (z_{1-\frac{\delta}{2}})^3$
$\frac{\sqrt{g}(p-p_0)}{\sqrt{g} p-p_0 }$ $\frac{\sqrt{g}\frac{1}{c}\sum_{i=1}^c p_i - p_{0_i} }{\sqrt{g}\frac{1}{c}\sum_{i=1}^c p_i - p_{0_i} }$	$E(m, \delta) = \sqrt{\frac{z_{1-\frac{\delta}{4}}}{2\sqrt{m}}} + \frac{z_{1-\frac{\delta}{4}}}{2\sqrt{m}} + \sqrt{\frac{z_{1-\frac{\delta}{4}}}{2\sqrt{m}} \frac{z_{1-\frac{\delta}{4}}}{2\sqrt{m}}}$ $E_h(m, \delta) = \sqrt{s_g z_{1-\frac{\delta}{4}}} + s_p z_{1-\frac{\delta}{4}} + \sqrt{s_g z_{1-\frac{\delta}{4}} s_p z_{1-\frac{\delta}{4}}}$

ple queries). An easy example of an instance-averaging utility is the classification accuracy.

In many KDD tasks, utility functions are common that weight the generality g of a subgroup and the deviation of the probability of a certain feature p from the default probability p_0 equally [13]. Hence, these functions multiply generality and distributional unusualness of subgroups. Another class of utility functions is derived from the binomial test heuristic [9].

5 Constant Memory Sampling

Algorithm GSS as described in the preceding section has empirically been shown to improve efficiency significantly, sometimes up to several orders of magnitude [14]. However, this works only as long as the hypothesis space H can be kept in main memory in its entirety. In this section, we will therefore now develop a constant-memory algorithm which removes this restriction, i.e., processes arbitrarily large hypothesis spaces in a fixed buffer size.

To this end, assume that we can allocate a constant amount of random-access memory large enough to hold b hypotheses along with their associated statistics. The idea of the algorithm is to iteratively load as many hypotheses into our buffer as will fit, and “compress” them into a set of n solution candidates using the generic sequential sampling algorithm GSS of [14]. We store these potential solutions (let us call these first-level candidates $C^{(1)}$) in our buffer, and iterate until we have processed all of H (the ideal case), or until $C^{(1)}$ fills so much of the buffer that less than n spaces are left for new hypotheses.

To gain new space, we now compress $C^{(1)}$ in turn into a set of n candidates using the GSS algorithm, adding these to the candidate set $C^{(2)}$ at the next higher level. $C^{(2)}$ of course is also stored in the buffer. Note that we may not always gain space when compressing at level d ($d = 1, \dots$), since the buffer may have been exhausted before $C^{(d)}$ has acquired more than n hypotheses. Thus, we repeat the compression upwards until we finally have gained space for at least

one more new hypotheses. We then load as many new hypotheses as will fit, and continue the process.

When H is finally exhausted, there will be sets of candidates $C^{(d)}$ at different levels d , so we need to do one final iterated round of compressions until we are finally left with n hypotheses at the topmost level which we can return.

Table 3. Algorithm LCM-GSS

Algorithm Layered Constant-Memory Sequential Sampling. **Input:** n (number of desired hypotheses), ε and δ (approximation and confidence parameters), $b > n$ size of hypothesis buffer. **Output:** n approximately best hypotheses (with confidence $1 - \delta$).

1. **Let** d_{max} the smallest number such that $cap(d_{max}, b, n) \geq |H|$
 2. **Let** $C^{(d)} := \emptyset$ for all $d \in \{1, \dots, d_{max}\}$
 3. **Let** $FreeMemory := b$
 4. **While** $H \neq \emptyset$
 - (a) **While** $FreeMemory \geq 0$
 - i. **Let** $B := GEN(FreeMemory, H)$.
 - ii. **Let** $C^{(1)} := C^{(1)} \cup GSS(n, \frac{\varepsilon}{d_{max}}, \delta(B), B)$.
 - iii. **Let** $FreeMemory := FreeMemory - \min(|B|, n)$
 - (b) **Let** $d := 1$
 - (c) **While** $FreeMemory = 0$
 - i. **If** $C^{(d)} \neq \emptyset$ **Then**
 - A. **Let** $C^{(d+1)} := C^{(d+1)} \cup GSS(n, \frac{\varepsilon}{d_{max}}, \delta(C^{(d)}), C^{(d)})$.
 - B. **Let** $C^{(d)} := \emptyset$
 - C. **Let** $FreeMemory := FreeMemory + |C^{(d)}| - n$
 - ii. **Let** $d := d + 1$
 5. **Let** $d := 1$
 6. **While** $\exists d' > d : C^{(d')} \neq \emptyset$
 - (a) **If** $C^{(d)} \neq \emptyset$ **Then**
 - i. **Let** $C^{(d+1)} := C^{(d+1)} \cup GSS(n, \frac{\varepsilon}{d_{max}}, \delta(C^{(d)}), C^{(d)})$.
 - (b) **Let** $d := d + 1$
 7. **Return** $GSS(n, \varepsilon_{i+1}, \delta_{i+1}, C^{(d)})$.
-

The algorithm is given in detail in Table 3. In writing up our algorithm, we assume that we are given a generator GEN which can provide us with a requested number of previously unseen hypotheses from H . Such a generator can easily be defined for most hypothesis spaces using a refinement operator and a proper indexing scheme.

As the main subroutine of LCM-GSS, we use the GSS algorithm described in the preceding section. Since we use this algorithm on selected subsets of H , we must make the available hypothesis space an explicit parameter as described in Table 1. Note also that the GSS algorithm, when used on the upper levels of

our algorithm, can keep the test statistics of each hypotheses acquired on lower levels, thus further reducing the need for additional samples.

In step 1, we determine the needed number of levels of compression based on the following lemma.

Lemma 1. *When using d_{max} levels, buffer size $b > n$, solution size n , algorithm LCM-GSS will process*

$$\min(|H|, \text{cap}(d_{max}, b, n))$$

hypotheses, where

$$\text{cap}(d, b, n) := (b - n \cdot \lfloor (\frac{b}{n}) \rfloor) + \sum_{i=1}^{\lfloor (\frac{b}{n}) \rfloor} \text{cap}(d-1, b, n)$$

and $\text{cap}(1, b, n) := b$

Proof. (Sketch) Let us first consider the simple case of an empty buffer of size b . If we want to use only a single layer of compression, all we can do is fill the buffer and compress using GSS, so we can handle $\text{cap}(1, b, n) := b$ hypotheses. When we allow a second level of compression, in the first iteration of step 4(a)i, we can load and compress b hypotheses. In the next iteration, we need to store the n candidates from the previous iteration, so we can load only $b - n$ new hypotheses. Since at each iteration, n additional candidates need to be stored, we can repeat at most $\lfloor (\frac{b}{n}) \rfloor$ times. We will then have filled $n \cdot \lfloor (\frac{b}{n}) \rfloor$ buffer elements. Since the remainder is smaller than n , we can simply fill the buffer with $b - n \cdot \lfloor (\frac{b}{n}) \rfloor$ additional elements, and then compress the entire buffer into a final solution (Step 7). Thus, in total, using two levels of compression the number of hypotheses we can handle is given in Equation 3

$$\text{cap}(2, b, n) := (b - n \cdot \lfloor (\frac{b}{n}) \rfloor) + \sum_{i=1}^{\lfloor (\frac{b}{n}) \rfloor} b - (i-1)n \quad (2)$$

$$= (b - n \cdot \lfloor (\frac{b}{n}) \rfloor) + \sum_{i=1}^{\lfloor (\frac{b}{n}) \rfloor} \text{cap}(1, n, b - (i-1)n) \quad (3)$$

A similar argument can be applied when d levels are being used. We first can run $d-1$ levels starting with an empty buffer which is finally reduced to n hypotheses, so we can then run another $d-1$ levels, but only in a buffer of size $b-n$, etc. Again we can repeat this at most $\lfloor (\frac{b}{n}) \rfloor$ times, and can then fill the remaining buffer space with less than n additional hypotheses. In general, the recursion given in the lemma results, where of course the total number of hypotheses processed will not be larger than $|H|$ due to step 4. $\diamond$

The following corollary justifies the restriction of the algorithm to buffer sizes that are larger than the number of desired solutions and shows that when this restriction is met, the algorithm is guaranteed to handle arbitrarily large hypothesis spaces.

Corollary 1. *As long as $b > n$, algorithm LCM-GSS can process arbitrarily large hypothesis spaces.*

Proof. Consider choosing $b := n + 1$. We then have $\lfloor (\frac{b}{n}) \rfloor = 1$, and thus

$$\text{cap}(1, b, n) = n + 1 \quad (4)$$

$$\text{cap}(2, b, n) = (b - n \cdot \lfloor (\frac{b}{n}) \rfloor) + \sum_{i=1}^{\lfloor (\frac{b}{n}) \rfloor} \text{cap}(1, b, n) \quad (5)$$

$$= (n + 1 - n) + n + 1 = n + 2 \quad (6)$$

and so on. $\diamond$

Perhaps it is instructive to illustrate how the algorithm will operate with $b = n + 1$. When first exiting the while loop (4a), $C^{(1)}$ will contain $n + 2$ elements, which will be reduced to n elements of $C^{(2)}$ by the while loop (4c). The next iteration of the while loop (4a) will simply add one hypothesis to $C^{(1)}$, which will finally be compressed into n hypotheses in $C^{(3)}$ by the while loop (4c). Thus, when using d levels in this way, we can process $n + d$ hypotheses.

Lemma 2. *Algorithm LCM-GSS never stores more than b hypotheses, and is thus a constant memory algorithm.*

We are now ready to state the central theorem that shows that our algorithm indeed delivers the desired results with the required confidence.

Theorem 1. *When using buffer size b , solution size n , as long as $b > n$, algorithm LCM-GSS will output a group G of exactly n hypotheses (assuming that $|H| > n$) such that, with confidence $1 - \delta$, no other hypothesis in H has a utility which is more than ε higher than the utility of any hypothesis that has been returned:*

$$\Pr[\exists h \in H \setminus G : f(h) > f_{\min} + \varepsilon] \leq \delta \quad (7)$$

where $f_{\min} = \min_{h' \in G} \{f(h')\}$; assuming that $|H| \geq n$.

Proof. We only sketch the proof here. Clearly, each individual compression using GSS is guaranteed to meet the guarantees based on the parameters given to GSS. When several layers are combined, unfortunately the maximal errors made in each compression layer sum up. Since we are using $d_{\max}$ layers, the total error is bounded by

$$\sum_{i=1}^{d_{\max}} \frac{\varepsilon}{d_{\max}} = d_{\max} \cdot \frac{\varepsilon}{d_{\max}} = \varepsilon$$

For confidence δ , we have to choose $\delta(S)$ for a hypothesis set S properly when running the algorithm. Now note that each hypothesis in H gets processed only once at the lowest level (when creating $C^{(1)}$). The capacity of this lowest level is $\text{cap}(d_{\max}, b, n)$. At the next level up, the winners of the first level get processed again, up to the top-most level, so the total number of hypotheses processed (many of them several times) is

$$M := \sum_{i=0}^{d_{\max}-1} \text{cap}(d_{\max} - i, b, n)$$

Thus, the union of all hypothesis sets ever compressed in the algorithm has at most this size M . Therefore, if we allocate $\delta(S) := \delta \cdot \frac{|S|}{M}$ we know that the sum of all the individual $\delta(S)$ will not exceed δ as required. $\diamond$

6 Experiments

In our experiments, we want to study the order of magnitude of examples which are required by our algorithm for realistic tasks. Furthermore, we want to measure how many additional examples the constant memory sampling algorithm needs, compared to an algorithm with unbounded memory usage.

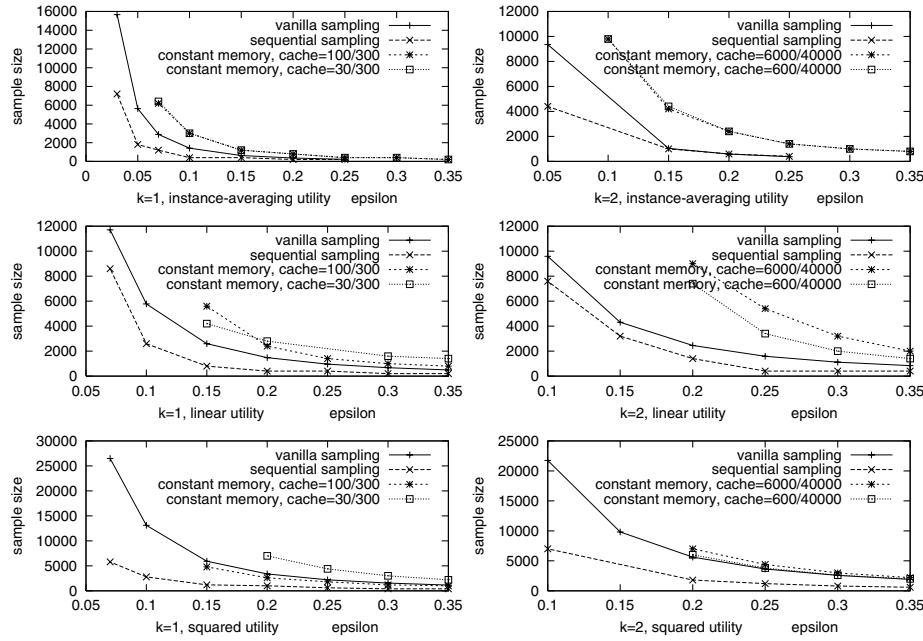


Fig. 1. Sample sizes for the juice purchases database

As baseline, we use a “vanilla” sampling algorithm that determines its sample bound without reference to the data. The vanilla algorithm determines the smallest number M that suffices to bound the utility of each hypothesis in the space with an error of up to $\epsilon/2$. Like our sequential algorithm, the vanilla sampling algorithm uses normal approximations instead of the loose Chernoff bounds.

We implemented a simple subgroup discovery algorithm. Hypotheses consist of conjunctions of up to k attribute value tests, continuous attributes are discretized in advance. The vanilla sampling algorithm determines a sample size M like our algorithm does in step 2, but using the full available error probability δ rather than only $\frac{\delta}{2}$. Hence, the non-sequential algorithm has a lower worst-case sample size than the sequential one but never exits or returns any hypothesis before that worst-case sample bound has been reached. Sequential and vanilla sampling algorithm use the same normal approximation and come with identical guarantees on the quality of the returned solution.

We used a database of 14,000 fruit juice purchase transactions. Each transaction is described by 29 attributes which specify properties of the purchased juice as well as customer attributes. The task is to identify groups of customers that differ from the overall average with respect to their preference for cans, recyclable bottles, or non-recyclable bottles. We studied hypothesis spaces of size 288 ($k = 1$, hypotheses test one attribute) and 37,717 ($k = 2$, conjunctions of two tests). We used the LCM-GSS algorithm with a cache equals to the hypothesis space, and with two decreasingly smaller cache sizes.

Since δ has only a minor (logarithmic) influence on the resulting sample size, all results presented in Figure 1 were obtained with $\delta = 0.1$. We varied the utility function; the target attribute has three possible values, so we used the utility functions $f_0 = \frac{1}{3} \sum_{i=1}^3 |p_i - p_{0_i}|$, $f_1 = g \frac{1}{3} \sum_{i=1}^3 |p_i - p_{0_i}|$, and $f_2 = g^2 \frac{1}{3} \sum_{i=1}^3 |p_i - p_{0_i}|$.

Figure 1 shows the sample size of the vanilla algorithm as well as the sample size required before the sequential algorithm returns the last (tenth) hypothesis and terminates. Figure 1 also shows the sample size required by LCM-GSS with two different cache sizes. In every single experiment, the sequential algorithm terminated earlier than the vanilla sampling algorithm; as ε becomes small, the relative benefit of sequential sampling can reach orders of magnitude.

When the cache is smaller than the hypothesis space, then the constant memory property has to be paid for by a larger sample size. The error constant ϵ is split up into the number of levels of the decision process. Based on the algorithm, our expectation was that LCM-GSS with two layers and error constant ϵ needs roughly as many examples as GSS with error constant $\frac{\epsilon}{2}$ when the cache is smaller than the hypothesis space but as least as large as the desired number of solutions times the number of caches needed. Our experiments confirm that, as a rule of thumb, LCM-GSS with error constant ϵ and GSS with error constant $\frac{\epsilon}{2}$ need similarly many examples.

7 Discussion

Sequential analysis is a very promising approach to reducing the sample size required to guarantee a high quality of the returned hypotheses. Sample sizes in the order of what the Chernoff and Hoeffding bounds suggest are only required when all hypotheses exhibit identical empirical utility values (in this case, identifying which one is really best is difficult). In all other cases, the single best, or the n best hypotheses can be identified much earlier.

The main contribution of this paper is a generalization of sequential analysis to arbitrarily large hypothesis spaces which we achieve by providing a fixed-memory sampling algorithm. We have to pay for the fixed-memory property by taking slightly larger sample sizes into account.

In machine learning, the small amount of available data is often the limiting factor. By contrast, in KDD the databases are typically so large that, when a machine learning algorithm is applied, computation time becomes critical.

Sampling algorithms like the GSS algorithm enable mining very (arbitrarily) large databases; the limiting factor is the main memory since all hypotheses and their statistics have to be represented explicitly. The LCM-GSS algorithm overcomes this limitation of sequential sampling and thereby enables mining very large databases with large hypothesis spaces. When we decrease the acceptable error threshold ϵ , then the computation time required to process the necessary sample size becomes the limiting factor again.

Acknowledgement

The research reported here was partially supported by Grant “Information Fusion / Active Learning” of the German Research Council (DFG), and was partially carried out when Stefan Wrobel was at the University of Magdeburg.

References

1. R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. Verkamo. Fast discovery of association rules. In *Advances in Knowledge Discovery and Data Mining*, 1996. 399
2. H. Dodge and H. Romig. A method of sampling inspection. *The Bell System Technical Journal*, 8:613–631, 1929. 398
3. C. Domingo, R. Gavelda, and O. Watanabe. Adaptive sampling methods for scaling up knowledge discovery algorithms. Technical Report TR-C131, Dept. de LSI, Politecnica de Catalunya, 1999. 398, 399
4. Y. Freund. Self-bounding learning algorithms. In *Proceedings of the International Workshop on Computational Learning Theory (COLT-98)*, 1998. 397, 398
5. Russell Greiner. PALO: A probabilistic hill-climbing algorithm. *Artificial Intelligence*, 83(1–2), July 1996. 398, 399
6. D. Haussler. Decision theoretic generalizations of the PAC model for neural net and other learning applications. *Information and Computation*, 100(1):78–150, 1992. 398
7. D. Haussler, M. Kearns, S. Seung, and N. Tishby. Rigorous learning curve bounds from statistical mechanics. *Machine Learning*, 25, 1996. 398
8. G. Hulten and P. Domingos. Mining high-speed data streams. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, 2000. 399
9. W. Klösgen. Problems in knowledge discovery in databases and their treatment in the statistics interpreter explora. *Journal of Intelligent Systems*, 7:649–673, 1992. 402
10. W. Klösgen. Explora: A multipattern and multistrategy discovery assistant. In *Advances in Knowledge Discovery and Data Mining*, pages 249–271. AAAI, 1996. 399, 400
11. J. Langford and D. McAllester. Computable shell decomposition bounds. In *Proceedings of the International Conference on Computational Learning Theory*, 2000. 397, 398
12. O. Maron and A. Moore. Hoeffding races: Accelerating model selection search for classification and function approximating. In *Advances in Neural Information Processing Systems*, pages 59–66, 1994. 398, 399

13. G. Piatetski-Shapiro. Discovery, analysis, and presentation of strong rules. In *Knowledge Discovery in Databases*, pages 229–248, 1991. 402
14. T. Scheffer and S. Wrobel. Incremental maximization of non-instance-averaging utility functions with applications to knowledge discovery problems. In *Proceedings of the International Conference on Machine Learning*, 2001. 398, 399, 400, 401, 402
15. T. Scheffer and S. Wrobel. Finding the most interesting patterns in a database quickly by using sequential sampling. *Journal of Machine Learning Research*, In Print. 398, 399, 401
16. H. Toivonen. Sampling large databases for association rules. In *Proc. VLDB Conference*, 1996. 398
17. A. Wald. *Sequential Analysis*. Wiley, 1947. 398
18. Stefan Wrobel. An algorithm for multi-relational discovery of subgroups. In *Proc. First European Symposium on Principles of Data Mining and Knowledge Discovery (PKDD-97)*, pages 78–87, Berlin, 1997. 399

Answering the Most Correlated N Association Rules Efficiently

Jun Sese and Shinichi Morishita

Department of Complexity Science and Engineering
Graduate School of Frontier Science, University of Tokyo
{sesejun,moris}@gi.k.u-tokyo.ac.jp

Abstract. Many algorithms have been proposed for computing association rules using the support-confidence framework. One drawback of this framework is its weakness in expressing the notion of correlation. We propose an efficient algorithm for mining association rules that uses statistical metrics to determine correlation. The simple application of conventional techniques developed for the support-confidence framework is not possible, since functions for correlation do not meet the anti-monotonicity property that is crucial to traditional methods. In this paper, we propose the heuristics for the vertical decomposition of a database, for pruning unproductive itemsets, and for traversing a set-enumeration tree of itemsets that is tailored to the calculation of the N most significant association rules, where N can be specified by the user. We experimentally compared the combination of these three techniques with the previous statistical approach. Our tests confirmed that the computational performance improves by several orders of magnitude.

1 Introduction

A great deal of research has examined the analysis of association rules [2]. Most of these studies have proposed efficient algorithms for computing association rules so that both support and confidence are sufficiently high. However, several researchers have remarked that one drawback of the support and confidence framework is its weakness in expressing the notion of correlation [6,1,10,11]. For instance, in practice, the analysis of scientific data calls for a method of discovering correlations among various phenomena, even when the database is noisy. The number of parameters taken into account can sometimes be in the millions. Such cases require an efficient way of selecting combinations of parameters (items) that are highly correlated with the phenomena of interest. For instance, in the human genome, the number of point-wise mutations in human DNA sequences is estimated to be in the millions. However, there is a need to discover combinations of mutations that are strongly correlated with common diseases, even in the presence of large amounts of noise. Such new applications demand fast algorithms for handling large datasets with millions of items, and for mining association rules that produce significant correlations.

Table 1. Transactions and Association Rules

a	b	c	d	e
1	1	1	1	1
1	1	0	1	1
1	0	1	0	1
1	0	0	0	1
0	1	1	0	1
0	1	0	0	1
0	0	1	1	1
0	0	0	1	1

(A) Examples of Transactions

$I \Rightarrow C$	support	confidence	correlated?
$\{x\} \Rightarrow \{y\}$	25%	50%	No
$(x, y \in \{a, b, c, d\}, x \neq y)$			
Many Rules with Singleton Sets			
$\{a, b\} \Rightarrow \{d\}$	25%	100%	Yes
$\{a\} \Rightarrow \{e\}$	50%	100%	No

(B) Examples of Association Rules

In the following, we present an example that illustrates the difference between correlation and the traditional criteria: support and confidence.

Motivating Example. In Table 1(A) taken from [11], each row except the first represents an itemset, and each column denotes an item. “1” indicates the presence of an item in the row, while “0” indicates the absence of an item. For example, the fourth row expresses $\{a, c, e\}$. Let I be an itemset, and let $Pr(I)$ denote the ratio of the number of transactions that include I to the number of all transactions. In our current example, $Pr(\{a, b\}) = 25\%$ and $Pr(\{a, b, c\}) = 12.5\%$. Association rules are in the form $I_1 \Rightarrow I_2$, where I_1 and I_2 are disjoint itemsets.

The *support* for rule $I_1 \Rightarrow I_2$ is the fraction of a transaction that contains both I_1 and I_2 , namely, $Pr(I_1 \cup I_2)$. The *confidence* of rule $I_1 \Rightarrow I_2$ is the fraction of a transaction containing I_1 that also contains I_2 , namely, $Pr(I_1 \cup I_2 | I_1)$. Table 1(B) shows some association rules derived from the database in Table 1(A). From Table 1(B), we may conclude that rule $\{a\} \Rightarrow \{e\}$ is the most valuable, since both its support and confidence are the highest.

Statistically speaking, the first and third rules do not make sense, because in each rule the assumptive itemset, say I , and the conclusive itemset, say C , are independent; that is, $Pr(I \cup C) = Pr(I) \times Pr(C)$. Conversely, in the second rule, the assumption and conclusion are highly and positively correlated.

This example suggests that the usefulness of association rules should be measured by the significance of the correlation between the assumption and the conclusion. For this purpose, the chi-squared value is typically used, because of its solid grounding in statistics. The larger the chi-squared value, the higher is the correlation.

Related Work. To address this problem, application of the Apriori algorithm has been investigated [6, 1, 11]. This algorithm proposed by Brin *et al.* [6] enumerates large itemsets first and then selects correlated itemsets based on the chi-squared measure. However, the algorithm does not discard the first non-correlated rule in Table 1(B) because of its use of the support threshold. To avoid the use of a support threshold, Aggarwal and Yu [1] proposed the genera-

tion of a *strongly collective itemset* that requires correlation among the items of any subset. However, the new algorithm discards the second rule in Table 1(B) because the algorithm might be too restrictive to output some desired rules.

To calculate the optimal solution of association rules using common statistical measures, Bayardo and Agrawal [5] presented a method of approaching optimal solutions by scanning what is called a support/confidence border. To further exploit Bayardo’s idea by combining it with the traversing itemset lattices developed by Agrawal and Srikant [3], we [11] proposed an algorithm called the AprioriSMP. The novel aspect of AprioriSMP is its ability to estimate a tight upper bound on the statistical metric associated with any superset of an itemset. AprioriSMP uses these upper bounds to prune unproductive supersets while traversing itemset lattices. In our example, if the chi-squared value is selected as the statistical metric, AprioriSMP prioritizes the second rule over the first according to their chi-squared values.

Several papers [13,7,18,4,9,16] have focused on methods to improve Apriori. These methods effectively use a specific property of the support-confidence framework, called the *anti-monotonicity* of the support function. For example, for any $I \subseteq J$, $Pr(I) \geq Pr(J)$, we are allowed to discard any superset of itemset I when $Pr(I)$ is strictly less than the given minimum threshold. By contrast, statistical metrics are not anti-monotonic. Therefore, whether we can improve the performance of AprioriSMP using the ideas of these traditional methods is not a trivial question.

Main Result. To overcome this difficulty, we propose a new algorithm, TidalSMP. Table 2 shows the differences between Apriori, AprioriSMP, and our TidalSMP algorithm. TidalSMP is the effective ensemble of three techniques: a vertical layout database to accelerate the database scan and calculate the statistical measure, null elimination to eliminate unproductive itemsets quickly, and set-enumeration tree (or SE-tree in short) traversal to generate candidate itemsets rapidly. Tests indicate that TidalSMP accelerates performance by several orders of magnitude over AprioriSMP, even for difficult cases such as rules with very low support. Furthermore, even in the presence of significant noise involving the most correlated rule, the execution time of the algorithm is independent of the amount of noise.

Table 2. Comparative table with TidalSMP

	Apriori	AprioriSMP	TidalSMP
Layout	Horizontal	Horizontal	Vertical
Traverse	Lattice	Lattice	Set-Enumeration tree
Pruning	Threshold	Upper-bound	Upper-bound and Null Elimination
Evaluation	Support	Statistical Measure	Statistical Measure

2 Preliminaries

TidalSMP frequently uses a method to estimate a tight upper bound of the statistical indices, such as the chi-squared values, entropy gain, or correlation coefficient. For the sake of simplicity in this paper, we present the chi-squared value method.

Chi-squared Values.

Definition 1 Let $I \Rightarrow C$ be an association rule, D be a set of transactions, and n be the number of transactions in D . Let O_{ij} where $(i, j) \in \{I, \bar{I}\} \times \{C, \bar{C}\}$ denotes the number of transactions that contain both i and j . Let O_i where $i \in \{I, \bar{I}, C, \bar{C}\}$ denote the number of transactions that contain i . For instance, $O_{I\bar{C}}$ represents the number of transactions that contain I , but do not include C . O_I represents the number of transactions that contain I , which is equal to the sum of the values in the row $O_{IC} + O_{I\bar{C}}$. Let x, y , and m denote O_I, O_{IC} and O_C , respectively. For each pair $(i, j) \in \{I, \bar{I}\} \times \{C, \bar{C}\}$, we calculate an expectation under the assumption of independence: $E_{ij} = n \times O_i/n \times O_j/n$. The chi-squared value expressed as $chi(x, y)$ is the normalized deviation of observation from expectation; namely,

$$chi(x, y) = \sum_{i \in \{I, \bar{I}\}, j \in \{C, \bar{C}\}} \frac{(O_{ij} - E_{ij})^2}{E_{ij}}.$$

In this definition, each O_{ij} must be non-negative; hence, $0 \leq y \leq x$, and $0 \leq m - y \leq n - x$. $chi(x, y)$ is defined for $0 < x < n$ and $0 < m < n$. We extend the domain of $chi(x, y)$ to include $(0, 0)$ and (n, m) , and define $chi(0, 0) = chi(n, m) = 0$. Incidentally, it is often helpful to explicitly state that x and y are determined by I , and we define $x = x(I)$ and $y = y(I)$.

Table 3. Notation Table for the Chi-squared Value

	C	$\bar{C}$	$\sum row$
I	$O_{IC} = y$	$O_{I\bar{C}}$	$O_I = x$
$\bar{I}$	$O_{\bar{I}C}$	$O_{\bar{I}\bar{C}}$	$O_{\bar{I}} = n - x$
$\sum column$	$O_C = m$	$O_{\bar{C}} = n - m$	n

Theorem 1 [11] For any $J \supseteq I$,

$$chi(x(J), y(J)) \leq \max\{chi(y(I), y(I)), chi(x(I) - y(I), 0)\}.$$

For any I , $0 = chi(0, 0) \leq chi(x(I), y(I))$.

Definition 2 $u(I) = \max\{chi(y(I), y(I)), chi(x(I) - y(I), 0)\}$.

Theorem 1 states that for any $J \supseteq I$, $\text{chi}(x(J), y(J))$ is bounded by $u(I)$. It is easy to see that $u(I)$ is tight in the sense that there could exist $J \supseteq I$ such that $\text{chi}(x(J), y(J)) = u(I)$.

Optimization Problem and AprioriSMP. Many practical applications generate a large number of association rules whose support value is more than the user-specified threshold. To resolve this problem, we use the chi-squared value and define an optimization problem that searches for the N most significant rules.

Optimization Problem: For a fixed conclusion C , compute the optimal association rules of the form $I \Rightarrow C$ that maximize the chi-squared value, or list the N most significant solutions.

This problem is NP-hard if we treat the maximum number of items in an itemset as a variable [11]. In real applications, however, the maximum number is usually bounded by a constant; hence, the problem is tractable from the viewpoint of computational complexity. AprioriSMP [11] is a feasible solution of the optimization problem. There is plenty of room to improve the performance of AprioriSMP because AprioriSMP inherits the itemset generation method of Apriori. To clarify the problem, we define $\mathcal{P}_k$ and $\mathcal{Q}_k$ as follows.

Definition 3 Let τ denote the temporarily N th best chi-squared value during the computation. An itemset is called a k -itemset if it contains k distinct items. We call an itemset I *promising* if $u(I) \geq \tau$ because a superset of I may provide a chi-squared value no less than τ . Let $\mathcal{P}_k$ denote the set of all promising k -itemsets. We use calligraphic letters to express collections of itemsets. We call an itemset I *potentially promising* if every proper subset of I is promising. Let $\mathcal{Q}_k$ denote the set of potentially promising k -itemsets.

We now focus on the N th best chi-squared value τ . In the k -itemset generation step, AprioriSMP generates all the potentially promising k -itemsets $\mathcal{Q}_k$, calculates their chi-squared values and τ , and then selects promising itemsets $\mathcal{P}_k$ whose chi-squared values are more than τ . This procedure requires us to calculate the chi-squared value of all the itemsets in $\mathcal{Q}_k$; hence, the cost of scanning transactions to calculate chi-squared values is high. Moreover, AprioriSMP calculates chi-squared values of all 1-itemsets in $\mathcal{Q}_1$.

3 Vertical Layout and Null Elimination

In order to reduce the cost of scanning transactions and reduce the size of $\mathcal{Q}_1$, we select a set of transactions in the form of a vertical layout because almost all the layouts used to find association rules depend on the anti-monotonicity of the evaluative function. Furthermore, some researchers [8,15,17] have recently cited the advantage of vertical layouts that maintain a set of transactions containing the item, for each item. In the following, we show the benefit of using vertical layouts to improve the performance of AprioriSMP.

Fig. 1(A) illustrates the bit-vector layout for a set of transactions in which each column denotes an item, while each row represents a transaction. We use letters of the alphabet to denote items, while numerals denote transactions. We decompose the bit-vector layout into vertical layout in Fig. 1(C) despite the horizontal layout in Fig. 1(B) introduced by Agrawal and Srikant.

Vertical Layout In order to express the conclusion for the optimization problem, let us select an item obj with special properties.

Definition 4 We call a fixed conclusion item the *objective item*. Let obj denote the objective item.

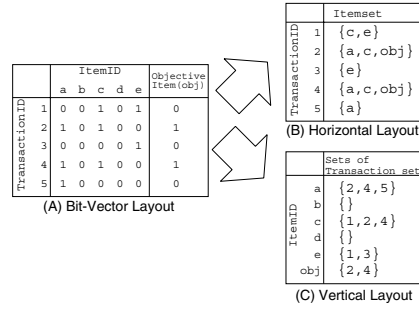


Fig. 1. Database layouts

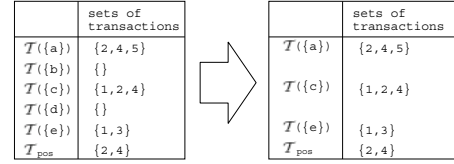


Fig. 2. Null elimination

Since the conclusion is fixed, let us focus on rules in the form $I \Rightarrow \{obj\}$. For itemset I , we need to calculate $chi(x(I), y(I))$. Since $x(I)$ ($y(I)$, respectively) is the number of transactions that contain I ($I \cup \{obj\}$), we need to develop an efficient way of listing transactions that contain I or $I \cup \{obj\}$. For this purpose, we introduce several terms.

Definition 5 Let $\mathcal{T}$ denote a set of transactions. We denote a set of transactions using the calligraphic letter $\mathcal{T}$. Let I be an itemset. Let $\mathcal{T}(I)$ denote $\{T | T \text{ is a transaction, and } I \subseteq T\}$, which is the set of transactions that contain I . Let obj be an objective item, and let T be a transaction. T is called *positive* (or *negative*) if $obj \in T$ ($obj \notin T$). $\mathcal{T}_{pos}$ and $\mathcal{T}_{neg}$ are then defined:

$$\begin{aligned} \mathcal{T}_{pos} &= \{T \in \mathcal{T} | T \text{ is a positive transaction.}\} \\ \mathcal{T}_{neg} &= \{T \in \mathcal{T} | T \text{ is a negative transaction.}\} \end{aligned}$$

Observe that $\mathcal{T}_{pos}(I)$ is exactly the set of transactions that contain $I \cup \{obj\}$. Consequently, we have $x(I) = |\mathcal{T}(I)|$, $y(I) = |\mathcal{T}_{pos}(I)|$.

In Fig. 1(C), let $\mathcal{T}$ be $\{1, 2, 3, 4, 5\}$. Then $\mathcal{T}(\{a\}) = \{2, 4, 5\}$, $\mathcal{T}_{pos}(\{a\}) = \{2, 4\}$, and $\mathcal{T}_{neg}(\{a\}) = \{5\}$.

Initializing the Vertical Layout and Null Elimination. Not using the support-confidence framework makes AprioriSMP generate all 1-itemsets. The

following observation, however, helps us to discard useless itemsets in Q_1 without eliminating productive itemsets.

Observation 1 (Null Elimination) Let I be an itemset such that $\mathcal{T}(I)$ is empty. $chi(x(I), y(I))$ is minimum; hence, it is safe to eliminate I from consideration.

Proof Let I be an itemset such that $\mathcal{T}(I)$ is the empty set. It is immediately apparent that $x(I) = |\mathcal{T}(I)| = 0$ and $y(I) = |\mathcal{T}_{pos}(I)| = 0$, which implies that $chi(x(I), y(I)) = chi(0, 0) = 0$. Since $chi(0, 0)$ is the minimum from Theorem 1, $chi(x(I), y(I))$ is no greater than $chi(x, y)$ for any x, y .

For instance, Fig. 2 shows how $\{b\}$ and $\{d\}$ are eliminated. In practice, this procedure is effective for reducing the size of the initial vertical layout. Incidentally, in Fig. 2, $\mathcal{T}_{pos}$ is displayed as a substitute for its equivalent value $\mathcal{T}(\{obj\})$.

Incremental Generation of Vertical Layouts. We now show how to generate vertical layouts incrementally without scanning all the transactions from scratch. To be more precise, given $\mathcal{T}(I_1)$ and $\mathcal{T}(I_2)$, which were computed in the previous steps, we develop an efficient way of computing $\mathcal{T}(I_1 \cup I_2)$ and $\mathcal{T}_{pos}(I_1 \cup I_2)$ from $\mathcal{T}(I_1)$ and $\mathcal{T}(I_2)$ instead of the whole transaction $\mathcal{T}$. The first step is the representation of $\mathcal{T}(I_i)$ as the union of $\mathcal{T}_{pos}(I_i)$ and $\mathcal{T}_{neg}(I_i)$. It is fairly straightforward to prove the following property.

Observation 2 Let I be an itemset. $\mathcal{T}(I) = \mathcal{T}_{pos}(I) \cup \mathcal{T}_{neg}(I)$, $\mathcal{T}_{pos}(I) = \mathcal{T}(I) \cap \mathcal{T}_{pos}$, and $\mathcal{T}_{neg}(I) = \mathcal{T}(I) - \mathcal{T}_{pos}$

For instance, in Fig. 3, when $\mathcal{T} = \{1, 2, 3, 4, 5\}$, $\mathcal{T}(\{a\}) = \{2, 4, 5\}$, $\mathcal{T}_{pos}(\{a\}) = \{2, 4\}$, and $\mathcal{T}_{neg}(\{a\}) = \{5\}$. The following observation is helpful for computing $\mathcal{T}_{pos}(I_1 \cup I_2)$ efficiently.

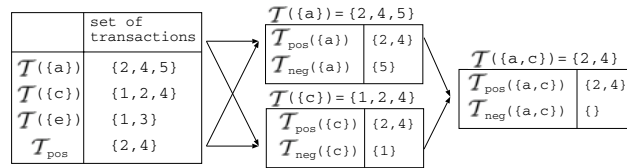


Fig. 3. Incremental Itemset Generation

Observation 3 Let I_1 and I_2 be itemsets.

$$\mathcal{T}_{pos}(I_1 \cup I_2) = \mathcal{T}_{pos}(I_1) \cap \mathcal{T}_{pos}(I_2) \text{ and } \mathcal{T}_{neg}(I_1 \cup I_2) = \mathcal{T}_{neg}(I_1) \cap \mathcal{T}_{neg}(I_2)$$

Combining Observations 2 and 3 allows us to calculate vertical layouts incrementally. Fig. 3 illustrates this process. We also note that the scanning cost of each k -itemset decreases as k increases during the computation because $|\mathcal{T}_{pos}(I_1 \cup I_2)| \leq \min\{|\mathcal{T}_{pos}(I_1)|, |\mathcal{T}_{pos}(I_2)|\}$ and $|\mathcal{T}_{neg}(I_1 \cup I_2)| \leq \min\{|\mathcal{T}_{neg}(I_1)|, |\mathcal{T}_{neg}(I_2)|\}$.

4 Set-Enumeration Tree Traversal

Let us consider whether an Apriori-like lattice traversal is useful for creating candidates according to statistical metrics. Note that there are many ways to generate one itemset by merging smaller itemsets; for instance, $\{a, b, c\}$ can be obtained by joining $\{a, b\}$ and $\{b, c\}$, or $\{a, b\}$ and $\{a, c\}$. Multiple choices for generating candidate itemsets may be detrimental to the overall performance when the number of long candidate itemsets becomes huge, since the cost of scanning itemsets could be enormous. Furthermore, scanning itemsets might be meaningless, because the N th best chi-squared value τ would be changed as the need arises and the itemsets might be unproductive when they were scanned. To settle the former problem, Bayardo [4] proposed using set-enumeration trees [14] in order to mine long-pattern association rules in the support-confidence framework. To resolve both problems, we present a method with a set-enumeration tree (SE-tree) tailored to the statistical-metric framework.

Set-Enumeration Tree. The SE-tree search framework is a systematic and complete tree expansion procedure for searching through the power set of a given set B . The idea is to first impose a total order on the elements of B . The root node of the tree will enumerate the empty set. The children of a node N will enumerate those sets that can be formed by appending a single element of B to N , with the restriction that this single element must follow every element already in N according to the total order. For example, a fully expanded SE-tree over a set of four elements, where each element of the set is denoted by its position in the ordering, appears in Fig. 4.

Fig. 4 illustrates an SE-tree. In the tree, beginning at the bottom root $\{\}$, there exists a unique path to each node. Thus, $\{a, b, c\}$ can be obtained by appending a , b , and c to $\{\}$. Conversely, in the complete lattice, there are multiple ways of generating $\{a, b, c\}$; for instance, joining $\{a, b\}$ and $\{b, c\}$ or $\{a, b\}$ and $\{a, c\}$. Overall, when considering the generation of candidate itemsets, SE-trees are simpler to use than complete lattices.

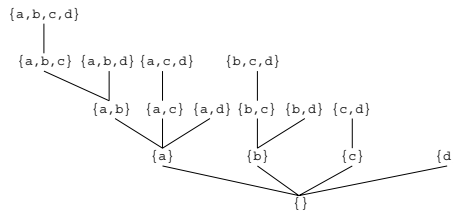


Fig. 4. SE-tree over $\{a, b, c, d\}$

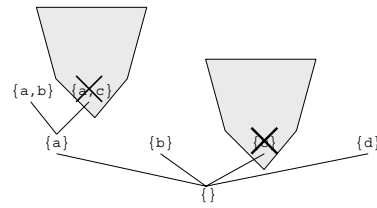


Fig. 5. Pruning branches

Pruning Branches.

Definition 6 Assume that all the items in an itemset are listed in a fixed total order. Let J be an itemset of m items. Let $\text{branch}(J)$ denote the set, $\{I \mid I \text{ be an}$

itemset of at least m items, and the set of the first m items in I be equal to J , which is called the *branch* rooted at J .

Two branches in Fig. 4 are:

$$\begin{aligned}\text{branch}(\{a, b\}) &= \{\{a, b\}, \{a, b, c\}, \{a, b, c, d\}, \{a, b, d\}\} \\ \text{branch}(\{b\}) &= \{\{b\}, \{b, c\}, \{b, d\}, \{b, c, d\}\}\end{aligned}$$

Note that $\text{branch}(J)$ does not include all the supersets of J . For example, $\{a, b\}$ is a superset of $\{b\}$, but is not a member of $\text{branch}(\{b\})$.

We now introduce a method of pruning the branches on an SE-tree using the upper bounds of statistical values. Theorem 1 leads us to the following observation.

Observation 4 If $\tau > u(J)$, for any I in $\text{branch}(J)$, $\tau > u(I)$.

This is because any I in $\text{branch}(J)$ is a superset of J . This property enables us to prune all the itemsets in $\text{branch}(J)$ at once. For example, in Fig. 5, when $\tau > u(\{c\})$, $\text{branch}(\{c\})$ can be pruned altogether. Furthermore, since $\tau > u(\{c\}) \geq u(\{a, c\})$, we can also eliminate $\text{branch}(\{a, c\})$.

Itemset Generation. We now describe how to generate new itemsets.

Definition 7 Assume that all the items are ordered. Let I be an itemset. Let $\text{head}(I)$ ($\text{tail}(I)$, respectively) denote the minimum (maximum) item of I , and let us call the item *head* (*tail*).

For example, when $I = \{b, c, d\}$, $\text{head}(I) = b$ and $\text{tail}(I) = d$.

SE-tree traversal generates a new set of $(k + 1)$ -itemsets from the set of k -itemsets $\mathcal{Q}_k$ and the set of 1-itemsets $\mathcal{B}_1$. It selects $Q \in \mathcal{Q}_k$ and $B \in \mathcal{B}_1$ such that $\text{tail}(Q) < \text{head}(B)$, and it generates a $(k + 1)$ -itemset by appending B to Q . For instance, let us consider the case when $\mathcal{Q}_2 = \{\{a, b\}\}$ and $\mathcal{B}_1 = \{\{a\}, \{b\}, \{d\}\}$. Of all possible pairs of $Q \in \mathcal{Q}_2$ and $B \in \mathcal{B}_1$, $Q = \{a, b\}$ and $B = \{d\}$ meet the condition that $\text{tail}(Q) < \text{head}(B)$. Hence, we put $\{a, b, d\}$, the appendage of B to Q , into $\mathcal{Q}_3$.

5 Pseudo-code for TidalSMP

We now provide pseudo-code descriptions of TidalSMP in Fig. 6-8. Let τ be the temporarily N th best chi-squared value. Fig. 6 presents the overall structure of TidalSMP. Fig. 7 shows pseudo-codes for vertical layout decomposition and null elimination, respectively. The program in Fig. 7 makes the decomposition database conform to the vertical layout and generates 1-itemset candidates following Observation 1. The code in Fig. 8 performs a level-wise generation of the SE-tree including the calculation of the intersection of two positive or negative itemsets according to Observation 3. Whenever a new itemset is created, its number of transactions, chi-squared value, and upper bound of the chi-squared value are calculated and stored to avoid duplicate computation.

TidalSMP

```

 $k := 1;$ 
 $(Q_1, L) = \text{TidalSMP-init};$ 
//  $L$ : list of top  $N$  chi-squared values
 $B_1 := Q_1; k++;$ 
repeat begin
     $(Q_{k+1}, B_1, L) :=$ 
        SE-tree Traversal( $Q_k, B_1, L$ );  $k++;$ 
end until  $Q_k = \phi;$ 
Return  $\tau$  with its
corresponding itemset;
```

Fig. 6. Pseudo-Code of TidalSMP

TidalSMP-init

```

 $L :=$  list of top  $N$  values in
     $\{chi(x(I), y(I)) | I \text{ is a 1-itemset}\},$ 
 $\tau :=$   $N$ th best in  $L;$ 
for each  $J \in \{I | I \text{ is a 1-itemset},$ 
     $T(I) \neq \phi, \tau \leq u(I)\}$ 
    // Null-Elimination
    Put  $J$  into  $Q_1;$ 
    Calculate  $T_{pos}(J)$  and  $T_{neg}(J);$ 
end
Return  $Q_1$  and  $L;$ 
```

Fig. 7. Pseudo-Code of TidalSMP-init

SE-tree Traversal(Set of k -itemsets $Q_k,$
 Set of 1-itemsets $B_1,$
 list of top N values L)

```

 $\tau :=$   $N$ th best in  $L;$ 
for each  $Q \in Q_k, B \in B_1$ 
    st.  $\text{tail}(Q) < \text{head}(B)$ 
    if  $u(B) < \tau;$  then
        Delete  $B$  from  $B_1;$ 
        // delete one branch
    else
         $T_{pos}(B \cup Q) := T_{pos}(B) \cap T_{pos}(Q);$ 
         $T_{neg}(B \cup Q) := T_{neg}(B) \cap T_{neg}(Q);$ 
        Put  $J$  into  $Q_{k+1}$ 
        if  $T(J) \neq \phi$  and  $u(J) \geq \tau;$ 
             $L :=$  list of top  $N$  values in
                 $L \cup \{chi(x(J), y(J))\};$ 
         $\tau :=$   $N$ th best in  $L;$ 
    end
end
Return  $Q_{k+1}, B_1, \tau;$ 
```

Fig. 8. Pseudo-Code of SE-tree Traversal

6 Experimental Results

We evaluated the overall performance of TidalSMP implemented in C++ with a Pentium-III 450-MHz processor and 768 MB of main memory on Linux. We generated a test dataset using the method introduced by Agrawal and Srikant [3]. To intentionally create an optimal association rule, we arbitrarily selected one maximal potentially large itemset, called X , and doubled the probability so that this itemset would be picked during generation of the test dataset. The other itemsets were selected according to the method in [3]. We then selected item c in X as the objective item, making $(X - \{c\}) \Rightarrow \{c\}$ the optimal association rule. The parameters and their default values used in the test dataset generator were as follows:

- |D|: Number of transactions
- |T|: Average size of transactions
- |I|: Average size of maximal and potentially large itemsets
- |M|: Number of maximal and potentially large itemsets
- W: Number of items

Figs. 9-12 present the experimental results. We used the default parameters $|D| = W = 10K$, $|T| = 20$, $|I| = 10$ and $|M| = |D|/10$ unless otherwise stated,

and we calculated the association rule with the maximum chi-squared value. Each figure shows the execution time of our algorithms, including both the time required to load the database from a hard disk and the time elapsed using our algorithms. The datasets in the secondary disk were originally stored in an Apriori-like layout. Those datasets were then loaded into the main memory in a vertical layout.

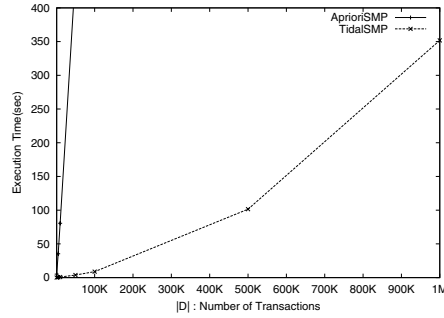


Fig. 9. Scalability of the performance (D≤1M.W50K)

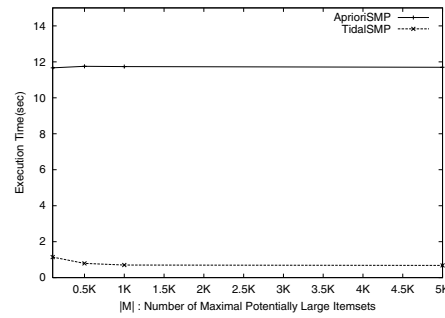


Fig. 10. The performance of low support (D10K.W10K.M≥100)

Scalability. Fig. 9 demonstrates the performance of TidalSMP when $|D|$ ranges from 1K to 1M. TidalSMP accelerates the performance by several orders of magnitude over AprioriSMP. For TidalSMP, the execution time increases quadratically in $|D|$ because the number of occurrences of items in vertical layouts is quadratic in $|D|$.

Rules with Low Support. We next investigated the performance of mining the optimal association rule $(X - \{c\}) \Rightarrow \{c\}$ with low support. Such a dataset can be obtained by increasing $|M|$. For instance, when $|M| = 5K$ and $|D| = 10K$, the average support of the optimal association rule is 0.04% because the probability that X is randomly selected is defined as $2/|M| (= 0.04\%)$. Similarly, if we set $|M|$ to 0.5K, the support is 0.4%. Fig. 10 shows that the execution time decreases when the support for the statistically optimal association rule also decreases. This might appear to contradict our expectations, but it really could happen because the implementation of the vertical layout is effective in this situation. Note that the lower the support for the optimal rule becomes, the smaller the size of each vertical layout.

Tolerance of Noise. The use of statistical values is expected to allow derivation of the most correlated rule even in the presence of many noisy transactions that are irrelevant to the optimal solution, since noise can naturally be ignored by statistical inference. In order to verify this conjecture, we performed two experiments.

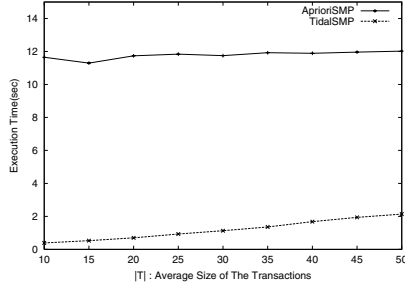


Fig. 11. Tolerance of noise ($T \leq 50, I = 10, D = 10K, W = 10K$)

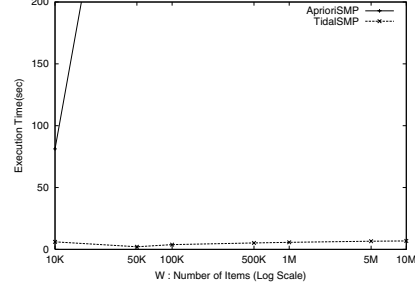


Fig. 12. Tolerance of noise by numerous items ($D = 50K, W \leq 10M$)

First, we intentionally supplied a large number of noisy transactions by increasing $|T|$ to 50 while setting $|I|$ to 10. Fig. 11 shows that the execution time increased moderately and was proportional to $|T|$.

Next, we considered the case in which we increased the number of items to ten million. Note that the x-axis W is in log scale in 12. Fig. 12 shows that the execution time is independent of the number of items, except when the number of items is less than 10 K. The result suggests that the addition of items irrelevant to the optimal association rule does not have an impact on the overall execution time. One reason for the performance improvement is the quick elimination of unproductive itemsets by null elimination and the dynamic change of the threshold τ .

Both experiments indicated that mining statistically optimal association rules tolerates the presence of noise in datasets.

7 Conclusion

We have presented the heuristics for the vertical decomposition of a database, for pruning unproductive itemsets, and for traversing the SE-tree of itemsets that are tailored to the calculation of association rules with significant statistical metrics. This combination of tree techniques accelerates the overall performance. Therefore, for an experimental database that contains more than 10 million items or a million transactions, our algorithm can efficiently calculate the optimal association rules of the database.

Finding the correlation between itemsets is applicable to various problems. We have been applying this technique to the analysis of correlation between multiple genotypes and the objective phenotype of interest [12].

References

1. C. C. Aggarwal and P. S. Yu. A new framework for itemset generation. In *Proc. of PODS'98*, pp. 18-24, June 1998. 410, 411

2. R. Agrawal, T. Imielinski, and A. N. Swami. Mining association rules between sets of items in large databases. In *Proc. of SIGMOD'93*, pp. 207-216, May 1993. 410
3. R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. of VLDB'94*, pp. 487-499, Sept. 1994. 412, 419
4. R. J. Bayardo Jr. Efficiently mining long patterns from databases. In *Proc. of SIGMOD'98*, pp. 85-93, June 1998. 412, 417
5. R. J. Bayardo and R. Agrawal Mining the most interesting rules. In *Proc. of SIGKDD'99*, pp. 145-153, Aug. 1999. 412
6. S. Brin, R. Motwani, and C. Silverstein. Beyond market baskets: Generalizing association rules to correlations. In *Proc. of SIGMOD'97*, pp. 265-276, May 1997. 410, 411
7. S. Brin, R. Motwani, J. D. Ullman, and S. Tsur. Dynamic itemset counting and implication rules for market basket analysis. In *Proc. of SIGMOD'97*, pp. 265-276, May 1997. 412
8. B. Dunkel and N. Soparkar. Data organization and access for efficient data mining. In *Proc. of ICDE'99*, pp. 522-529, March 1999. 414
9. J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *Proc. of SIGMOD'00*, pp. 1-12, May 2000. 412
10. B. Liu, W. Hsu, and Y. Ma. Pruning and summarizing the discovered associations. In *Proc. of SIGKDD'99*, pp. 125-134, 1999. 410
11. S. Morishita and J. Sese. Traversing lattice itemset with statistical metric pruning. In *Proc. of PODS'00*, pp. 226-236, May 2000. 410, 411, 412, 413, 414
12. A. Nakaya, H. Hishigaki, and S. Morishita. Mining the quantitative trait loci associated with oral glucose tolerance in the OLETF rat. In *Proc. of Pacific Symposium on Biocomputing*, pp. 367-379, Jan. 2000. 421
13. J. S. Park, M. S. Chen, and P. S. Yu. An effective hash-based algorithm for mining association rules. In *Proc. of SIGMOD'95*, pp. 175-186, May 1995. 412
14. R. Rymon. Search through systematic set enumeration. In *Proc. of KR'92*, pp. 539-550, 1992. 417
15. P. Shenoy, J. R. Haritsa, S. Sudarshan, G. Bhalotia, M. Bawa, and D. Shah. Turbocharging vertical mining of large databases. In *Proc. of SIGMOD'00*, pp. 22-33, May 2000. 414
16. G. I. Webb. Efficient search for association rules. In *Proc. of SIGKDD'00*, pp. 99-107, Aug. 2000. 412
17. M. J. Zaki. Generating non-redundant association rules. In *Proc. of SIGKDD'00*, pp. 34-43, Aug. 2000. 414
18. M. J. Zaki, S. Parthasarathy, M. Ogihara, and W. Li. New algorithms for fast discovery of association rules. In *Proc. of KDD'97*, pp. 343-374, Aug. 1997. 412

Mining Hierarchical Decision Rules from Clinical Databases Using Rough Sets and Medical Diagnostic Model

Shusaku Tsumoto

Department of Medicine Informatics, Shimane Medical University
School of Medicine
89-1 Enya-cho Izumo City, Shimane 693-8501 Japan
`tsumoto@computer.org`

Abstract. One of the most important problems on rule induction methods is that they cannot extract rules, which plausibly represent experts' decision processes. On one hand, rule induction methods induce probabilistic rules, the description length of which is too short, compared with the experts' rules. On the other hand, construction of Bayesian networks generates too lengthy rules. In this paper, the characteristics of experts' rules are closely examined and a new approach to extract plausible rules is introduced, which consists of the following three procedures. First, the characterization of decision attributes (given classes) is extracted from databases and the classes are classified into several groups with respect to the characterization. Then, two kinds of sub-rules, characterization rules for each group and discrimination rules for each class in the group are induced. Finally, those two parts are integrated into one rule for each decision attribute. The proposed method was evaluated on a medical database, the experimental results of which show that induced rules correctly represent experts' decision processes.

1 Introduction

One of the most important problems in data mining is that extracted rules are not easy for domain experts to interpret. One of its reasons is that conventional rule induction methods [8] cannot extract rules, which plausibly represent experts' decision processes [10]: the description length of induced rules is too short, compared with the experts' rules. For example, rule induction methods, including AQ15 [4] and PRIMEROSE [10], induce the following common rule for muscle contraction headache from databases on differential diagnosis of headache:

$$[location = whole] \wedge [Jolt\ Headache = no] \wedge [Tenderness\ of\ M1 = yes] \\ \rightarrow \text{muscle contraction headache.}$$

This rule is shorter than the following rule given by medical experts.

$$\begin{aligned}
& [\text{Jolt Headache} = \text{no}] \\
& \wedge ([\text{Tenderness of M0} = \text{yes}] \vee [\text{Tenderness of M1} = \text{yes}] \vee [\text{Tenderness of M2} = \text{yes}]) \\
& \wedge [\text{Tenderness of B1} = \text{no}] \wedge [\text{Tenderness of B2} = \text{no}] \wedge [\text{Tenderness of B3} = \text{no}] \\
& \wedge [\text{Tenderness of C1} = \text{no}] \wedge [\text{Tenderness of C2} = \text{no}] \wedge [\text{Tenderness of C3} = \text{no}] \\
& \wedge [\text{Tenderness of C4} = \text{no}] \\
& \rightarrow \text{muscle contraction headache}
\end{aligned}$$

where $[\text{Tenderness of B1} = \text{no}]$ and $[\text{Tenderness of C1} = \text{no}]$ are added.

These results suggest that conventional rule induction methods do not reflect a mechanism of knowledge acquisition of medical experts.

In this paper, the characteristics of experts' rules are closely examined and a new approach to extract plausible rules is introduced, which consists of the following three procedures. First, the characterization of each decision attribute (a given class), a list of attribute-value pairs the supporting set of which covers all the samples of the class, is extracted from databases and the classes are classified into several groups with respect to the characterization. Then, two kinds of sub-rules, rules discriminating between each group and rules classifying each class in the group are induced. Finally, those two parts are integrated into one rule for each decision attribute. The proposed method was evaluated on medical databases, the experimental results of which show that induced rules correctly represent experts' decision processes.

2 Background: Problems with Rule Induction

As shown in the introduction, rules acquired from medical experts are much longer than those induced from databases the decision attributes of which are given by the same experts. This is because rule induction methods generally search for shorter rules, compared with decision tree induction. In the case of decision tree induction, the induced trees are sometimes too deep and in order for the trees to be learningful, pruning and examination by experts are required. One of the main reasons why rules are short and decision trees are sometimes long is that these patterns are generated only by one criteria, such as high accuracy or high information gain. The comparative study in this section suggests that experts should acquire rules not only by one criteria but by the usage of several measures. Those characteristics of medical experts' rules are fully examined not by comparing between those rules for the same class, but by comparing experts' rules with those for another class. For example, the classification rule for muscle contraction headache given in Section 1 is very similar to the following classification rule for disease of cervical spine:

$$\begin{aligned}
& [\text{Jolt Headache} = \text{no}] \\
& \wedge ([\text{Tenderness of M0} = \text{yes}] \vee [\text{Tenderness of M1} = \text{yes}] \vee [\text{Tenderness of M2} = \text{yes}]) \\
& \wedge ([\text{Tenderness of B1} = \text{yes}] \vee [\text{Tenderness of B2} = \text{yes}] \vee [\text{Tenderness of B3} = \text{yes}]) \\
& \vee [\text{Tenderness of C1} = \text{yes}] \vee [\text{Tenderness of C2} = \text{yes}] \vee [\text{Tenderness of C3} = \text{yes}] \\
& \vee [\text{Tenderness of C4} = \text{yes}] \\
& \rightarrow \text{disease of cervical spine}
\end{aligned}$$

The differences between these two rules are attribute-value pairs, from tenderness of B1 to C4. Thus, these two rules can be simplified into the following form:

$$\begin{aligned} a_1 \wedge A_2 \wedge \neg A_3 &\rightarrow \text{muscle contraction headache} \\ a_1 \wedge A_2 \wedge A_3 &\rightarrow \text{disease of cervical spine} \end{aligned}$$

The first two terms and the third one represent different reasoning. The first and second term a_1 and A_2 are used to differentiate muscle contraction headache and disease of cervical spine from other diseases. The third term A_3 is used to make a differential diagnosis between these two diseases. Thus, medical experts firstly selects several diagnostic candidates, which are very similar to each other, from many diseases and then make a final diagnosis from those candidates.

In the next section, a new approach for inducing the above rules is introduced. The differences between these two rules are attribute-value pairs, from tenderness of B1 to C4. Thus, these two rules can be simplified into the following form:

3 Rough Set Theory and Probabilistic Rules

3.1 Rough Set Notations

In the following sections, we use the following notations introduced by Grzymala-Busse and Skowron [9], which are based on rough set theory [5]. These notations are illustrated by a small database shown in Table 1, collecting the patients who complained of headache.

Let U denote a nonempty, finite set called the universe and A denote a nonempty, finite set of attributes, i.e., $a : U \rightarrow V_a$ for $a \in A$, where V_a is called the domain of a , respectively. Then, a decision table is defined as an information system, $A = (U, A \cup \{d\})$. For example, Table 1 is an information system with $U = \{1, 2, 3, 4, 5, 6\}$ and $A = \{age, location, nature, prodrome, nausea, M1\}$ and $d = class$. For $location \in A$, $V_{location}$ is defined as $\{ocular, lateral, whole\}$.

The atomic formulae over $B \subseteq A \cup \{d\}$ and V are expressions of the form $[a = v]$, called descriptors over B , where $a \in B$ and $v \in V_a$. The set $F(B, V)$ of formulas over B is the least set containing all atomic formulas over B and closed with respect to disjunction, conjunction and negation. For example, $[location = ocular]$ is a descriptor of B .

For each $f \in F(B, V)$, f_A denote the meaning of f in A , i.e., the set of all objects in U with property f , defined inductively as follows.

1. If f is of the form $[a = v]$ then, $f_A = \{s \in U | a(s) = v\}$
2. $(f \wedge g)_A = f_A \cap g_A$; $(f \vee g)_A = f_A \cup g_A$; $(\neg f)_A = U - f_A$

For example, $f = [location = whole]$ and $f_A = \{2, 4, 5, 6\}$. As an example of a conjunctive formula, $g = [location = whole] \wedge [nausea = no]$ is a descriptor of U and g_A is equal to $\{2, 5\}$.

By the use of the framework above, classification accuracy and coverage, or true positive rate is defined as follows.

Table 1. An example of database

	age	loc	nat	prod	nau	M1	class
1	50...59	occ	per	no	no	yes	m.c.h.
2	40...49	who	per	no	no	yes	m.c.h.
3	40...49	lat	thr	yes	yes	no	migra
4	40...49	who	thr	yes	yes	no	migra
5	40...49	who	rad	no	no	yes	m.c.h.
6	50...59	who	per	no	yes	yes	psycho

DEFINITIONS: loc: location, nat: nature, prod: prodrome, nau: nausea, M1: tenderness of M1, who: whole, occ: occular, lat: lateral, per: persistent, thr: throbbing, rad: radiating, m.c.h.: muscle contraction headache, migra: migraine, psycho: psychological pain,

Definition 1.

Let R and D denote a formula in $F(B, V)$ and a set of objects which belong to a decision d . Classification accuracy and coverage (true positive rate) for $R \rightarrow d$ is defined as:

$$\alpha_R(D) = \frac{|R_A \cap D|}{|R_A|} (= P(D|R)), \text{ and } \kappa_R(D) = \frac{|R_A \cap D|}{|D|} (= P(R|D)),$$

where $|S|$, $\alpha_R(D)$, $\kappa_R(D)$ and $P(S)$ denote the cardinality of a set S , a classification accuracy of R as to classification of D and coverage (a true positive rate of R to D), and probability of S , respectively.

In the above example, when R and D are set to $[nau = 1]$ and $[class = \text{migraine}]$, $\alpha_R(D) = 2/3 = 0.67$ and $\kappa_R(D) = 2/2 = 1.0$.

It is notable that $\alpha_R(D)$ measures the degree of the sufficiency of a proposition, $R \rightarrow D$, and that $\kappa_R(D)$ measures the degree of its necessity. For example, if $\alpha_R(D)$ is equal to 1.0, then $R \rightarrow D$ is true. On the other hand, if $\kappa_R(D)$ is equal to 1.0, then $D \rightarrow R$ is true. Thus, if both measures are 1.0, then $R \leftrightarrow D$.

3.2 Probabilistic Rules

According to the definitions, probabilistic rules with high accuracy and coverage are defined as:

$$R \xrightarrow{\alpha, \kappa} d \text{ s.t. } R = \bigvee_i R_i = \bigvee \bigwedge_j [a_j = v_k], \alpha_{R_i}(D) \geq \delta_\alpha \text{ and } \kappa_{R_i}(D) \geq \delta_\kappa,$$

where δ_α and δ_κ denote given thresholds for accuracy and coverage, respectively. For the above example shown in Table 1, probabilistic rules for m.c.h. are given as follows:

$$\begin{aligned} [M1 = \text{yes}] &\rightarrow \text{m.c.h. } \alpha = 3/4 = 0.75, \kappa = 1.0, \\ [nau = \text{no}] &\rightarrow \text{m.c.h. } \alpha = 3/3 = 1.0, \kappa = 1.0, \end{aligned}$$

where δ_α and δ_κ are set to 0.75 and 0.5, respectively.

3.3 Characterization Sets

In order to model medical reasoning, a statistical measure, coverage plays an important role in modeling, which is a conditional probability of a condition (R) under the decision D (P(R—D)). Let us define a characterization set of D, denoted by $L(D)$ as a set, each element of which is an elementary attribute-value pair R with coverage being larger than a given threshold, δ_κ . That is,

$$L_{\delta_\kappa} = \{[a_i = v_j] | \kappa_{[a_i=v_j]}(D) \geq \delta_\kappa\}$$

Then, three types of relations between characterization sets can be defined as follows:

$$\begin{aligned} \text{Independent type: } & L_{\delta_\kappa}(D_i) \cap L_{\delta_\kappa}(D_j) = \phi, \\ \text{Boundary type: } & L_{\delta_\kappa}(D_i) \cap L_{\delta_\kappa}(D_j) \neq \phi, \text{ and} \\ \text{Positive type: } & L_{\delta_\kappa}(D_i) \subseteq L_{\delta_\kappa}(D_j). \end{aligned}$$

All three definitions correspond to the negative region, boundary region, and positive region[4], respectively, if a set of the whole elementary attribute-value pairs will be taken as the universe of discourse. For the above example in Table 1, let D_1 and D_2 be m.c.h. and migraine and let the threshold of the coverage is larger than 0.6. Then, since

$$\begin{aligned} L_{0.6}(m.c.h.) &= \{[age = 40 - 49], [location = whole], [nature = persistent], \\ &\quad [prodrome = no], [nausea = no], [M1 = yes]\}, \text{ and} \\ L_{0.6}(migraine) &= \{[age = 40 - 49], [nature = throbbing], \\ &\quad [nausea = yes], [M1 = no]\}, \end{aligned}$$

the relation between m.c.h. and migraine is boundary type when the threshold is set to 0.6. Thus, the factors that contribute to differential diagnosis between these two are: $[location = whole]$, $[nature = persistent]$, $[nature = throbbing]$, $[prodrome = no]$, $[nausea = yes]$, $[nausea = no]$, $[M1 = yes]$, $[M1 = no]$. In these pairs, three attributes: nausea and M1 are very important. On the other hand, let D_1 and D_2 be m.c.h. and psycho and let the threshold of the coverage is larger than 0.6. Then, since

$$\begin{aligned} L_{0.6}(psycho) &= \{[age = 50 - 59], [location = whole], [nature = persistent], \\ &\quad [prodrome = no], [nausea = yes], [M1 = yes]\}, \end{aligned}$$

the relation between m.c.h. and psycho is also boundary. Thus, in the case of Table 1, age, nausea and M1 are very important factors for differential diagnosis.

According to the rules acquired from medical experts, medical differential diagnosis is a focusing mechanism: first, medical experts focus on some general category of diseases, such as vascular or muscular headache. After excluding the possibility of other categories, medical experts proceed into the further differential diagnosis between diseases within a general category. In this type of reasoning, subcategory type of characterization is the most important one. However, since medical knowledge has some degree of uncertainty, boundary type with high overlapped region may have to be treated like subcategory type. To check this boundary type, we use rough inclusion measure defined below.

3.4 Rough Inclusion

In order to measure the similarity between classes with respect to characterization, we introduce a rough inclusion measure μ , which is defined as follows.

$$\mu(S, T) = \frac{|S \cap T|}{|S|}.$$

It is notable that if $S \subseteq T$, then $\mu(S, T) = 1.0$, which shows that this relation extends subset and superset relations. This measure is introduced by Polkowski and Skowron in their study on rough mereology [6]. Whereas rough mereology firstly applies to distributed information systems, its essential idea is rough inclusion: rough inclusion focuses on set-inclusion to characterize a hierarchical structure based on a relation between a subset and superset. Thus, application of rough inclusion to capturing the relations between classes is equivalent to constructing rough hierarchical structure between classes, which is also closely related with information granulation proposed by Zadeh [12]. Let us illustrate how this measure is applied to hierarchical rule induction by using Table 1. When the threshold for the coverage is set to 0.6,

$$\begin{aligned} \mu(L_{0.6}(m.c.h.), L_{0.6}(migraine)) &= \frac{|\{[age=40-49]\}|}{|\{[age=40-49], [location=whole], \dots\}|} = \frac{1}{6} \\ \mu(L_{0.6}(m.c.h.), L_{0.6}(psycho)) &= \frac{|\{[location=whole], [nature=persistent], [prodrome=no], [M1=yes]\}|}{|\{[age=40-49], [location=whole], \dots\}|} = \frac{4}{6} = \frac{2}{3} \\ \mu(L_{0.6}(migraine), L_{0.6}(psycho)) &= \frac{|\{[nausea=yes]\}|}{|\{[age=40-49], [nature=throbbing], \dots\}|} = \frac{1}{4} \end{aligned}$$

These values show that the characterization set of m.c.h. is closer to that of psycho than that of migraine. Therefore, if the threshold for rough inclusion is set to 0.6, the characterization set of m.c.h. is roughly included by that of psycho. On the other hand, the characterization set of migraine is independent of those of m.c.h. and psycho. Thus, the differential diagnosis process consists of two process: the first process should discriminate between migraine and the group of m.c.h. and psycho. Then, the second process discriminate between m.c.h. and psycho. This means that the discrimination rule of m.c.h. is composed of (discrimination between migraine and the group)+ (discrimination between m.c.h. and psycho). In the case of L0.6, since the intersection of the characterization set of m.c.h. and psycho is $\{[location = whole], [nature = persistent], [prodrome = no], [M1 = yes]\}$, and the differences in attributes between this group and migraine is nature, M1. So, one of the candidates of discrimination rule is

$$[nature = throbbing] \wedge [M1 = no] \rightarrow migraine$$

The second discrimination rule is derived from the difference between the characterization set of m.c.h. and psycho: So, one of the candidate of the second discrimination rule is: $[age = 40 - 49] \rightarrow m.c.h.$ or $[nausea = no] \rightarrow m.c.h.$ Combining these two rules, we can obtain a diagnostic rule for m.c.h. as:

$$\neg([nature = throbbing] \wedge [M1 = no]) \wedge [age = 40 - 49] \rightarrow m.c.h.$$

4 Rule Induction

Rule induction(Fig 1.) consists of the following three procedures. First, the characterization of each given class, a list of attribute-value pairs the supporting set of which covers all the samples of the class, is extracted from databases and the classes are classified into several groups with respect to the characterization. Then, two kinds of sub-rules, rules discriminating between each group and rules classifying each class in the group are induced(Fig 2). Finally, those two parts are integrated into one rule for each decision attribute(Fig 3).¹

```

procedure Rule Induction (Total Process);
  var
     $i$  : integer;    $M, L, R$  : List;
     $L_D$  : List; /* A list of all classes */
  begin
    Calculate  $\alpha_R(D_i)$  and  $\kappa_R(D_i)$  for each elementary relation  $R$  and each class  $D_i$ ;
    Make a list  $L(D_i) = \{R | \kappa_R(D) = 1.0\}$  for each class  $D_i$ ;
    while ( $L_D \neq \phi$ ) do
      begin
         $D_i := first(L_D)$ ;  $M := L_D - D_i$ ;
        while ( $M \neq \phi$ ) do
          begin
             $D_j := first(M)$ ;
            if ( $(\mu(L(D_j), L(D_i)) \leq \delta_\mu)$ ) then  $L_2(D_i) := L_2(D_i) + \{D_j\}$ ;
             $M := M - D_j$ ;
          end
          end
          Make a new decision attribute  $D'_i$  for  $L_2(D_i)$ ;
           $L_D := L_D - D_i$ ;
        end
        Construct a new table ( $T_2(D_i)$ ) for  $L_2(D_i)$ .
        Construct a new table( $T(D'_i)$ ) for each decision attribute  $D'_i$ ;
        Induce classification rules  $R_2$  for each  $L_2(D)$ ; /* Fig.2 */
        Store Rules into a List  $R(D)$ 
        Induce classification rules  $R_d$  for each  $D'$  in  $T(D')$ ; /* Fig.2 */
        Store Rules into a List  $R(D') (= R(L_2(D_i)))$ 
        Integrate  $R_2$  and  $R_d$  into a rule  $R_D$ ; /* Fig.3 */
      end {Rule Induction };

```

Fig. 1. An algorithm for rule induction

¹ This method is an extension of PRIMEROSE4 reported in [11]. In the former paper, only rigid set-inclusion relations are considered for grouping; on the other hand, rough-inclusion relations are introduced in this approach. Recent empirical comparison between set-inclusion method and rough-inclusion method shows that the latter approach outperforms the former one.

```

procedure Induction of Classification Rules;
  var
     $i$  : integer;    $M, L_i$  : List;
  begin
     $L_1 := L_{er}$ ; /*  $L_{er}$ : List of Elementary Relations */
     $i := 1$ ;    $M := \{\}$ ;
    for  $i := 1$  to  $n$  do      /*  $n$ : Total number of attributes */
      begin
        while (  $L_i \neq \{\}$  ) do
          begin
            Select one pair  $R = \wedge[a_i = v_j]$  from  $L_i$ ;
             $L_i := L_i - \{R\}$ ;
            if ( $\alpha_R(D) \geq \delta_\alpha$ ) and ( $\kappa_R(D) \geq \delta_\kappa$ )
              then do  $S_{ir} := S_{ir} + \{R\}$ ; /* Include  $R$  as Inclusive Rule */
            else  $M := M + \{R\}$ ;
          end
           $L_{i+1} :=$  (A list of the whole combination of the conjunction formulae in  $M$ );
        end
      end {Induction of Classification Rules };

```

Fig. 2. An algorithm for classification rules

Example

Let us illustrate how the introduced algorithm works by using a small database in Table 1. For simplicity, two thresholds δ_α and δ_μ are set to 1.0, which means that only deterministic rules should be induced and that only subset and superset relations should be considered for grouping classes.

After the first and second step, the following three sets will be obtained: $L(m.c.h.) = \{[prod = no], [M1 = yes]\}$, $L(migra) = \{[age = 40..49], [nat = who], [prod = yes], [nau = yes], [M1 = no]\}$, and $L(psycho) = \{[age = 50..59], [loc = who], [nat = per], [prod = no], [nau = no], [M1 = yes]\}$. Thus, since a relation $L(psycho) \subset L(m.c.h.)$ holds (i.e., $\mu(L(m.c.h.), L(psycho)) = 1.0$), a new decision attribute is $D_1 = \{m.c.h., psycho\}$ and $D_2 = \{migra\}$, and a partition $P = \{D_1, D_2\}$ is obtained. From this partition, two decision tables will be generated, as shown in Table 2 and Table 3 in the fifth step.

In the sixth step, classification rules for D_1 and D_2 are induced from Table 2. For example, the following rules are obtained for D_1 .

$$\begin{array}{ll}
[M1 = yes] & \rightarrow D_1 \alpha = 1.0, \kappa = 1.0, \text{ supported by } \{1,2,5,6\} \\
[prod = no] & \rightarrow D_1 \alpha = 1.0, \kappa = 1.0, \text{ supported by } \{1,2,5,6\} \\
[nau = no] & \rightarrow D_1 \alpha = 1.0, \kappa = 0.75, \text{ supported by } \{1,2,5\} \\
[nat = per] & \rightarrow D_1 \alpha = 1.0, \kappa = 0.75, \text{ supported by } \{1,2,6\} \\
[loc = who] & \rightarrow D_1 \alpha = 1.0, \kappa = 0.75, \text{ supported by } \{2,5,6\} \\
[age = 50..59] & \rightarrow D_1 \alpha = 1.0, \kappa = 0.5, \text{ supported by } \{2,6\}
\end{array}$$

In the seventh step, classification rules for $m.c.h.$ and $psycho$ are induced from Table 3. For example, the following rules are obtained from $m.c.h.$.


```

procedure Rule Integration;
  var
     $i$  : integer;    $M, L_2$  : List;  $R(D_i)$  : List; /* A list of rules for  $D_i$  */
     $L_D$  : List; /* A list of all classes */
  begin
    while ( $L_D \neq \phi$ ) do
      begin
         $D_i := first(L_D)$ ;  $M := L_2(D_i)$ ;
        Select one rule  $R' \rightarrow D'_i$  from  $R(L_2(D_i))$ .
        while ( $M \neq \phi$ ) do
          begin
             $D_j := first(M)$ ;
            Select one rule  $R \rightarrow d_j$  for  $D_j$ ;
            Integrate two rules:  $R \wedge R' \rightarrow d_j$ .
             $M := M - \{D_j\}$ ;
          end
           $L_D := L_D - D_i$ ;
        end
      end
    end {Rule Combination}

```

Fig. 3. An algorithm for rule integration

Table 2. A table for a new partition P

	age	loc	nat	prod	nau	M1	class
1	50...59	occ	per	0	0	1	D_1
2	40...49	who	per	0	0	1	D_1
3	40...49	lat	thr	1	1	0	D_2
4	40...49	who	thr	1	1	0	D_2
5	40...49	who	rad	0	0	1	D_1
6	50...59	who	per	0	1	1	D_1

$$\begin{aligned}
[nau = no] &\rightarrow m.c.h. \alpha = 1.0, \kappa = 1.0, \text{ supported by } \{1,2,5\} \\
[age = 40...49] &\rightarrow m.c.h. \alpha = 1.0, \kappa = 0.67, \text{ supported by } \{2,5\}
\end{aligned}$$

In the eighth step, these two kinds of rules are integrated in the following way. Rule $[M1 = yes] \rightarrow D_1$, $[nau = no] \rightarrow m.c.h.$ and $[age = 40...49] \rightarrow m.c.h.$ have a supporting set which is a subset of $\{1,2,5,6\}$. Thus, the following rules are obtained:

$$\begin{aligned}
[M1 = yes] \ \& \ [nau=no] &\rightarrow m.c.h. \alpha = 1.0, \kappa = 1.0, \text{ supported by } \{1,2,5\} \\
[M1 = yes] \ \& \ [age=40...49] &\rightarrow m.c.h. \alpha = 1.0, \kappa = 0.67, \text{ supported by } \{2,5\}
\end{aligned}$$

5 Experimental Results

The above rule induction algorithm was implemented in PRIMEROSE4.5 (Probabilistic Rule Induction Method based on Rough Sets Ver 4.5), and was applied

Table 3. A table for D_1

	age	loc	nat	prod	nau	M1	class
1	50...59	occ	per	0	0	1	m.c.h.
2	40...49	who	per	0	0	1	m.c.h.
5	40...49	who	rad	0	0	1	m.c.h.
6	50...59	who	per	0	1	1	psycho

to databases on differential diagnosis of headache, meningitis and cerebrovascular diseases (CVD), whose precise information is given in Table 4. In these experiments, δ_α and δ_κ were set to 0.75 and 0.5, respectively. Also, the threshold for grouping is set to 0.8.² This system was compared with PRIMEROSE4.0 [11], PRIMEROSE [10] C4.5 [7], CN2 [2], AQ15 [4] with respect to the following points: length of rules, similarities between induced rules and expert's rules and performance of rules.

In this experiment, length was measured by the number of attribute-value pairs used in an induced rule and Jaccard's coefficient was adopted as a similarity measure [3]. Concerning the performance of rules, ten-fold cross-validation was applied to estimate classification accuracy.

Table 4. Information about databases

Domain	Samples	Classes	Attributes
Headache	52119	45	147
CVD	7620	22	285
Meningitis	141	4	41

Table 5 shows the experimental results, which suggest that PRIMEROSE4.5 outperforms PRIMEROSE4(set-inclusion approach) and the other four rule induction methods and induces rules very similar to medical experts' ones.

6 Discussion: What Is Discovered?

Several interesting rules for migraine were found. Since migraine is a kind of vascular disease, the first part discriminates between migraine and other diseases. This part is obtained as :

$$\begin{aligned}
 &[Nature : Persistent] \& \neg [History : acuteorparoxysmal] \\
 &\& [JoltHeadache : yes] \rightarrow \{commonmigraine, classicmigraine\}
 \end{aligned}$$

² These values are given by medical experts as good thresholds for rules in these three domains.

Table 5. Experimental results

Method	Length	Similarity	Accuracy
Headache			
PRIMEROSE4.5	8.8 ± 0.27	0.95 ± 0.08	$95.2 \pm 2.7\%$
PRIMEROSE4.0	7.3 ± 0.35	0.74 ± 0.05	$88.3 \pm 3.6\%$
Experts	9.1 ± 0.33	1.00 ± 0.00	$98.0 \pm 1.9\%$
PRIMEROSE	5.3 ± 0.35	0.54 ± 0.05	$88.3 \pm 3.6\%$
C4.5	4.9 ± 0.39	0.53 ± 0.10	$85.8 \pm 1.9\%$
CN2	4.8 ± 0.34	0.51 ± 0.08	$87.0 \pm 3.1\%$
AQ15	4.7 ± 0.35	0.51 ± 0.09	$86.2 \pm 2.9\%$
Meningitis			
PRIMEROSE4.5	2.6 ± 0.19	0.91 ± 0.08	$82.0 \pm 3.7\%$
PRIMEROSE4.0	2.8 ± 0.45	0.72 ± 0.25	$81.1 \pm 2.5\%$
Experts	3.1 ± 0.32	1.00 ± 0.00	$85.0 \pm 1.9\%$
PRIMEROSE	1.8 ± 0.45	0.64 ± 0.25	$72.1 \pm 2.5\%$
C4.5	1.9 ± 0.47	0.63 ± 0.20	$73.8 \pm 2.3\%$
CN2	1.8 ± 0.54	0.62 ± 0.36	$75.0 \pm 3.5\%$
AQ15	1.7 ± 0.44	0.65 ± 0.19	$74.7 \pm 3.3\%$
CVD			
PRIMEROSE4.5	7.6 ± 0.37	0.89 ± 0.05	$74.3 \pm 3.2\%$
PRIMEROSE4.0	5.9 ± 0.35	0.71 ± 0.05	$72.3 \pm 3.1\%$
Experts	8.5 ± 0.43	1.00 ± 0.00	$82.9 \pm 2.8\%$
PRIMEROSE	4.3 ± 0.35	0.69 ± 0.05	$74.3 \pm 3.1\%$
C4.5	4.0 ± 0.49	0.65 ± 0.09	$69.7 \pm 2.9\%$
CN2	4.1 ± 0.44	0.64 ± 0.10	$68.7 \pm 3.4\%$
AQ15	4.2 ± 0.47	0.68 ± 0.08	$68.9 \pm 2.3\%$

which are reasonable for medical expert knowledge. Rather, medical experts pay attention to the corresponding parts and grouping of other diseases:

$$\begin{aligned}
& [Nature : Persistent] \& \neg [History : acuteorparoxysmal] \\
& \& [JoltHeadache : yes] \rightarrow \{meningitis, Braintumor\}, \\
& [Nature : Persistent] \& \neg [History : acuteorparoxysmal] \\
& \& [JoltHeadache : no] \rightarrow \{musclecontractionheadache\},
\end{aligned}$$

The former one is much more interesting and unexpected to medical experts, while the latter one is reasonable.

The second part discriminates between common migraine and classic migraine. These parts are obtained as :

$$\begin{aligned}
& [Age > 40] \& [Prodrome : no] \rightarrow CommonMigraineand \\
& [Age < 20] \& [Prodrome : yes] \rightarrow ClassicMigraine,
\end{aligned}$$

where the attribute age is unexpected to medical experts. Migraine can be observed mainly by women, and it is observed that the frequency of headache decreases as women are getting older. Thus, the factor age support these experiences.

7 Conclusion

In this paper, the characteristics of experts' rules are closely examined, whose empirical results suggest that grouping of diseases are very important to realize automated acquisition of medical knowledge from clinical databases. Thus, we focus on the role of coverage in focusing mechanisms and propose an algorithm on grouping of diseases by using this measure. The above experiments show that rule induction with this grouping generates rules, which are similar to medical experts' rules and they suggest that our proposed method should capture medical experts' reasoning. The proposed method was evaluated on three medical databases, the experimental results of which show that induced rules correctly represent experts' decision processes.

Acknowledgments

This work was supported by the Grant-in-Aid for Scientific Research (13131208) on Priority Areas (No.759) "Implementation of Active Mining in the Era of Information Flood" by the Ministry of Education, Science, Culture, Sports, Science and Technology of Japan.

References

1. Aha, D. W., Kibler, D., and Albert, M. K., Instance-based learning algorithm. *Machine Learning*, 6, 37-66, 1991.
2. Clark, P. and Niblett, T., The CN2 Induction Algorithm. *Machine Learning*, 3, 261-283, 1989. 432
3. Everitt, B. S., *Cluster Analysis*, 3rd Edition, John Wiley & Son, London, 1996. 432
4. Michalski, R. S., Mozetic, I., Hong, J., and Lavrac, N., The Multi-Purpose Incremental Learning System AQ15 and its Testing Application to Three Medical Domains, in *Proceedings of the fifth National Conference on Artificial Intelligence*, 1041-1045, AAAI Press, Menlo Park, 1986. 423, 432
5. Pawlak, Z., *Rough Sets*. Kluwer Academic Publishers, Dordrecht, 1991. 425
6. Polkowski, L. and Skowron, A.: Rough mereology: a new paradigm for approximate reasoning. *Intern. J. Approx. Reasoning* **15**, 333-365, 1996. 428
7. Quinlan, J. R., *C4.5 - Programs for Machine Learning*, Morgan Kaufmann, Palo Alto, 1993. 432
8. *Readings in Machine Learning*, (Shavlik, J. W. and Dietterich, T. G., eds.) Morgan Kaufmann, Palo Alto, 1990. 423
9. Skowron, A. and Grzymala-Busse, J. From rough set theory to evidence theory. In: Yager, R., Fedrizzi, M. and Kacprzyk, J.(eds.) *Advances in the Dempster-Shafer Theory of Evidence*, pp.193-236, John Wiley & Sons, New York, 1994. 425
10. Tsumoto, S., Automated Induction of Medical Expert System Rules from Clinical Databases based on Rough Set Theory. *Information Sciences* **112**, 67-84, 1998. 423, 432
11. Tsumoto, S. Extraction of Experts' Decision Rules from Clinical Databases using Rough Set Model *Intelligent Data Analysis*, 2(3), 1998. 429, 432

12. Zadeh, L. A., Toward a theory of fuzzy information granulation and its certainty in human reasoning and fuzzy logic. *Fuzzy Sets and Systems* **90**, 111-127, 1997. 428
13. Ziarko, W., Variable Precision Rough Set Model. *Journal of Computer and System Sciences*. 46, 39-59, 1993.

Efficiently Mining Approximate Models of Associations in Evolving Databases

Adriano Veloso¹, Bruno Gusmão¹, Wagner Meira Jr.¹, Marcio Carvalho¹,
Srini Parthasarathy², and Mohammed Zaki³

¹ Computer Science Department, Universidade Federal de Minas Gerais, Brazil
{adrianov,gusmao,meira,mlbc}@dcc.ufmg.br

² Department of Computer and Information Science, The Ohio-State University, USA
srini@cis.ohio-state.edu

³ Computer Science Department, Rensselaer Polytechnic Institute, USA
zaki@cs.rpi.edu

Abstract. Much of the existing work in machine learning and data mining has relied on devising efficient techniques to build accurate models from the data. Research on how the accuracy of a model changes as a function of dynamic updates to the databases is very limited. In this work we show that extracting this information: knowing which aspects of the model are changing; and how they are changing as a function of data updates; can be very effective for interactive data mining purposes (where response time is often more important than model quality as long as model quality is not too far off the best (exact) model).

In this paper we consider the problem of generating approximate models within the context of association mining, a key data mining task. We propose a new approach to incrementally generate approximate models of associations in evolving databases. Our approach is able to detect how patterns evolve over time (an interesting result in its own right), and uses this information in generating approximate models with high accuracy at a fraction of the cost (of generating the exact model). Extensive experimental evaluation on real databases demonstrates the effectiveness and advantages of the proposed approach.

1 Introduction

One of the main characteristics of the digital information era is the ability to store huge amounts of data. However, extracting knowledge, often referred to as data mining, from such data efficiently poses several important challenges. First, the volume of data operated on is typically very large, and the tasks involved are inherently I/O intensive. Second, the computational demands are quite high. Third, many of these datasets are dynamic (E-commerce databases, Web-based applications), in the sense that they are constantly being updated (evolving datasets).

Researchers have evaluated data stratification mechanisms such as sampling to handle the first problem and memory efficient and parallel computing techniques to handle the second problem. Simply re-executing the algorithms to

handle the third problem results in excessive wastage of computational resources and often does not meet the stringent interactive response times required by the data miner. In these cases, it may not be possible to mine the entire database over and over again. This has motivated the design of incremental algorithms, i.e., algorithms that are capable of updating the frequent itemsets, and thus its associations, by taking into account just the transactions recorded since the last mine operation. In this paper, we propose an approximate incremental algorithm to mine association rules that advances the state-of-the-art in this area.

Association mining is a key data mining task. It is used most often for market basket data analysis, but more recently it has also been used in such far-reaching domains as bioinformatics [7], text mining [14] and scientific computing [9]. Previous research efforts have produced many efficient sequential algorithms [6,1,8,18,19], several parallel algorithms [20,13,3], and a few incremental algorithms for determining associations [16,15,2,4].

The majority of the incremental algorithms studied employ specific data structures to maintain the information previously mined so that it can be augmented by the updates. These techniques are designed to produce exact results, as would be produced by an algorithm running on the entire original database. However, if response time is paramount, these algorithms may still be unacceptable. In this case, it is needed is a way to efficiently estimate the association parameters (support, confidence) without actually computing them and thus saving on both computational and I/O time. Our approach relies on extracting historical trends associated with each itemset and using them to estimate these parameters. For instance, if an itemset support is roughly constant across time, it may not be necessary to compute its exact frequency value. An approximate value may have the same effect. On the other hand, if an itemset shows a consistent increase or decrease trend, its support may be estimated as a function of the number of updates after the last exact count number and the slope associated with the trend.

The main contributions of this paper can be summarized as follows:

- We propose an approximate incremental algorithm, WAVE, for mining association rules, based on trends of itemset frequency value changes.
- We evaluate the above algorithm based on the quality of its estimates (i.e., how close they are from to the exact model) and its performance (when compared against a state-of-the-art incremental algorithm) when mining several real datasets.

We begin by formally presenting the problem of finding association rules in the next section. In Section 3 we present our approach for mining approximate models of associations. The effectiveness of our approach is experimentally analyzed in Section 4. Finally, in Section 5 we conclude our work and present directions for future work.

2 Problem Description and Related Work

2.1 Association Mining Problem

The association mining task can be stated as follows: Let $\mathcal{I} = \{1, 2, \dots, n\}$ be a set of n distinct attributes, also called items, and let $\mathcal{D}$ be the input database. Typically $\mathcal{D}$ is arranged as a set of transactions, where each transaction T has a unique identifier TID and contains a set of items such that $T \subseteq \mathcal{I}$. A set of items $X \subseteq \mathcal{I}$ is called an itemset. For an itemset X , we denote its corresponding *tidlist* as the set of all $TIDs$ that contain X as a subset. The support of an itemset X , denoted $\sigma(X)$, is the percentage of transactions in $\mathcal{D}$ in which X occurs as a subset. An itemset is frequent if its support $\sigma(X) \geq \text{minsup}$, where *minsup* is a user-specified minimum support threshold.

An association rule is an expression $A \xrightarrow{p} B$, where A and B are itemsets. The support of the rule is $\sigma(A \cup B)$ (i.e., the joint probability of a transaction containing both A and B), and the confidence $p = \sigma(A \cup B) / \sigma(A)$ (i.e., the conditional probability that a transaction contains B , given that it contains A). A rule is frequent if the itemset $A \cup B$ is frequent. A rule is confident if $p \geq \text{minconf}$, where *minconf* is a user-specified minimum confidence threshold.

Finding frequent itemsets is computationally and I/O intensive. Let $|\mathcal{I}| = m$ be the number of items. The search space for enumeration of all frequent itemsets is 2^m , which is exponential in m . This high computational cost may be acceptable when the database is static, but not in domains with evolving data, since the itemset enumeration process will be frequently repeated. In this paper we only deal with how to efficiently mine frequent itemsets in evolving databases.

2.2 Related Work

There has been a lot of research in developing efficient algorithms for mining frequent itemsets. A general survey of these algorithms can be found in [17]. Most of these algorithms enumerate all frequent itemsets. There also exist methods which only generate frequent closed itemsets [18] and maximal frequent itemsets [6]. While these methods generate a reduced number of itemsets, they still need to mine the entire database in order to generate the set of valid associations, therefore these methods are not efficient in mining evolving databases.

Some recent effort has been devoted to the problem of incrementally mine frequent itemsets [10,15,16,2,4,5,12]. An important subproblem is to determine how often to update the current model. While some algorithms update the model after a fixed number of new transactions [16,15,2,4], the DELI algorithm, proposed by Lee and Cheung [10], uses statistical sampling methods to determine when the current model is outdated. A similar approach proposed by Ganti *et al* (DEMON [5]) monitors changes in the data stream to determine when to update. An efficient incremental algorithm, called ULI, was proposed by Thomas [15] *et al*. ULI strives to reduce the I/O requirements for updating the set of frequent itemsets by maintaining the previous frequent itemsets and the *negative border* [11] along with their support counts. The whole database is scanned just

once, but the incremental database must be scanned as many times as the size of the longest frequent itemset.

The proposed algorithm, WAVE, is different from the above approaches in several ways. First, while these approaches need to perform $O(n)$ database scans (n is the size of the largest frequent itemset), WAVE requires only one scan on the incremental database and only a partial scan on the original database. Second, WAVE supports selective updates, that is, instead of determining when to update the whole set of frequent itemsets, WAVE identifies specifically which itemsets need to be updated and then updates only those itemsets. Finally, because WAVE employs simple estimation procedures it has the ability to improve the prediction accuracy while maintaining the update costs very small. The combination of incremental techniques and on-the-fly data stream analysis makes WAVE an efficient algorithm for mining frequent itemsets and associations in evolving, and potentially streaming databases.

3 The ZIGZAG and WAVE Algorithms

In previous work [16] we presented the ZIGZAG algorithm, a method which efficiently updates the set of frequent itemsets in evolving databases. ZIGZAG is based on maintaining maximal frequent itemsets (and associated supports of all frequent itemset subsets) across database updates. On an update, the maximal frequent itemsets are updated by a backtracking search approach, which is guided by the results of the previous mining iteration. In response to a user query ZIGZAG uses the upto-date maximal frequent itemsets¹ to construct the lattice of frequent itemsets in the database. As shown in [16] this approach of maintaining and tracking maximal itemsets across database updates, results in significant I/O and computational savings when compared with other state-of-the-art incremental approaches.

WAVE is an extension to ZIGZAG. WAVE essentially maintains the same data structure but adds the capability to determine when and how to update the maintained information. It relies on its ability to detect trends and estimate itemset frequency behavior as a function of updates. If an itemset can be well estimated, the exact frequency is not computed, otherwise it will be computed. In comparison to ZIGZAG, WAVE can significantly reduce the computation required to process an update but this reduction comes at some cost to accuracy (since we often estimate rather than compute frequencies). Contrasting to other incremental approaches [15,2,4,5] which generally monitor changes in the database to detect the best moment to update the entire set of itemsets, we choose instead to perform selective updates, that is, the support of every single itemset is completely updated only when we cannot compute a good estimate of its frequency.

Figure 1 depicts a real example that motivates our selective approach. This figure shows the correlation of two sets of popular itemsets. These popular itemsets are ranked by support (i.e., popularity ranking) and their relative positions

¹ The maximal frequent itemsets solely determine all frequent itemsets

are compared. When the set of popular itemsets is totally accurate, all the popular itemsets are in the correct position. From Figure 1 we can see a comparison of a totally accurate set of popular itemsets and a ranked set of itemsets which is becoming outdated as the database evolves. As we can see in this figure, although there were significant changes in the support of some popular itemsets, there are also a large number of popular itemsets which remain accurate (i.e., in the correct position) and do not need to be updated, and also a large number of popular itemsets which had evolved in a systematic way, following some type of trend. Our method relies on accurately identifying and categorizing such itemsets. We describe these categories next:

Invariant: The support of the itemset does not change significantly over time (i.e., it varies within a predefined threshold) as we add new transactions. This itemset is stable, and therefore, it need not be updated.

Predictable: It is possible to estimate the support of the itemset within a tolerance. This itemset presents a trend, that is, its support increases or decreases in a systematic way over time.

Unpredictable: It is not possible, given a set of approximation tools, to obtain a good estimate of the itemset support. Note, that it is desirable to have few unpredictable itemsets as these are the ones that cannot be estimated.

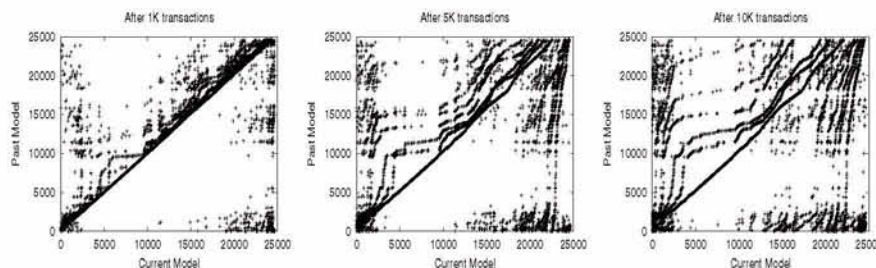


Fig. 1. Evolution of Frequent Itemsets. The X-Axis represents a Totally Accurate Ranking, while the Y-Axis represents an out-dated Ranking

There are many techniques that can be used to estimate the support of a given itemset. The search for such tools is probably endless, and is out of scope of this paper. We believe that the use of costly (time-wise) and sophisticated tools is unlikely to be useful, since their cost may approach or surpass the cost of executing an exact incremental mining algorithm such as ZIGZAG.

Using simple prediction tools (discussed later in this section) one can classify the set of all frequent itemsets across these three categories. Table 1 depicts the percentage of itemsets in each category for the WCup and WPortal databases as an illustration of the approximate approach's potential. From this table we can

Table 1. Ratio between Invariant, Predictable and Unpredictable Itemsets

Database	Invariant	Predictable	Unpredictable
WCup	7.2%	45.3%	47.5%
WPortal	9.1%	52.1%	38.8%

see that both databases present a significant number of invariant and predictable itemsets.

Note that there exists a major difference between invariant and predictable itemsets. If there is a large number of invariant itemsets in the database, the set of popular itemsets generated will remain accurate for a long time. On the other hand, if there is a large number of predictable itemsets, the model will lose accuracy over time. However, using simple models we show that one can generate pretty good estimates of these predictable itemsets, potentially maintaining the accuracy of the support of the popular itemsets.

WAVE is comprised of two phases. The first phase uses the tidlists associated with 1-items whose union is the itemset whose support we want to estimate. The second phase analyzes the sampled results to determine whether it is necessary to count the actual support of the itemset. Each of these phases is described below.

Phase 1: Discretized Support Estimation — The starting point of Phase 1 is the tidlists associated with 1-itemsets, which are always up-to-date since they are simply augmented by novel transactions. Formally, given two tidlists l_α and l_β associated with the itemsets α and β , we define that the exact tidlist of $\alpha \cup \beta$ is $l_{\alpha \cup \beta} = l_\alpha \cap l_\beta$. We estimate the upper bound on the merge of two tidlists as follows. We divide the tidlists into n bins. The upper bound of the intersection of corresponding bins is the smallest of the two bin values (each bin value corresponding to the number of entries in the bin). Note, that as long as transactions are ordered temporally, each bin gives us an approximate idea as to how a particular itemset behaved during a given time frame. The upper bounds associated with the bins are then used as input to our estimation technique, described next.

Phase 2: Support Estimation based on Linear Trend Detection — Phase 2 takes as input the information provided by Phase 1 in order to detect trends in itemset frequency. Trend detection is a valuable tool to predict the frequent itemsets behavior in the context of evolving databases. One of the most widespread trend detection techniques is linear regression, that finds a straight line that more closely describes the dataset. The model used by the linear regression is expressed as the function $y = a + bx$, where a is the y -intercept and b is the slope of the line that represents the linear relationship between x and y . In our scenario the x variable represents the number of transactions while the y variable represents the estimated support (obtained as a function of the upper bound estimates

from Phase 1). The *method of least squares* determines the values of a and b that minimize the sum of the squares of the errors, and it is widely used for generating linear regression models.

To verify the goodness of the model generated by the linear regression, we use the R^2 metric (which takes on values in the range 0 to 1) that reveals how closely the estimated y -values correlate to its actual y -values. A R^2 value close to 1 indicates that the regression equation is very reliable. In such cases, WAVE provides an approximated technique to find the support of predictable itemsets, an approach that does not have an analog in the itemset mining research. Whenever an itemset is predictable, its support can be simply predicted using the linear regression model, rather than recomputed with expensive database scans. Figure 2 shows the R^2 distribution for the two databases used in the experiments. This estimate technique achieves extraordinary savings in computational and I/O requirements, as we will see in Section 4.

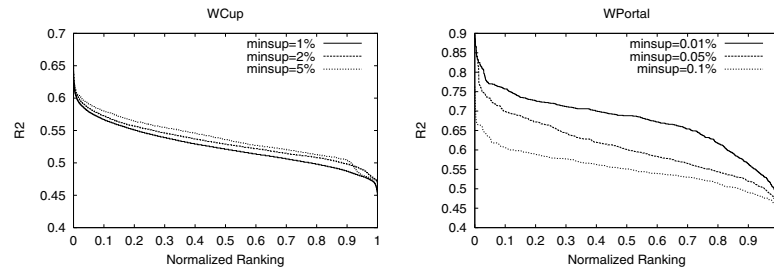


Fig. 2. R^2 Distribution in WCup and WPortal Databases

4 Experimental Evaluation

In this section we evaluate the precision, performance and scalability of WAVE and compare it to other incremental approaches. Real databases from actual applications were used as inputs in the experiments. The first database, WCup, comes from click stream data from the official site of the 1998 World Soccer Cup. WCup was extracted from a 62-day log, comprising 2,128,932 transactions over 4,768 unique items with an average transaction length of 8.5 items and a standard deviation of 11.2. The second database represents the access patterns of a Web Portal. The database, WPortal, comprises 432,601 transactions over 1,182 unique items, and each transaction contains an average length of 2.9 items. Our evaluation is based on three parameters given to WAVE:

Approximation tolerance— R^2 : the maximum approximation error acceptable.

Longevity: the number of transactions added to the database which triggers a complete update process.

Base length: the number of transactions effectively mined before we start the estimating process.

Thus, for each minimum support used, we performed multiple executions of the algorithm in different databases, where each execution employs a different combination of R^2 , *longevity*, and *base length*. Further, we employed three metrics in our evaluation:

Precision: This metric quantifies how good the approximation is. It is the linear correlation of two ordered sets of itemsets. The ranking criteria is the support, that is, two ordered sets are totally correlated if they are of the same length, and the same itemset appears in corresponding positions in both sets.

Work: This metric quantifies the amount of work performed by WAVE when compared to ULI. We measure the elapsed time for each algorithm while mining a given database in a dedicated single-processor machine. We then calculate the work as the ratio between the elapsed time for our approach and the elapsed time for ULI.

Resource consumption: This metric quantifies the amount of memory used by each algorithm. Observing this metric is interesting for the sake of practical evaluation of the use of WAVE in large databases.

The experiments were run on an *IBM - NetFinity* 750MHz processor with 512MB main memory. The source code for ULI [15], the state-of-the-art algorithm which was used to perform our comparisons, was kindly provided to us by its authors. Timings used to calculate the work metric are based on wall clock time.

4.1 Accuracy Experiments

Here we report the accuracy results for the databases described above. Firstly, we evaluate the precision achieved by WAVE. Next, we evaluate the gains in precision provided by WAVE. We employed different databases, minimum supports, *base lengths*, *longevities*, and R^2 . Figure 3(a) depicts the precision achieved by WAVE in the WCup database. From this figure we can observe that, as expected, the precision increases with the R^2 used. Surprisingly, for this database the precision decreases with the *base length* used. Further, the precision decreases with both the *longevity* and minimum support.

Slightly different results were observed for the same experiment using the WPortal database. As expected the precision decreases with the *longevity*. For *base lengths* as small as 50K transactions the lowest precision was achieved by the largest minimum support. We believe that this is because these small *base lengths* do not provide sufficient information about the database. For *base lengths* as large as 100K transactions, the lowest precision was always achieved by the lowest

minimum support. Interestingly, the highest precision was initially provided by the highest minimum support, but as we increase the R^2 value we notice a crossover point after which the second largest support value was the most precise.

We also evaluate the gains in precision achieved by WAVE. From Figure 4(a) we can observe that, using the WCup database, WAVE provides larger gains in precision for smaller values of minimum support. The opposite trend is observed when we evaluate the precision varying the *longevity*, that is, in general larger gains are achieved by larger *longevities*. It is obvious that WAVE loses precision over the time, but this result shows that WAVE can maintain a more accurate picture of the frequent itemsets for more time. Finally, the precision increases with the R^2 value, that is, increasing the precision criteria results in improved prediction precision.

The gains in precision achieved by WAVE were also evaluated using the WPortal database, and the results are depicted in Figure 4(b). In general we observe large gains for smaller values of minimum support. We can also observe that, in all cases, the higher the value of longevity, the larger is the gain in precision. One more time WAVE shows to be very robust in preserving the precision.

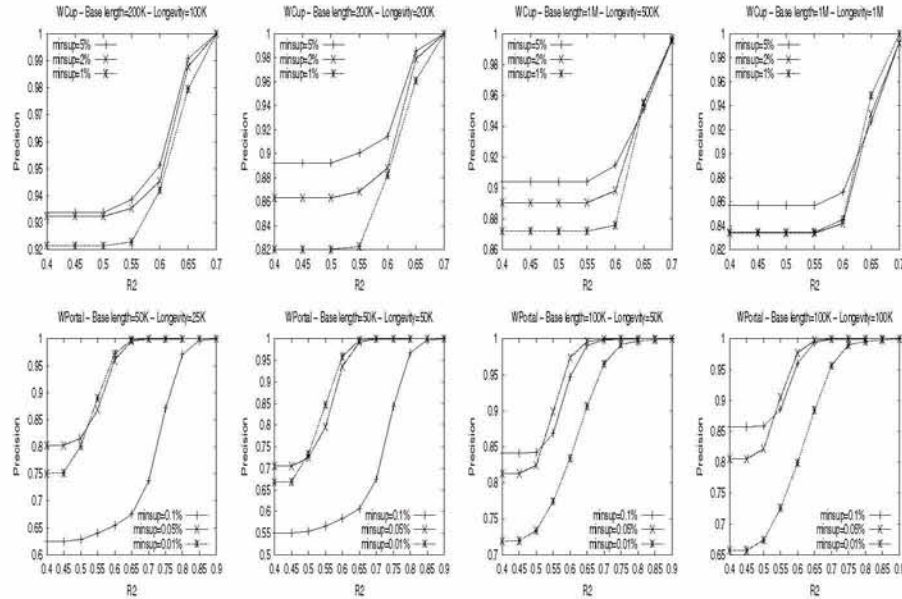


Fig. 3. Precision achieved by WAVE when varying *minimum support*, R^2 , *base length*, and *longevity* for a) WCup Database (top row), and b) WPortal Database (bottom row)

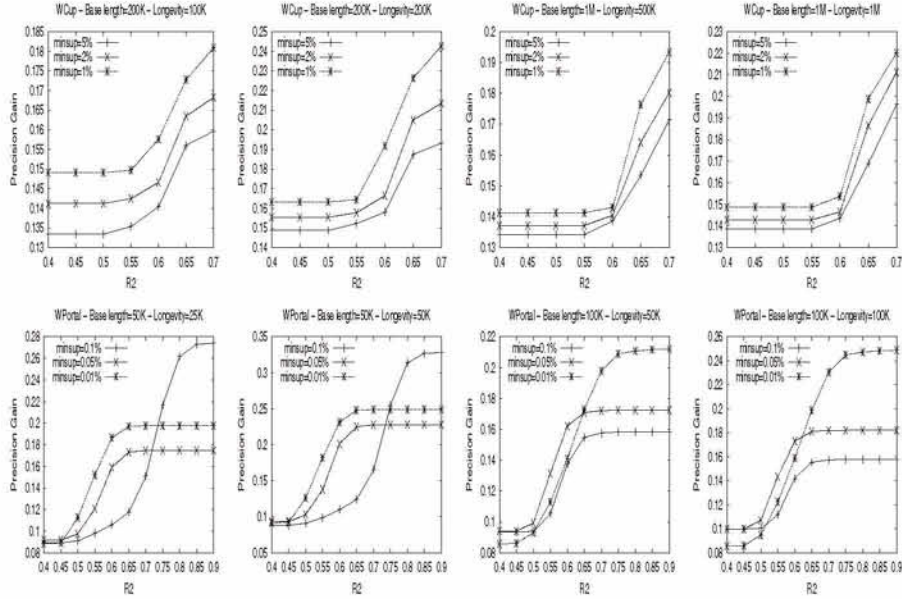


Fig. 4. Precision Gains provided by WAVE when varying *minimum support*, R^2 , *base length*, and *longevity* for a) WCup Database (top row), and b) WPortal Database (bottom row)

4.2 Performance Experiments

Now we verify the amount of work performed by WAVE in order to generate an approximate model of associations. From Figure 5(a) we can observe the results obtained using the WCup database. WAVE performs less work for smaller values of minimum support. This is mainly because ULI spent much more time than WAVE in mining with smaller values of minimum support. We can also observe that WAVE performs the same amount of work when the R^2 threshold reaches the value 0.7, no matter how much the minimum support value is. The reason is that there are only few itemsets with an approximation as good as 0.7, and all these itemsets have a support higher than 5%, which was the highest minimum support used in this experiment.

We also verify the performance of WAVE using the WPortal database. In Figure 5(b) we can observe that in general, for this database, WAVE performs less work for smaller values of minimum support. This trend was observed when the database has a size of 50K transactions, but an interesting result arises for databases with larger sizes as 100K transactions. For smaller values of R^2 , WAVE performs less work for larger values of minimum support, but when we increase the value of R^2 , WAVE performs less work for smaller values of minimum support. The reason is that when the minimum support is too small, a great

number of itemsets present a poor estimate. When the R^2 value is also small, even these poor estimates (not so poor as the R^2 value) are performed. However the relative number of estimates and candidates generated is higher for higher values of minimum support, and, as a consequence, more estimates were performed for higher values of minimum supports. For this database, in all cases, the larger the longevity, the smaller is the work performed by WAVE. Finally, as we can observe in this figure, WAVE performs less work for larger databases.

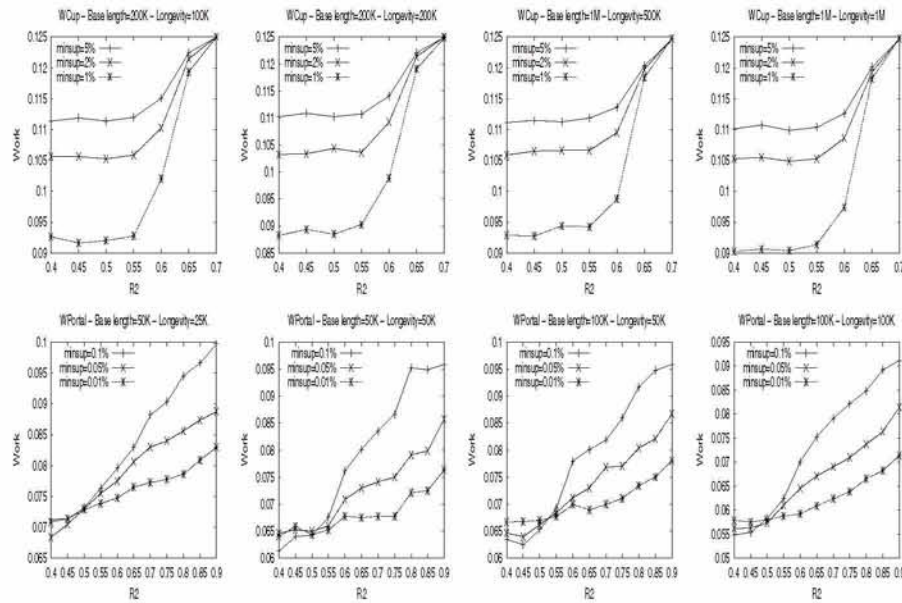


Fig. 5. Work Performed by WAVE when varying *minimum support*, R^2 , *base length*, and *longevity* for a) WCup Database (top row), and b) WPortal Database (bottom row)

4.3 Scalability Experiments

In this section we compare the amount of memory used by WAVE and ULI, when we employ different databases, minimum supports, *base lengths*, *longevities*, and R^2 . Note that the amount of memory used by ULI does not depend on the R^2 employed. From Figure 6(a), where we plot the relative amount of memory used by WAVE and ULI to mine the WCup database, we can observe that in all cases WAVE uses less memory than ULI. The amount of memory used by WAVE exponentially decreases with the R^2 used. This result was expected since for smaller values of R^2 a larger number of estimates are performed. When we

decrease the minimum support value, the relative use of memory also decreases. This is because WAVE is more scalable than ULI, with respect to memory usage. The relative memory usage is smaller when we employ larger *longevities*. Finally, the larger the *base length* used, the less relative memory usage is observed. As can be seen in Figure 6(b), similar results were observed when we used the WPortal database.

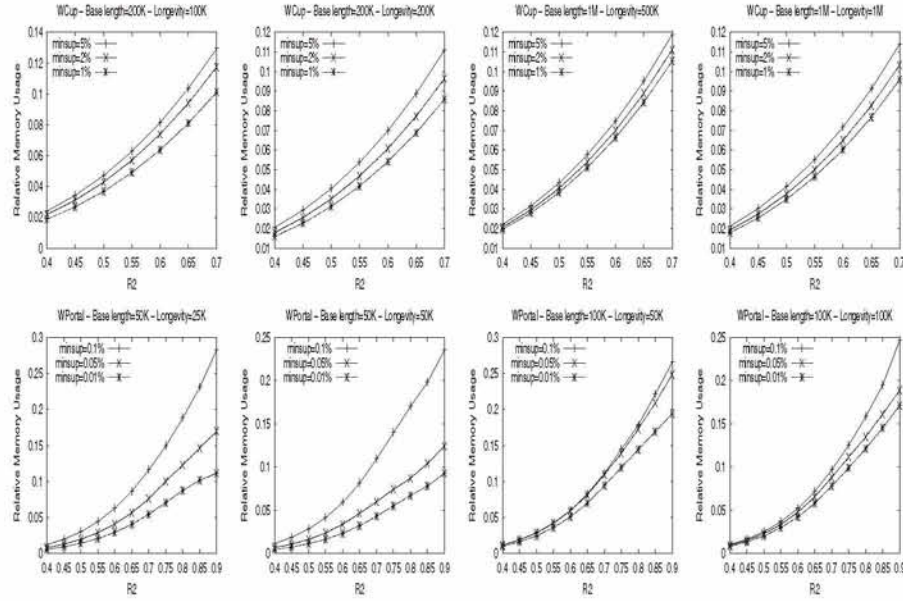


Fig. 6. Relative Memory Usage when varying *minimum support*, R^2 , *base length*, and *longevity* for a) WCup Database (top row), and b) WPortal Database (bottom row)

5 Conclusions and Future Work

This paper introduced WAVE, an algorithm capable of generating highly accurate approximate models of associations in evolving databases. WAVE is able to efficiently maintain the model of associations up-to-date within a tolerance threshold value. The resulting accuracy is similar to what would be obtained by reapplying any conventional association mining algorithm to the entire database. Extensive empirical studies on real and synthetic datasets show that WAVE yields very accurate models while at the same time being space and time efficient.

We plan to apply WAVE to more real-world problems; its ability to do selective updates should allow it to perform very well on a broad range of tasks.

Currently WAVE incrementally maintains the information about the previously frequent itemsets and discards the other ones, but in some domains these recently infrequent itemsets may become useful down the line – identifying such situations based on trend detection and taking advantage of them is another interesting direction for future work.

References

1. R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. of the 20th Int'l Conf. on Very Large Databases*, SanTiago, Chile, June 1994. 436
2. D. Cheung, J. Han, V. Ng, and C. Y. Wong. Maintenance of discovered association rules in large databases: An incremental updating technique. In *Proc. of the 12th Intl. Conf. on Data Engineering*, February 1996. 436, 437, 438
3. D. Cheung, K. Hu, and S. Xia. Asynchronous parallel algorithm for mining association rules on a shared-memory multiprocessors. In *ACM Symposium on Parallel Algorithms and Architectures*, pages 279–288, 1998. 436
4. D. Cheung, S. Lee, and B. Kao. A general incremental technique for maintaining discovered association rules. In *Proc. of the 5th Intl. Conf. on Database Systems for Advanced Applications*, pages 1–4, April 1997. 436, 437, 438
5. V. Ganti, J. Gehrke, and R. Ramakrishnan. Demon: Mining and monitoring evolving data. In *Proc. of the 16th Int'l Conf. on Data Engineering*, pages 439–448, San Diego, USA, May 2000. 437, 438
6. K. Gouda and M. Zaki. Efficiently mining maximal frequent itemsets. In *Proc. of the 1st IEEE Int'l Conference on Data Mining*, San Jose, USA, November 2001. 436, 437
7. J. Han, H. Jamil, Y. Lu, L. Chen, Y. Liao, and J. Pei. Dna-miner: A system prototype for mining dna sequences. In *Proc. of the 2001 ACM-SIGMOD Int'l. Conf. on Management of Data*, Santa Barbara, CA, May 2001. 436
8. J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *Proc. of the ACM SIGMOD Int'l Conf. on Management of Data*, May 2000. 436
9. C. Kamath. On mining scientific datasets. In et al R. L. Grossman, editor, *Data Mining for Scientific and Engineering Applications*, pages 1–21. Kluwer Academic Publishers, 2001. 436
10. S. Lee and D. Cheung. Maintenance of discovered association rules: When to update? In *Research Issues on Data Mining and Knowledge Discovery*, page March, 1997. 437
11. H. Mannila and H. Toivonen. Levelwise search and borders of theories in knowledge discovery. In *Technical Report TR C-1997-8*, U. of Helsinki, January 1997. 437
12. S. Parthasarathy, M. Zaki, M. Ogihara, and S. Dwarkadas. Incremental and interactive sequence mining. ACM Conference on Information and Knowledge Management (CIKM), Mar 1999. 437
13. S. Parthasarathy, M. Zaki, M. Ogihara, and W. Li. Parallel data mining for association rules on shared-memory systems. In *Knowledge and Information Systems*, Santa Barbara, CA, February 2001. 436
14. M. Rajman and R. Besan. Text mining - knowledge extraction from unstructured textual data. In *Proc. of the 6th Int'l Conf. Federation of Classification Societies*, pages 473–480, Roma, Italy, 1998. 436
15. S. Thomas, S. Bodagala, K. Alsabti, and S. Ranka. An efficient algorithm for the incremental updation of association rules. In *Proc. of the 3rd Int'l Conf. on Knowledge Discovery and Data Mining*, August 1997. 436, 437, 438, 442

16. A. Veloso, W. Meira Jr., M. B. de Carvalho, B. Pôssas, S. Parthasarathy, and M. Zaki. Mining frequent itemsets in evolving databases. In *Proc. of the 2nd SIAM Int'l Conf. on Data Mining*, Arlington, USA, May 2002. 436, 437, 438
17. A. Veloso, B. Rocha, W. Meira Jr., and M. de Carvalho. Real world association rule mining. In *Proc. of the 19th British National Conf. on Databases (to appear)*, July 2002. 437
18. M. Zaki and C. Hsiao. Charm: An efficient algorithm for closed itemset mining. In *Proc. of the 2nd SIAM Int'l Conf. on Data Mining*, Arlington, USA, May 2002. 436, 437
19. M. Zaki, S. Parthasarathy, M. Ogihara, and W. Li. New algorithms for fast discovery of association rules. In *Proc. of 3rd Int'l Conf. Knowledge Discovery and Data Mining*, August 1997. 436
20. M. Zaki, S. Parthasarathy, M. Ogihara, and W. Li. New parallel algorithms for fast discovery of association rules. *Data Mining and Knowledge Discovery: An International Journal*, 4(1):343–373, December 1997. 436

Explaining Predictions from a Neural Network Ensemble One at a Time

Robert Wall, Pádraig Cunningham, and Paul Walsh

Department of Computer Science, Trinity College Dublin

Abstract. This paper introduces a new method for explaining the predictions of ensembles of neural networks on a case by case basis. The approach of explaining individual examples differs from much of the current research which focuses on producing a global model of the phenomenon under investigation. Explaining individual results is accomplished by modelling each of the networks as a rule-set and computing the resulting coverage statistics for each rule given the data used to train the network. This coverage information is then used to choose the rule or rules that best describe the example under investigation. This approach is based on the premise that ensembles perform an implicit problem space decomposition with ensemble members specialising in different regions of the problem space. Thus explaining an ensemble involves explaining the ensemble members that best *fit* the example.

1 Introduction

Neural networks have been shown to be excellent predictors. In many cases their prediction accuracy exceeds that of more traditional machine learning methods. They are, however, unstable. This means that although two networks may be trained to approximate the same function, the response of both neural networks to the same input may be very different. Ensembles of networks have been used to counteract this problem. An ensemble comprises a group of networks each trained to approximate the same function. The results of executing each of these networks is then combined using a method such as simple averaging [3] in the case of regression problems, or voting in the case of classification problems. Ensembles used in this way show great promise not only in increasing the stability but also the accuracy of neural networks. The more diverse the members of the ensemble, the greater the increase in accuracy of the ensemble over the average accuracy of the individual members [6].

A further problem with neural networks is their ‘black box’ like nature. Users are not able to interpret the complex hyperplanes that are used internally by the network to partition the input space. A neural network may prove to be a better predictor for a particular task than alternative interpretable approaches but it is a black box. Therefore, substantial research has been done on the problem of translating a neural network from its original state into alternative more understandable forms. However, despite the obvious advantages of ensembles, much

less work has been done on the problem of translating ensembles of networks into more understandable forms.

Zenobi & Cunningham [14] argue that the effectiveness of ensembles stems in part from the ensemble performing an implicit decomposition of the problem space. This has two consequences for explaining ensembles. First it implies that a comprehensible model of the ensemble may be considerably more complex than an individual network. But more importantly, it means that parts of the ensemble will be irrelevant in explaining some examples.

Due to the increased complexity of ensembles, the objective of producing a global explanation of the behaviour of the ensemble is very difficult to achieve. So the goal of our research is focused on explaining specific predictions - a goal that is achievable for ensembles. Whereas, in this paper we concentrate on explaining ensembles of neural networks, our approach can be applied to any ensemble where outputs of an ensemble member can be explained by rules. This local, rather than global, approach to explanation is further elaborated in the next section. A brief introduction to the types of neural networks investigated is given in section 3.2. The behaviour of individual networks is modelled using rules derived from a decision tree that is built to model the outputs of an individual neural network, this is discussed in section 3.3. A method for selecting the most predictive of these rules for any given case is then presented in section 3.4. Also included in section 3.5 are some comments on how different policies may be used in different circumstances depending on the user of the system. Finally, section 4 includes an evaluation of the results with comments from an independent expert in the area of study.

2 Explanation

Explanation is important in Machine Learning for three reasons:

- to provide insight into the phenomenon under investigation
- to explain predictions and thus give users confidence
- to help identify areas of the problem space with poor coverage, allowing a domain expert to introduce extra-examples into the training set to correct poor rules

The first of these objectives is 'Knowledge Discovery' and can be achieved by producing a global model of the phenomenon. This global model might be a decision tree or a set of rules. Since Machine Learning techniques are normally used in *weak theory* domains it is difficult to imagine a scenario where such a global model would *not* be of interest. The second objective is more modest but we argue is adequate in a variety of scenarios. In the next two subsections we discuss why producing global explanations of ensembles is problematic and consider situations where local (i.e. example oriented) explanation is adequate.

2.1 Explaining Neural Networks

Many domains could benefit greatly from the prediction accuracy that neural networks have been shown to possess. However, because of problems with the black-box nature of neural networks (particularly in domains such as medical decision support), there is a reluctance to use neural networks. Capturing this prediction accuracy in a comprehensible format is behind the decision to generate rules based on neural networks in this research.

Most of the work on explaining neural networks has focused on extracting rules that explain them; a review of this work is available in [13]; a more in depth discussion of specific methods is available in [1].

The research on rule extraction can be separated into two approaches, direct decomposition and black box approaches. In a direct decomposition approach interpretable structures (typically trees or rules) are derived from an analysis of the internal structure of the network. With black box approaches, the internals of the network are not considered, instead the input/output behaviour of the network is analysed (see section 3.3). Clearly, the first set of techniques is architecture-specific while the black-box approaches should work for all architectures. The big issue with these approaches is the fidelity of the extracted rules; that is, how faithful the rule-set behaviour is to that of the net.

2.2 Explaining Ensembles

For the black-box approaches described in the previous section the contents of the black-box can be an *ensemble* of neural networks, as easily as a single neural net.

Domingos [8] describes a decision tree-based rule extraction technique that uses the ensemble as an oracle to provide a body of artificial examples to feed a comprehensible learner. Craven and Shavlik [5] describe another decision tree-based rule extraction technique that uses a neural network as an oracle to provide a body of artificial examples to feed a comprehensible learner. Clearly, this technique would also work for an ensemble of neural networks.

The big issue with such an approach is the *fidelity* of the extracted rules; that is, how closely they model the outputs of the ensemble. Craven and Shavlik report fidelity of 91% on an elevator control problem. Emphasising the importance of the ensemble, Domingos reports that his technique preserves 60% of the improvements of the ensemble over single models. He reports that there is a trade-off between fidelity and complexity in the comprehensible models generated; models with high fidelity tend to be quite complex. It is not surprising that comprehensible models that are very faithful to the ensemble will be very complex; and thus less comprehensible.

2.3 Global versus Local Explanation

The focus of this paper is on explaining predictions on a case by case basis. This is different to the current thrust of neural network explanation research.

One author who has also taken this approach is Sima [12] and his approach is reviewed by Cloete and Zurada [4]. Local explanations of time-series predictions have also been explored by Das et al. [7].

Most other researchers have focused on producing global model explanations. These models aim to fully describe all situations in which a particular event will occur. Although this may be useful in many situations, it is argued here that it is not always appropriate. For example, it may be useful in the problem of predicting success in IVF (in-vitro fertilisation) research for instance, studied by Cunningham et al. [6], to produce a global model of the phenomenon. Such a model would allow practitioners to spend time understanding the conditions leading to success and to focus their research on improving their techniques. Also, a global model would allow the targeting of potential recipients of the treatment who have a high probability of success. This would lead to a monetary saving for the health service and would avoid great disappointment for couples for whom the treatment would most likely fail. A global model might also allow doctors to suggest changes a couple might make in order to improve their chances of success with the treatment.

In the accident and emergency department of a busy hospital, the explanation requirement would be quite different. Here the need is for decision support rather than knowledge discovery. What is needed is an explanation of a decision in terms of the symptoms presented by individual patients.

3 System Description

3.1 Datasets

Two datasets were used in the analysis presented in this paper. Since the objective of the research is to produce explanations of predictions the main evaluation is done on some Bronchiolitis data for which we have access to a domain expert. This data relates to the problem of determining which children displaying symptoms of Bronchiolitis should be kept in hospital overnight for observation. This data set comprising 132 cases has a total of 22 features, composed of 10 continuous and 12 symbolic and a single binary output reflecting whether the child was kept overnight or not.

In order to provide some insight into the operation of the system we also include some examples of explanations for the Iris data-set [2]. This is included to show graphically the types of rules that are selected by the system.

3.2 Neural Networks

The neural networks used in this system are standard feed-forward networks trained using backpropagation. It is well known that although neural networks can learn complex relationships, they are quite unstable. They are unstable in the sense that small changes in either the structure of the network (i.e. number of hidden units, initial weights etc.) or in the number of training data may lead

to quite different predictions from the network. An effective solution is to use a group (ensemble) of networks trained to approximate the same function, and to aggregate the outputs of the ensemble members to produce a prediction [6,9].

One technique for dividing the data and combining the networks is bagging [3] (short for bootstrap aggregating). This involves randomly selecting examples with replacement from the full set of data available for training. If the size of these bootstrap sets is the same as the full training set, roughly a third of the examples will not be selected at all for each individual sample. These remaining samples can be used as a validation set to avoid overfitting the network to the data. For regression tasks Breiman [3] simply takes the average of the individual network outputs as the ensemble output. For the classifications tasks used in this evaluation, majority voting is used to determine the ensemble prediction.

Ensembles have the added benefit that in reducing the instability of networks the prediction performance is also improved by averaging out any errors that may be introduced by individual networks. The more unstable the networks, the more diverse the networks and thus the greater the improvement of the ensemble over the accuracy of the individual networks.

3.3 Rule Extraction

The approach to explaining ensembles of neural networks that we describe here involves extracting rules from the individual networks in the ensemble, finding the rules that contribute to the prediction and selecting the rules that best fit the example. The approach we use for rule extraction is a fairly standard black-box approach - similar to that used by Domingos [8]. One major difference between our approach and that of Domingos is that Domingos built a single tree based on the results of using the ensemble as an oracle. We also implemented this solution and compared it with our approach; the results of both approaches are included in the evaluation. Our rule extraction process uses the neural networks as *oracles* to train a decision trees using C4.5 [10]. C4.5Rules is then used to extract rules from this decision tree. The steps are as follows:

1. Generate artificial data by small perturbations on the available training data.
2. Use the neural network to predict an output (i.e. label) for this data.
3. Use this labeled data to train a C4.5 decision tree.
4. Extract rules from this decision tree using C4.5Rules

This yields a set of rules that model the neural net with reasonable fidelity. This number of rules actually produced can be controlled by setting the pruning parameter in the process of building the tree.

3.4 Rule Selection

After training an ensemble of networks and building decision trees to model the behaviour of the individual networks we are left with a group of rule-sets, one for each network. The task then is to find the most predictive of these rules for a given input. This is accomplished by executing the following steps:

- Apply each of the rule-sets to the example to produce a prediction from that rule-set
- The rule-sets vote among themselves to decide the overall ensemble prediction
 - Any rule-set that did not vote for this predicted outcome is now discarded
 - Rules that did not vote for the winning prediction within the remaining rule-sets are also discarded
 - This leaves only rules that contributed to the winning prediction

It is from this subset of relevant rules that the most relevant rules will be chosen. In order to select the most relevant rules, it is first necessary to know some statistics about these rules. These are computed after initially producing each rule set.

Rule Coverage Statistics After producing each rule-set, it is necessary to propagate each data item in the set of data used to train the network through each rule. If a rule fires for a particular example and both the example and the rule have the same target, then this example is saved with the rule. The number of examples saved with the rule is considered to be the coverage for that rule.

However, it is possible to go beyond a simple coverage figure. This is done by analysing the individual rule antecedents with respect to the examples covered. For each antecedent in the rule that tests a numeric feature, the mean and standard deviation of the values of that feature contained within the examples covered by that rule can be calculated. This is shown graphically for a single feature in Figure 1. For antecedents testing symbolic features, a perfect fitness score is automatically assigned since any example firing that rule must by definition have the value of that symbolic feature.

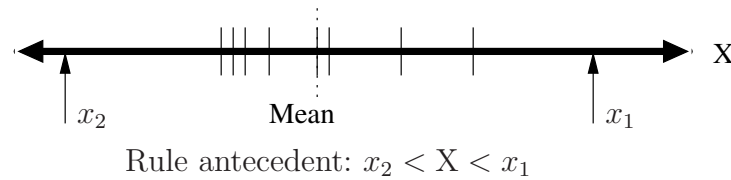


Fig. 1. Number line showing limit of rule antecedent test and several values from examples that fired this rule

Having calculated the above statistics for each of the antecedents, it is now possible to calculate the "fit" of future unseen examples to each of the rules.

Firstly a fit is calculated for each of the numeric features in the rule. This is calculated using equation 1.

$$\text{Fit}_X = \left| \frac{x - \mu}{\sigma} \right| \quad (1)$$

The antecedent with the maximum (i.e. poorest) fitness score is then selected as the fit for the rule as a whole. This is similar to the approach taken in MYCIN [11] as shown in equation 2.

$$\text{MB}[h_1 \wedge h_2, e] = \min(\text{MB}[h_1, e], \text{MB}[h_2, e]) \quad (2)$$

In this case the measure of belief(MB) in two terms in conjunction in a rule would be the MB of the weaker term.

Finally, basing the fitness on the distance from the mean is not appropriate in situations where a term is only limited on one side (e.g the first example in section 4.1). In those situations, an example with a feature value on the far side of the mean to the limit is given the maximum fitness, i.e. it is considered to fit the rule well.

3.5 Rule Selection Policies

This fitness measure gives us our main criteria for ranking rules and, so far, has proved quite discriminating in examples examined. However in the Bronchiolitis scenario (see section 4.2) ties can occur when a group of rules all have maximum fitness. Ties can be resolved in these situations by considering rule specificity, i.e. the number of terms in the rule. In situations where simple explanations are preferred, rules with few terms are preferred. In situations where elaborate explanations might be interesting rules with more terms in the left-hand-side can be ranked higher.

The doctor examining the results of the Bronchiolitis data suggested that, in practice, simple explanations might be appropriate for holding a patient overnight whereas more elaborate explanations might be necessary for discharge. The logic behind this is that a single symptom might be enough to cause concern about a child whereas to discharge a child no adverse symptoms should be observable.

So in selecting and ranking rules to explain the Bronchiolitis data the main criterion was the ranking based on the rule fit. Then ties were resolved by selecting the most simple rules for admissions and the most complex rules for discharges. This produced very satisfactory results.

4 Evaluation

Evaluation of this research is not straightforward. To appreciate the quality of the suggested rules, it is necessary to have a good understanding of the domain under investigation. For this reason, the results generated for the Bronchiolitis dataset were given to an expert in this area and his opinions are recorded in section 4.2.

For each of the datasets a total of ten examples were held back from training of the networks and used for evaluation only. For each one of these examples the five most predictive rules were chosen. Also included in the results was a second

set of rules selected from rules that were extracted from a decision tree that was trained to model the behaviour of the vote over the ensemble of networks. This second set of results was included as a comparison to see if the system could select more accurate rules given the more diverse rule-sets of the ensemble members.

4.1 Iris Dataset

In order to offer some insight into the operation of the system, we can show some examples of it in operation on the Iris dataset [2]. The Iris data contains three classes and four numeric features so the rules are much simpler than those produced for Bronchiolitis. This dataset is so simple in fact that the fidelity results are close to perfect. In order to make the problem somewhat more difficult (and to produce more diverse ensemble members) the total number of examples for each class was cut from 50 to 20. A plot of the training data in two dimensions is shown in Figure 2.

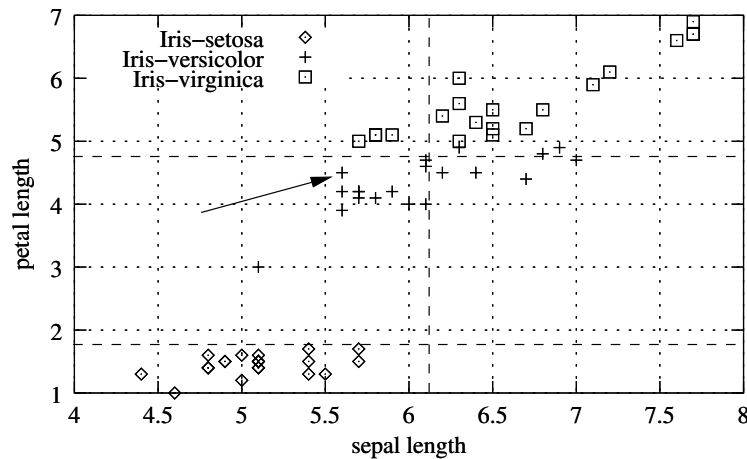


Fig. 2. Iris data plotted in two dimensions

From Figure 2 we can see how the two following rules were selected to explain two different examples. The number in the square brackets preceding the rules is the fit for the example for that rule. The first rule classifies an Iris-setosa. The zero fit indicates that the example being tested fell on the far side of the mean for the rule and hence was given a maximum fit as described in section 3.4. The second rule classifies the example indicated by the arrow in the figure. It has a fit of 0.76 because this example is actually quite close to the limit for the second term for that rule. The example fits the first term well but the poorer fitness is

chosen as the overall fitness for the rule. Nevertheless this rule was selected as the best rule from an ensemble of 9 members.

```
[0.000000]
IF petal_length <= 1.874200
  THEN Iris-setosa

[0.759346]
IF sepal_length <= 6.120790
  AND 1.874200 < petal_length <= 4.797420
  THEN Iris-versicolor
```

4.2 Bronchiolitis Dataset

In the case of the Bronchiolitis dataset Paul Walsh, a doctor in Crumlin Children's Hospital Dublin and the original provider of the data wrote a response to each of the selected rules for each of the tested examples.

Before analysing some of the comments of the expert, some statistics are included in Table 1 describing important characteristics of the network and rules. These statistics were calculated using 10 fold cross validation with an ensemble of five neural networks per fold. The accuracies for each network and its associated extracted rules were calculated on the remaining data in each fold. The fidelity between the network and rule results was also calculated. Finally the results from all the individual networks in a single fold were combined using voting to produce an accuracy for the ensemble.

Table 1. Average and standard deviation figures for the accuracy and fidelity using 10-fold cross validation on Bronchiolitis data

Average Ensemble Accuracy	73% $\pm$ 9
Average Network Accuracy	69% $\pm$ 12
Average Rule-set Accuracy	67% $\pm$ 12
Average Network/Rules Fidelity	84% $\pm$ 9

The fidelity result in Table 1 is of particular interest. The fidelity figure is a measure of how well the rules actually model the network behaviour. This measure is estimated by executing both the network and the rules with all the data. The fidelity is the percentage of times the rule-sets agree with the associated network. Clearly it is important that this figure be as high as possible.

The results in Table 1 show that we get a reasonable improvement in accuracy from using ensembles in the Bronchiolitis dataset. We also get increased stability, the individual network results in the ensembles varied more than the ensemble results.

In general many of the rules considered 'excellent' in the sets presented to the expert were among the first presented(i.e. those with the greatest fitness). From this it would appear that the fitness criterion was effective in selecting good rules. An example of one of these rules is included below:

```
IF Dehydration = None
  AND Retractions = None
  AND 92.397100 < SaO_2_2
  AND BS <= 0.358798
  THEN DISCHARGE
```

In the rule above there are two tests on numeric features and two on symbolic features. The fitness is influenced by the numeric features since the fitness on symbolic features will be 'perfect' if the rule applies to the example.

In more detail, the domain expert was asked to examine and rate rules explaining 10 examples. At most five ranked rules were presented to the expert as good explanations for the prediction(some examples had fewer than five rules that fired from all rule-sets in the ensemble). In addition to these ranked rules, rules that comprised solely of antecedents testing symbolic features were also presented. Rules comprising symbolic features only will automatically have perfect fitness. In total there 60 rules were presented to the expert to explain the 10 examples and 90% of these were correct explanations.

A very small number of rules were marked as definitely being incorrect(4 examples had wrong rules). Of the remaining rules, all contributed in aiding the explanation of the prediction according to the expert. In just eight rules, one of the antecedents in the rule selected did not add much to the knowledge contained within the rule, although the rule as a whole was still considered acceptable. Of the remaining rules, six were marked as excellent, indicating that those rules described almost exactly the published criteria for decision making and covered all of the most important features in a single rule.

By comparison, the rules selected from the set of rules derived from the decision tree built to model the ensemble as a whole were less useful. There were only seventeen rules in total(there were far fewer rules in the single ruleset to select from) and of these only two were marked as excellent and three contained wrong or misleading antecedents.

It is interesting to note that in both cases above, the rules with antecedents comprising tests of only symbolic features are only once marked as excellent and once marked as wrong. Most of the rest of these rules were described as "common sense" by the expert.

A final note of interest is that for predictions to send a child home, the explanations given were consistently more accurate(this was true for both the rules selected from the individual networks and from the rules derived from the ensemble targets). This could be due to the fact that any child that goes home, is more likely to display very well defined symptoms. While a child whose symptoms may not have reached critical levels is admitted because the doctor knows through intuition that the child will soon display those levels.

4.3 General Observations on Medical Datasets

Some final points should be noted about medical datasets in general that could lead to skewed results:

- Subjective features may exist where the opinions of those collecting the data may have differed.
- Several of the examples in the dataset may have been influenced by environmental factors that cannot be expressed in the data and may have been responsible for a prediction that otherwise wouldn't normally be the case. For example, in the Bronchiolitis scenario there might be concern about the home environment into which a child might be discharged.

Both of these facts are, for all practical purposes, unavoidable in medical datasets and present a particular challenge to the researcher during their analysis.

5 Conclusions & Future Work

These results encourage us that explanations built from rules derived from component neural networks will be more insightful than rules derived from the ensemble as a whole. This work was based on the hypothesis that the effectiveness of ensembles depends on members of the ensemble specialising in different regions of the problem space. Thus, an explanation of a prediction of an ensemble for an individual example needs to seek out this specialising member. Explanations based on viewing the ensemble as a black-box will be more bland. This preliminary evaluation seems to support this.

The process of rule ranking based on the fitness criterion described here is not yet a complete solution. Problems still exist for rules that have only symbolic features since these will automatically get maximum fitness and this will often not be appropriate. There is also the potential for rules with numeric features to have maximum fitness as explained in section 4; however, the use of antecedent specificity as a further criterion to address this issue shows promise.

It became clear during the evaluation that features that were not strongly predictive were turning up in rules where they were not useful. Because of this we propose to precede the whole process with a feature selection process to weed out poorly predictive features. In general, it seems to us wise to precede any explanation exercise with feature selection since it will relieve the explanation process of the burden of accounting for features that are not very relevant.

References

1. Andrews, R., Diederich, J. & Tickle A. (1995) A Survey and Critique of Techniques For Extracting Rules From Trained Artificial Neural Networks, *Knowledge Based Systems* 8, pp373-389. 451
2. Blake, C. L. & Merz, C. J. (1998). UCI Repository of machine learning databases [<http://www.ics.uci.edu/mllearn/MLRepository.html>]. Irvine, CA: University of California, Department of Information and Computer Science. 452, 456

3. Breiman, L., (1996) Bagging predictors, *Machine Learning*, 24:123-140. 449, 453
4. Cloete, I., & Zurada J. M., (2000) *Knowledge Based Neurocomputing*(MIT Press, Cambridge, Massachusettes). 452
5. Craven, M., & Shavlik, J., (1999) Rule Extraction: Where Do We Go from Here?, University of Wisconsin Machine Learning Research Group Working Paper, 99-1. 451
6. Cunningham, P., Carney, J., Jacob, S., (2000) Stability Problems with Artificial Neural Networks and the Ensemble Solution, *AI in Medicine*, pp217-225, Vol. 20, No. 3. 449, 452, 453
7. Das G., Lin K., Mannila H., Renganathan G., Smyth P., (1998) Rule Discovery from Time Series, *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, (AAAI Press). 452
8. Domingos P., (1998) Knowledge Discovery via Multiple Models, *Intelligent Data Analysis*, 2, 187-202. 451, 453
9. Hanson, L. K., Salomon, P., (1990) Neural Network Ensembles, *IEEE Pattern Analysis and Machine Intelligence*, 1990. 12, 10, 993-1001. 453
10. Quinlan, J. Ross., (1988) *C4.5 Programs for Machine Learning*(Morgan Kaufmann Publishers Inc., San Mateo, CA). 453
11. Shortliffe, E. H., (1976) Computer-Based Medical Consultations: MYCIN, New York, Elsevier. 455
12. Sima, J., (1995) Neural Expert Systems, *Neural Networks* 8(2) pp261-271. 452
13. Wall, R., Cunningham, P., (2000) Exploring the Potential for Rule Extraction from Ensembles of Neural Networks, 11th Irish Conference on Artificial Intelligence & Cognitive Science (AICS 2000), J. Griffith & C. O’Riordan (eds.) pp52-68 (also available as Trinity College Dublin Computer Science Technical Report TCD-CS-2000-24). 451
14. Zenobi G., & Cunningham P., (2001), Using Diversity in Preparing Ensembles of Classifiers Based on Different Feature Subsets to Minimize Generalisation Error, *12th European Conference in Machine Learning (ECML 2001)*, eds L. De Raedt & P. Flach, LNAI 2167, pp576-587, Springer Verlag. 450

Structuring Domain-Specific Text Archives by Deriving a Probabilistic XML DTD

Karsten Winkler* and Myra Spiliopoulou

Leipzig Graduate School of Management (HHL), Department of E-Business
Jahnallee 59, D-04109 Leipzig, Germany
{kwinkler,myra}@ebusiness.hhl.de
<http://ebusiness.hhl.de>

Abstract. Domain-specific documents often share an inherent, though undocumented structure. This structure should be made explicit to facilitate efficient, structure-based search in archives as well as information integration. Inferring a semantically structured XML DTD for an archive and subsequently transforming its texts into XML documents is a promising method to reach these objectives. Based on the KDD-driven DIASDEM framework, we propose a new method to derive an archive-specific structured XML document type definition (DTD). Our approach utilizes association rule discovery and sequence mining techniques to structure a previously derived flat, i.e. unstructured DTD. We introduce the notion of a probabilistic DTD that is derived by discovering associations among and frequent sequences of XML tags, respectively.

1 Introduction

Up to 80% of a company's information is stored in unstructured textual documents. Hence, *document warehousing* and *text mining* are emerging disciplines for capturing and exploiting the flood of textual information for decision making [1]. However, acquiring interesting and actionable knowledge from textual databases is still a major challenge for the data mining community. Creating semantic markup is one form of providing explicit knowledge about text archives to facilitate search and browsing or to enable information integration with related data sources. Unfortunately, most users are not willing to manually create meta-data due to the efforts and costs involved [2]. Thus, text mining techniques are required to (semi-) automatically create semantic markup.

In this paper, we describe a KDD methodology for establishing a quasi-schema in the form of a probabilistic XML document type definition (DTD). This work is pursued in the research project DIASDEM that focuses on text archives with domain-specific vocabulary and syntax. The DIASDEM framework for semantic tagging of domain-specific texts was introduced in [3,4].

* The work of this author is funded by the German Research Society (DFG grant: SP 572/4-1) within the research project DIASDEM. The German acronym stands for "Data Integration for Legacy Data and Semi-Structured Documents by Means of Data Mining Techniques".

Currently, the Java-based DIAsDEM Workbench derives a preliminary, flat and unstructured XML DTD from an archive of semantically tagged XML documents. However, we ultimately aim at integrating these XML documents with related, structured data sources. In this context, the derived list of XML tags should be transformed into a schema. As a first step, we derive an archive-specific *probabilistic DTD* from these tags, which (i) describes the most likely orderings of elements and (ii) adorns each element with statistical properties. We define a probabilistic DTD as a graph-based data structure describing the structural properties of the corresponding XML archive. Our future work consists of using this probabilistic DTD for the derivation of an archive-specific XML Schema and a relational schema, respectively. We introduce an algorithm for inferring a probabilistic DTD that utilizes association rule discovery algorithms and sequence mining techniques.

The rest of this paper is organized as follows: The next section discusses related work. Section 3 provides a concise presentation of the original DIAsDEM approach to semantic tagging. In section 4, we introduce the notion of probabilistic DTDs for textual archives and develop a method for deriving them. Section 5 summarizes a real-world case study. Finally, we conclude and give directions for future research.

2 Related Work

Concerning related knowledge discovery work [5,6,7], our approach shares with this research thread the objective of extracting semantic concepts from texts. However, concepts to be extracted in DIAsDEM must be appropriate to serve as XML DTD elements. Among other implications, discovering a concept that is peculiar to a single text unit is not sufficient for our purposes, although it may perfectly reflect the corresponding content. To derive a DTD, we need to discover groups of text units that share semantic concepts. Moreover, we concentrate on domain-specific texts, which significantly differ from average texts with respect to word frequency statistics. These archives can hardly be processed using standard text mining software, because the integration of domain knowledge is a prerequisite for successful knowledge discovery.

Currently, there are only a few research activities aiming at the transformation of texts into semantically annotated XML documents: Bruder et al. introduce the search engine GETESS that supports query processing on texts by creating and processing XML text abstracts [8]. These abstracts contain language-independent, content-weighted summaries of domain-specific texts. In DIAsDEM, we do not separate meta-data from original texts but rather provide a semantic annotation, keeping the texts intact for later processing or visualization. Additionally, the GETESS approach requires an a priori given DTD that corresponds to a domain-specific ontology. Erdmann et al. introduce a system that supports the semi-automated and ontology-based semantic annotation of Web pages [2]. The authors associate previously extracted text fragments

(mostly named entities) with concepts of an a priori given ontology. In contrast, DIAsDEM aims at deriving an XML DTD from unstructured text documents.

To transform existing contents into XML documents, Sengupta and Purao propose a method that infers DTDs by using already tagged documents as input [9]. In contrast, we propose a method that tags plain text documents and derives a DTD for them. Moore and Berman present a technique to convert textual pathology reports into XML documents [10]. In contrast to our work, the authors neither derive an XML DTD nor apply a knowledge discovery methodology. They rather employ natural language processing techniques and a medical thesaurus to map terms and noun groups onto medical concepts. Thereafter, medical concepts serve as XML tags that semantically annotate the corresponding terms. Closer to our approach is the work of Lumera, who uses keywords and rules to semi-automatically convert legacy data into XML documents [11]. However, his approach relies on establishing a rule base that drives the conversion, while we use a KDD methodology that reduces necessary human intervention.

Semi-structured data is an area of related database research [12]. A lot of effort has recently been put into methods inferring and representing structure in similar semi-structured documents [13,14]. However, these approaches only derive a schema for a given set of semi-structured documents. Given a collection of marked up semi-structured texts, some authors employ grammatical inference techniques to create an XML DTD [15,16]. In contrast to our approach, these authors infer a probabilistic DTD for manually marked up XML or SGML documents. In DIAsDEM, we have to simultaneously solve the problems of both semi-structuring texts by semantic tagging and inferring an appropriately structured XML DTD.

3 The DIAsDEM Framework

Our work on creating an archive-specific and probabilistic XML DTD is based on the DIAsDEM framework for semantic tagging of document archives with domain-specific XML tags [3,4].

In DIAsDEM, the notion of semantic tagging refers to annotating texts with domain-specific XML tags that can have attributes describing named entities (e.g., names of persons). Normally, a document consists of many structural text units such as sentences or paragraphs. Hence, rather than classifying entire documents or tagging single terms, we aim at semantically annotating these structural text units in order to make their semantics explicit. The following excerpt illustrates two tagged sentences contained in a German Commercial Register entry, whereas each sentence corresponds to a text unit:

```
<BusinessPurpose> Der Betrieb von Spielhallen in Teltow und das Aufstellen von
Geldspiel- und Unterhaltungsautomaten. </BusinessPurpose>
<AppointmentManagingDirector Person="Balski; Pawel"> Pawel Balski ist zum
Geschäftsführer bestellt. </AppointmentManagingDirector>
```

Semantic tagging in DIAsDEM is a two-phase process: In its first phase, our proposed knowledge discovery in textual databases (KDT) process discovers clusters of semantically similar text units, tags documents in XML according to the results and derives an XML DTD describing the archive-specific document structure. The KDT process results in a final set of clusters whose labels serve as XML tags and DTD elements. Huge amounts of new documents can be converted into XML documents in the second, batch-oriented and productive phase of the DIAsDEM framework.

Besides the initial text documents to be tagged, the following domain knowledge constitutes input to our KDT process: A thesaurus containing a domain-specific taxonomy of terms and concepts, a preliminary conceptual schema of the domain and descriptions of specific named entities, e.g. persons and companies. The conceptual domain schema reflects the semantics of named entities and the relationships among them, as they are initially conceived by application experts. This schema might serve as a reference for the DTD to be derived from discovered semantic tags, but there is no guarantee that the final DTD will be contained in or will contain this schema.

Similarly to a conventional KDD process, our process starts with a preprocessing phase that includes basic NLP preprocessing tasks such as tokenization, normalization and word stemming as well as named entity extraction. Instead of removing stop words, we establish a drastically reduced feature space by selecting a limited set of terms and concepts (so-called text unit descriptors) from the thesaurus and the conceptual schema. Text unit descriptors are currently chosen by the knowledge engineer because they must reflect important concepts of the application domain. All text units are mapped onto Boolean vectors of this feature space. Thereafter, the Boolean text unit vectors are further processed by applying an information retrieval weighting schema (i.e. TF-IDF).

In the pattern discovery phase, all text unit vectors contained in the initial archive are clustered based on content similarity. The objective is to discover dense and homogeneous text unit clusters. Clustering is performed in multiple iterations. Each iteration outputs a set of clusters, which is partitioned into “acceptable” and “unacceptable” ones according to our quality criteria. A cluster of text unit vectors is qualitatively “acceptable”, if and only if (i) its cardinality is large and the corresponding text units are (ii) homogeneous and (iii) can be semantically described by a small number of text unit descriptors. Members of “acceptable” cluster are subsequently removed from the dataset for later labeling, whereas the remaining text unit vectors are input data to the clustering algorithm in the next iteration. In each iteration, the cluster similarity threshold value is stepwisely decreased such that “acceptable” clusters become progressively less specific in content. The KDT process is based on a plug-in concept that allows the execution of different clustering algorithms within the DIAsDEM Workbench.

In the postmining phase, all “acceptable” clusters are semi-automatically assigned a semantic label. The DIAsDEM Workbench performs both a pre-selection and a ranking of candidate cluster labels for the expert to choose from.

The default cluster labels are derived from prevailing feature space dimensions (i.e. text unit descriptors) in each “acceptable” cluster. Cluster labels actually correspond to XML tags that are subsequently used to annotate cluster members. Thereafter, all original documents are annotated using valid XML tags that have attributes reflecting previously extracted named entities and their values. Finally, an unstructured XML DTD is derived that describes the semantic structure of the XML collection by enumerating existing XML tags. The following DTD excerpt was created in a recent case study [4]:

```
<!ELEMENT CommercialRegisterEntry ( #PCDATA | FoundationPartnership |
ShareCapital | AppointmentManagingDirector | (...) | ConclusionArticles |
BusinessPurpose | Owner )* > (...) <!ELEMENT Owner (#PCDATA)>
```

Obviously, this preliminary DTD has two shortcomings: It lacks structure and has no indications of mandatory, optional or interdependent elements. We alleviate these shortcomings by deriving a probabilistic DTD from the archive of XML documents described by this enumeration of tags.

4 Establishing a Probabilistic DTD

Our new method of establishing a probabilistic DTD aims at specifying the most appropriate ordering of XML tags, identifying correlated or mutually exclusive XML tags and adorning each XML tag and each correlation among them with statistical properties. These properties form the basis for reliable query processing, because they determine the expected precision and recall of the query results. In the following, we first introduce the statistical properties of DTD elements, we afterwards describe a methodology for computing these statistics for associated XML tags and sequences of tags. Finally, we apply a heuristic pruning algorithm to derive a structured probabilistic DTD of frequent elements.

4.1 Statistical Properties of Semantic DTD Elements

For the derived XML tags, we are interested in whether they should be observed as mandatory inside the DTD, whether they are associated with other tags and whether their most likely relative position inside the DTD can be assessed. In some application domains, a human expert might identify the mandatory parts of the DTD and specify the ordering of XML tags. However, archive documents may still violate specifications of the expert, either because the authors did not respect the specifications, or, as in our case, there has been no a priori DTD for the archive.

Therefore, we define statistical properties of XML tags, associations of tags and sequences of tags. Thereafter, structure and optionality of DTD elements can be determined on the basis of these property values. In particular, let d be an XML document contained in an XML archive $D = \{d_1, \dots, d_{|D|}\}$. Additionally, let $T = \{t_1, \dots, t_{|T|}\}$ be the set of XML tags contained in the derived XML

DTD, let $\{x, y_1, \dots, y_n\} \in T$ or abbreviated $x, y_1, \dots, y_n \in T$ be a set of $n + 1$ XML tags and $\langle y_1 \cdot \dots \cdot y_n \cdot x \rangle$ or abbreviated $y_1 \cdot \dots \cdot y_n \cdot x$ be a sequence of $n + 1$ adjacent XML tags. The function $Tags(d)$ returns the set of all XML tags contained in d . The function $Seqs(d)$ returns the set of all adjacent sequences of XML tags contained in d . For example, consider document d that consists of three semantically tagged text units, i.e. $t_1 \cdot t_2 \cdot t_1$. In this case, $Tags(d) = \{t_1, t_2\}$ and $Seqs(d) = \{t_1 \cdot t_2 \cdot t_1, t_1 \cdot t_2, t_2 \cdot t_1\}$.

TagSupport of tag x is defined as the relative frequency of tag x among the documents of the archive. It is an indicator of whether this tag is likely to be mandatory:

$$TagSupport(x) = \frac{|\{d \in D | x \in Tags(d)\}|}{|D|}$$

A tag may be mandatory in the entire archive or in a particular subset of documents that are characterized by other, associated tags. We use association rule discovery to identify DTD elements frequently appearing together and define *AssociationConfidence* of tag x with respect to the set of tags $y_1, \dots, y_n$ as follows:

$$AssociationConfidence(y_1, \dots, y_n \rightarrow x) = \frac{|\{d \in D | x, y_1, \dots, y_n \subseteq Tags(d)\}|}{|\{d \in D | y_1, \dots, y_n \subseteq Tags(d)\}|}$$

Similarly to conventional association rule discovery, we need to exclude spurious correlations caused by a very high support of a tag in the entire population. To alleviate this problem, we use lift or improvement of association rules and define *AssociationLift* of tag x given tags $y_1, \dots, y_n$ as follows:

$$AssociationLift(y_1, \dots, y_n \rightarrow x) = \frac{AssociationConfidence(y_1, \dots, y_n \rightarrow x)}{TagSupport(x)}$$

To identify potential orderings of tags in a structured DTD, we perform sequence mining among the XML tags of all documents. On the basis of tag sequences frequently appearing in the archive, we define the *SequenceConfidence* of tag x after a sequence of adjacent tags $y_1 \cdot \dots \cdot y_n$ as follows:

$$SequenceConfidence(y_1 \cdot \dots \cdot y_n \cdot x) = \frac{|\{d \in D | y_1 \cdot \dots \cdot y_n \cdot x \in Seqs(d)\}|}{|\{d \in D | y_1 \cdot \dots \cdot y_n \in Seqs(d)\}|}$$

This definition differs from the conventional statistics known for sequence mining [17], because we are concentrating on adjacent tags, disallowing the occurrence of arbitrary tags in-between. This constraint is necessary for the placement of associated tags in a structured DTD. Conventional sequence miners do not ensure that frequent sequences are comprised of adjacent elements. However, some Web usage miners are capable of distinguishing between adjacent and non-adjacent events [18,19,20].

Analogously to *AssociationLift*, we define the *SequenceLift* of tag x after a sequence of adjacent tags $y_1 \cdot \dots \cdot y_n$ as follows:

$$SequenceLift(y_1 \cdot \dots \cdot y_n \cdot x) = \frac{SequenceConfidence(y_1 \cdot \dots \cdot y_n \cdot x)}{TagSupport(x)}$$

The notion of support holds both for sets of associated tags and for sequences of adjacent tags. Instead of defining two support functions, we introduce the notion of an element “group” g , being either a set of tags $y_1, \dots, y_n$ or a sequence of adjacent tags $y_1 \cdot \dots \cdot y_n$. We define *GroupSupport* for groups of tags as follows:

$$GroupSupport(g) = \frac{|\{d \in D | g \in Tags(d) \cup Seqs(d)\}|}{|D|}$$

For any set of at least two tags, this property assumes one value for the set and as many values as are the perturbations of set members. In the following subsection, we show how the statistical information pertinent to individual tags, to groups of tags groups and to relationships among them is modeled in a seamless way.

4.2 A Graph of Associated Groups of XML Tags

Frequent groups of XML tags are discovered by applying association rule discovery and specialized sequence mining algorithms with appropriate support constraints. A formal data structure is required to represent the results. Additionally, an algorithm should be developed to derive a probabilistic DTD from this data structure. We use a directed “graph of associated groups” whose nodes are individual tags, sequences of adjacent tags or sets of co-occurring tags. Each node is adorned with statistical properties pertinent to its tag and its tag group, respectively. An edge represents either a relationship $y_1, \dots, y_n \rightarrow x$ or $y_1 \cdot \dots \cdot y_n \cdot x$. The groups of nodes $y_1, \dots, y_n$ and $y_1 \cdot \dots \cdot y_n$ are referred to as *source* nodes, whereas x is the *target* node. Similarly to nodes, each edge is adorned with statistics of the order-insensitive or order-sensitive association of tags it represents. Let $V \subseteq T \times (0, 1]$ be the set of graph nodes conforming to the signature:

$$\langle TagName, TagSupport \rangle$$

Note that XML tag x only appears in the graph if $TagSupport(x) > 0$. For groups of tags, we distinguish between order-sensitive and order-insensitive groups by imposing an arbitrary ordering upon groups of tags, e.g. lexicographical ordering. Thereafter, each tag group is observed as an order-sensitive or an order-insensitive list: An order-sensitive list is a sequence of tags, and an order-insensitive list is a set of tags.

More formally, let $P(V)$ be the set of all lists of elements in V , i.e. $(TagName, TagSupport)$ -pairs. A tag group $g \in P(V) \times \{0, 1\}$ has the form $\langle v_1, \dots, v_k \rangle, 1$, where $\langle v_1, \dots, v_k \rangle$ is a list of elements from V and the value 1 indicates that this list represents an order-sensitive group. Similarly, $g' = \langle v_1, \dots, v_k \rangle, 0$ would represent the unique order-insensitive group composed of $v_1, \dots, v_k \in V$.

For example, let $a, b \in V$ be two tags annotated with their *TagSupport*, whereby a precedes b lexicographically. The groups $\langle a, b \rangle, 1$ and $\langle b, a \rangle, 1$ are two distinct order-sensitive groups. $\langle a, b \rangle, 0$ is the order-insensitive group of the two elements. Finally, the group $\langle b, a \rangle, 0$ is not permitted, because the group is order-insensitive but the list violates the lexicographical ordering of list elements. Using $P(V) \times \{0, 1\}$, we define $V' \subseteq (P(V) \times \{0, 1\}) \times (0, 1]$ with signature:

$$\langle \text{TagGroup}, \text{GroupSupport} \rangle$$

V' contains only groups of annotations whose *GroupSupport* value is above a given threshold. Of course, the threshold value affects the size of the graph and the execution time of the algorithm traversing it to build the DTD. The set of nodes constituting our graph is $\mathcal{V} = V \cup V'$, indicating that a node may be a single tag or a group of tags with its/their statistics. An edge emanates from an element of V' and points to an element of V , i.e. from an associated group of tags to a single tag. Formally, we define the set of edges $E \subseteq (V \times V') \times \mathcal{X} \times \mathcal{X} \times \mathcal{X} \times \mathcal{X}$, where $\mathcal{X} := (0, 1] \cup \{NULL\}$, with signature:

$$\langle \text{Edge}, \text{AssociationConfidence}, \text{AssociationLift}, \\ \text{SequenceConfidence}, \text{SequenceLift} \rangle$$

In this signature, the statistical properties refer to the edge's target given the group of nodes in the edge's source. If the source is a sequence of adjacent tags, then *SequenceConfidence* and *SequenceLift* are the only valid statistical properties, because *AssociationConfidence* and *AssociationLift* are inapplicable. If the source is a set of tags, then the both *SequenceConfidence* and *SequenceLift* are inapplicable. Inapplicable statistical properties assume the NULL value.

4.3 Deriving DTD Components from the Graph of Associated Groups

The graph of associated groups has one node for each frequent tag and one node for each frequent group of tags. Depending on the support value threshold for discovering association rules and frequent sequences, the graph may contain a very large number of associated groups or rather the most frequent ones. In both cases, we perform further pruning steps to eliminate all associations that are of less importance in the context of a DTD. We consider the following pruning criteria:

- All edges with a lift (association lift or sequence lift) less than 1 are eliminated.
- All edges with a confidence less than a threshold are eliminated.
- All nodes containing tag groups that are not connected to a single tag by any edge, are removed. Such nodes are pruned after pruning all edges pointing to them.

- For each tag having k ingoing edges from tag groups, we retain only groups of maximal size, subject to a confidence threshold. This criterion states that if a tag x appears after a group g with confidence c and a subgroup g' of g with confidence c' , the subgroup g' is removed if $c' - c \leq \epsilon$, otherwise the group g is removed.

After this pruning procedure, the graph has been stripped off all groups that (i) reflect spurious associations, (ii) lead to tags with low confidence or (iii) can be replaced by groups that lead to frequent tags with higher confidence. The output of this procedure is a collection of frequent components of the envisaged DTD.

4.4 DTD as a Tree of Alternative XML Tag Sequences

The pruning phase upon the graph of associated groups delivers components of the probabilistic DTD. However, these components do not constitute a well-defined DTD, because there is no knowledge about their relative ordering and placement inside the type definition. Hence, we introduce a complementary DTD establishment algorithm, which derives complete sequences of DTD elements.

We observe an order-preserving DTD as a tree of alternative subsequences of XML tags. Each tag is adorned with its support with respect to the subsequence leading to it inside the tree: This support value denotes the number of documents starting with the same subsequence of tags. Each XML tag may appear in more than one subsequence, because of different predecessors in each one. Observing the DTD as a tree implies a common root. In the general case, each document of the archive may start at a different tag. We thus assume a dummy root whose children are tags appearing first in documents. In general, a tree node refers to a tag x , and its children refer to the tags appearing after x in the context of x 's own predecessors. In a sense, the DTD as a tree of alternatives resembles a DataGuide [21], although the latter contains no statistical adornments.

The tree-of-alternatives method is realized by the preprocessor module of the Web usage miner WUM [20]. This module is responsible for coercing sequences of events by common prefix and placing them in a tree structure that is called “aggregated tree”. This tree is input to the sequential pattern discovery process performed by WUM. The tag sequences contained in documents can be observed as sequences of events. Hence, the WUM preprocessor can also be used to build a DTD over an archive as a tree of alternative tag sequences.

The tree-of-alternatives can be pruned by the same set of criteria as applied in the graph of associated groups. Since each branch of the tree-of-alternatives corresponds to a sequence of adjacent tags, we only consider the statistical properties *SequenceLift*, *SequenceConfidence* and *GroupSupport* of sequences. It should also be stressed that the constraints placed upon the branches of this tree should be less restrictive than those applied upon *all* sequences of adjacent tags, because the tree branches correspond to complete sequences of tags, from the first tag in a document to the last one.

Finally, the DTD components retained on the graph of associated groups can be exploited to refine the tree-of-alternatives further. In particular, an ordered group of tags (i.e. a sequence of tags) $g = y_1 \dots y_n$ appearing as node in the graph may appear in several branches of the tree-of-alternatives. Then, if the same group appears in multiple branches with different prefixes (i.e. different sequences of tags prior to the group), these prefixes can be considered as alternatives inside the DTD, followed by a mandatory sequence of tags g .

In the current version of the DIAsDEM procedure for DTD establishment, we are still considering the graph of associated groups and the tree-of-alternatives as independent options, and are investigating the impact of heuristics combining them, like the aforementioned one, on the quality of the output DTD.

5 Case Study

To test the applicability of our approach, we have used 1,145 German Commercial Register entries published by the district court of Potsdam in 1999. These foundation entries of new companies have been semantically tagged by applying the original DIAsDEM framework as described in a recent case study [4].

Our current research successfully continued this case study by creating the DTD establishment graph for the previously derived, unstructured XML DTD illustrated in section 3. To this end, the Java-based DIAsDEM Workbench has been extended to analyze the XML document collection in order to compute *TagSupport* for all XML tags and to employ dedicated algorithms for association rule discovery (i.e. Weka [22]) and mining frequent sequences (i.e. WUM [20]) to compute *GroupSupport* as well as *Confidence* and *Lift* for tag associations and tag sequences. The following expert-given threshold values have been applied by the DIAsDEM Workbench: *TagSupport* > 0.5, *AssociationConfidence* > 0.75, *AssociationLift* > 1.2, *SequenceConfidence* > 0.5 and *SequenceLift* > 1.0.

Figure 1 depicts an excerpt of the DTD establishment graph that contains nodes corresponding to either individual XML tags (e.g., the tag *ShareCapital*

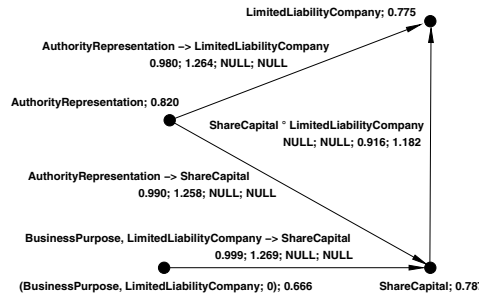


Fig. 1. Excerpt of the derived DTD establishment graph

with $TagSupport = 0.787$) or a group of XML tags (i.e. the tag set *BusinessPurpose, LimitedLiabilityCompany* with $GroupSupport = 0.666$). Figure 1 does not depict a node that corresponds to a sequence of tags. The edge connecting *AuthorityRepresentation* and *ShareCapital* represents the association rule $AuthorityRepresentation \rightarrow ShareCapital$ with very high $AssociationConfidence = 0.990$ and moderate $AssociationLift = 1.258$. Finally, the edge connecting *ShareCapital* and *LimitedLiabilityCompany* represents the frequent tag sequence $ShareCapital \cdot LimitedLiabilityCompany$ with high $SequenceConfidence = 0.916$ and moderate $SequenceLift = 1.182$.

Using the DTD establishment graph, the expert can acquire important insights into the semantic structure of an XML archive. By interactively visualizing this graph, a knowledge engineer can detect subsets of XML tags that are frequently occurring together in semantically annotated documents. In Figure 1, the tag group *BusinessPurpose, LimitedLiabilityCompany* occurs in 66% of the annotated XML documents. This fact indicates the existence of a semantic subgroup within the text archive focusing on legal matters of limited liability companies. The analysis of frequent sequences of adjacent tags reveals knowledge about the ordering of annotated text units within the XML documents. Referring to Figure 1, the tag *ShareCapital* is followed by *LimitedLiabilityCompany* with a probability of 0.916. Additionally, candidates for mandatory tags can be identified such as *AuthorityRepresentation* which is contained in 82% of the processed XML documents. Hence, the DTD establishment graph serves as an overall description of the semantic structure shared by either the entire domain-specific archive or subsets of semantically related XML documents.

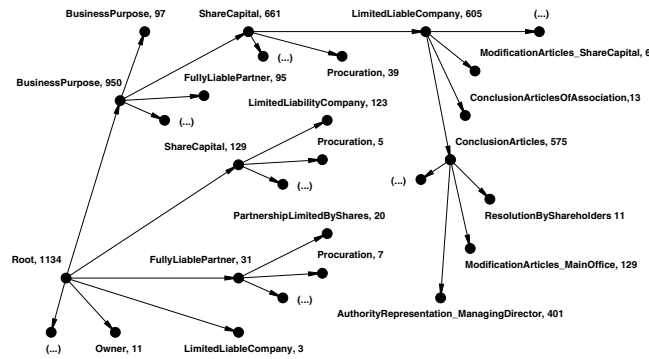


Fig. 2. Commercial Register DTD as a tree of alternative tag sequences

Figure 2 depicts the probabilistic XML DTD as a tree of alternative XML tag sequences that has been derived from the Commercial Register archive. Created by the preprocessing module of the Web Utilization Miner WUM [20], this tree explicitly describes sequences of XML tags frequently occurring in the archive.

For example, 950 (out of 1134) XML documents start with a text unit that is annotated with the XML tag *BusinessPurpose*. Following this tag sequence, 661 (out of 950) documents continue with a sentence that is tagged as *ShareCapital*. Using this probabilistic DTD, the expert can acquire knowledge about dominating sequences of XML tags which is essential for imposing an ordering upon the discovered XML tags.

6 Conclusion

Acquiring knowledge encapsulated in documents implies effective querying techniques as well as the combination of information from different texts. This functionality is usually confined to database-like query processors, while text search engines scan individual textual resources and return ranked results.

In this study, we have presented a methodology that structures document archives to enable query processing over them. We have proposed the derivation of an XML DTD over a domain-specific text archive by means of data mining techniques. Our main emphasis has been the combination of XML tags reflecting the semantics of many text units across the archive into a single DTD reflecting the semantics of the entire archive. The statistical properties of tags and their relationships form the basis for combining them into a unifying DTD. We use a graph data structure to depict all statistics that can serve as a basis for this operation, and we have proposed a mechanism that derives a DTD by employing a mining algorithm.

Our future work includes the implementation of further mechanisms to derive probabilistic DTDs and the establishment of a framework for comparing them in terms of expressiveness and accuracy. The data structure *probabilistic DTD* will be utilized to derive an archive-specific XML Schema and a relational schema, respectively. Ultimately, a full-fledged querying mechanism over text archives should be established. To this purpose, we intend to couple DTD derivation methods with a query mechanism for semi-structured data. Each semantic annotation corresponds to a label that semantically describes a discovered text unit cluster. Currently, the underlying clustering algorithm creates non-overlapping clusters. Hence, each text unit belongs to exactly one cluster. Since each text unit can only be annotated with the label of its cluster, the derived XML tags cannot be nested. An extension of the DIAsDEM Workbench to utilize a hierarchical clustering algorithm would allow for the establishment of subclusters and thus for the nesting of (sub)cluster labels. This is planned as future work as well.

References

1. Sullivan, D.: Document Warehousing and Text Mining. John Wiley & Sons, New York, Chichester, Weinheim (2001) 461
2. Erdmann, M., Maedche, A., Schnurr, H.P., Staab, S.: From manual to semi-automatic semantic annotation: About ontology-based text annotation tools. In: Proceedings of the COLING 2000 Workshop on Semantic Annotation and Intelligent Content, Luxembourg (2000) 461, 462

3. Graubitz, H., Spiliopoulou, M., Winkler, K.: The DIAsDEM framework for converting domain-specific texts into XML documents with data mining techniques. In: Proceedings of the First IEEE Int. Conference on Data Mining, San Jose, CA, USA (2001) 171–178 [461](#), [463](#)
4. Winkler, K., Spiliopoulou, M.: Semi-automated XML tagging of public text archives: A case study. In: Proceedings of EuroWeb 2001 “The Web in Public Administration”, Pisa, Italy (2001) 271–285 [461](#), [463](#), [465](#), [470](#)
5. Nahm, U. Y., Mooney, R. J.: Using information extraction to aid the discovery of prediction rules from text. In: Proceedings of the KDD-2000 Workshop on Text Mining, Boston, MA, USA (2000) 51–58 [462](#)
6. Feldman, R., Fresko, M., Kinar, Y., Lindell, Y., Liphstat, O., Rajman, M., Schler, Y., Zamir, O.: Text mining at the term level. In: Proceedings of the Second European Symposium on Principles of Data Mining and Knowledge Discovery, Nantes, France (1998) 65–73 [462](#)
7. Loh, S., Wives, L. K., Oliveira, J. P. M. d.: Concept-based knowledge discovery in texts extracted from the Web. *ACM SIGKDD Explorations* **2** (2000) 29–39 [462](#)
8. Bruder, I., Düsterhöft, A., Becker, M., Bedersdorfer, J., Neumann, G.: GETESS: Constructing a linguistic search index for an Internet search engine. In Bouzeghoub, M., Kedad, Z., Metais, E., eds.: *Natural Language Processing and Information Systems*. Number 1959 in *Lecture Notes in Computer Science*. Springer-Verlag (2001) 227–238 [462](#)
9. Sengupta, A., Purao, S.: Transitioning existing content: Inferring organization-specific document structures. In Turowski, K., Fellner, K. J., eds.: *Tagungsband der 1. Deutschen Tagung XML 2000, XML Meets Business*, Heidelberg, Germany (2000) 130–135 [463](#)
10. Moore, G. W., Berman, J. J.: Medical data mining and knowledge discovery. In: *Anatomic Pathology Data Mining*. Volume 60 of *Studies in Fuzziness and Soft Computing*, Heidelberg, New York, Physica-Verlag (2001) 72–117 [463](#)
11. Lumera, J.: Große Mengen an Altdaten stehen XML-Umstieg im Weg. *Computerwoche* **27** (2000) 52–53 [463](#)
12. Abiteboul, S., Buneman, P., Suciu, D.: *Data on the Web: From Relations to Semistructured Data and XML*. Morgan Kaufman Publishers, San Francisco (2000) [463](#)
13. Wang, K., Liu, H.: Discovering structural association of semistructured data. *IEEE Transactions on Knowledge and Data Engineering* **12** (2000) 353–371 [463](#)
14. Laur, P. A., Masseglia, F., Poncelet, P.: Schema mining: Finding regularity among semistructured data. In Zighed, D. A., Komorowski, J., Żytkow, J., eds.: *Principles of Data Mining and Knowledge Discovery: 4th European Conference, PKDD 2000*. Volume 1910 of *Lecture Notes in Artificial Intelligence*, Lyon, France, Springer, Berlin, Heidelberg (2000) 498–503 [463](#)
15. Carrasco, R. C., Oncina, J.: Learning deterministic regular grammars from stochastic samples in polynomial time. *RAIRO (Theoretical Informatics and Applications)* **33** (1999) 1–20 [463](#)
16. Young-Lai, M., Tompa, F. W.: Stochastic grammatical inference of text database structure. *Machine Learning* **40** (2000) 111–137 [463](#)
17. Agrawal, R., Srikant, R.: Mining sequential patterns. In: *Proc. of Int. Conf. on Data Engineering*, Taipei, Taiwan (1995) [466](#)
18. Baumgarten, M., Büchner, A. G., Anand, S. S., Mulvenna, M. D., Hughes, J. G.: Navigation pattern discovery from internet data. In: [\[23\]](#). (2000) 70–87 [466](#)
19. Gaul, W., Schmidt-Thieme, L.: Mining web navigation path fragments. In: [\[24\]](#). (2000) [466](#)

20. Spiliopoulou, M.: The laborious way from data mining to web mining. *Int. Journal of Comp. Sys., Sci. & Eng.*, Special Issue on “Semantics of the Web” **14** (1999) 113–126 [466](#), [469](#), [470](#), [471](#)
21. Goldman, R., Widom, J.: DataGuides: Enabling query formulation and optimization in semistructured databases. In: *VLDB’97*, Athens, Greece (1997) 436–445 [469](#)
22. Witten, I.H., Frank, E.: *Data Mining*. Morgan Kaufmann Publishers, San Francisco (2000) [470](#)
23. Masand, B., Spiliopoulou, M., eds.: *Advances in Web Usage Mining and User Profiling: Proceedings of the WEBKDD’99 Workshop*. LNAI 1836, Springer Verlag (2000) [473](#)
24. Kohavi, R., Spiliopoulou, M., Srivastava, J., eds.: *KDD’2000 Workshop WEBKDD’2000 on Web Mining for E-Commerce — Challenges and Opportunities*, Boston, MA, ACM (2000) [473](#)

Separability Index in Supervised Learning

Djamel A. Zighed, Stéphane Lallich, and Fabrice Muhlenbach

ERIC Laboratory – University of Lyon 2
5, av. Pierre Mendès-France, F-69676 BRON Cedex – FRANCE
{zighed,lallich,fmuhlenb}@univ-lyon2.fr

Abstract. We propose a new statistical approach for characterizing the class separability degree in R^p . This approach is based on a nonparametric statistic called “the Cut Edge Weight”. We show in this paper the principle and the experimental applications of this statistic. First, we build a geometrical connected graph like the Relative Neighborhood Graph of Toussaint on all examples of the learning set. Second, we cut all edges between two examples of a different class. Third, we calculate the relative weight of these cut edges. If the relative weight of the cut edges is in the expected interval of a random distribution of the labels on all the neighborhood graph’s vertices, then no neighborhood-based method will give a reliable prediction model. We will say then that the classes to predict are non-separable.

1 Introduction

Learning methods are very often requested in the data mining domain. The learning methods try to generate a prediction model φ from a learning sample Ω_l . Due to its construction method, the model is more or less reliable. This reliability is generally evaluated with *a posteriori* test sample Ω_t . The reliability depends on the learning sample, on the underlying statistical hypothesis, and on the implemented mathematical tools. Nevertheless, sometimes it does not exist any method that produce a reliable model, which can be explained by the following reasons:

- methods are not suitable to the problem we are trying to learn, so we have to find another method more adapted to the situation;
- the classes are not separable in the learning space. In this case, it is impossible to find a better learning method.

It will be very interesting to use mathematical tools that can characterize the class separability from a given learning sample. There already exist measures for learnability such as the VC-dimension provided by the statistical learning theory [20]. Nevertheless, VC-dimension is difficult to calculate in many cases. This problem has also been studied based on a statistical approach by Rao [16]. In the case of a normal distribution of the classes, Rao measures the learning ability degree through a test based on the population homogeneity. In a similar case, Kruskal and Wallis have defined a nonparametric test based on an equality

hypothesis of the scale parameters [1]. Recently, Sebban [18] and Zighed [23] have proposed a test based on the number of edges that connect examples of different classes in a geometrical neighborhood.

At first, they build a neighborhood structure by using some particular models like the Relative Neighborhood Graph of Toussaint [19]. After that, they calculate the number of edges that must be removed from the neighborhood graph to obtain clusters of homogeneous points in a given class. At last, they have established the law of the edge proportion that must be removed under the null hypothesis, denoted H_0 , of a random distribution of the labels. With this law, they can say if classes are separable or not by calculating the p-value of the test –e.g., the probability of having a calculated value as important as the observed value under H_0 .

In a more general view, we propose in this paper a theoretical framework and a nonparametric statistic that takes into consideration the weight of the removed edges. We exploit the works of the spatial autocorrelation, in particular the join-counts statistic, presented by Cliff and Ord [4] following the works of Moran [14], Krishna Iyer [9], Geary [7] and David [5]. Such process has been studied in the classification domain by Lebart [11] who used works based on the spatial contiguity, like the contiguity coefficient from Geary, to compare the local structures vs. the global structures in a k nearest neighbor graph.

To evaluate a learning method several points have to be distinguished. First, the quality of the results produced by the method have to be described, e.g., the determination coefficient R^2 in regression. Second, we have to test the hypothesis of the non-significance of the results. According to the number of instances, it could be known if the same value of R^2 is significant or not. Third, the robustness can be studied and the outliers can be searched. We propose a process that deals with all the previous points.

2 Class Separability, Clusters and Cut Edges

2.1 Notations

Machine learning methods intended to produce a function φ –like “decision rules” in the knowledge data discovery domain– that can predict the unknown belonging class $Y(\omega)$ of an instance ω extracted from the global population Ω , by knowing its representation $X(\omega)$.

In general, this representation $X(\omega)$ is provided by an expert who establishes *a priori* a set of attributes denoted: $X_1, X_2, \dots, X_p$. Let these attributes take their values in R , $X : \omega \in \Omega \mapsto X(\omega) = (X_1(\omega), X_2(\omega), \dots, X_p(\omega)) \in R^p$.

Within our context, all learning methods Φ must have recourse to a learning sample Ω_l and a test sample Ω_t . The former will be used for generating the prediction function φ , the latter will test the reliability of φ .

For all example $\omega \in (\Omega_l \cup \Omega_t)$, we suppose that its representation $X(\omega)$ and class $Y(\omega)$ are known. $Y : \Omega \mapsto \{y_1, \dots, y_k\}$, with k the number of classes of Y .

The learning ability of a method is strongly associated to its class separability degree in $X(\Omega)$. We consider that the classes will be easier to separate if they fulfill the following conditions:

- the instances of the same class appear mostly gathered in the same subgroup in the representation space;
- the number of groups are so small, at least it reaches the number of the classes;
- the borders between the groups are not complex.

2.2 Neighborhood Graphs and Clusters

To express the proximity between examples in the representation space, we use the “neighborhood graph” notion [23]. These graphs are the Relative Neighborhood Graph (RNG), the Gabriel Graph, the Delaunay Triangulation and the Minimal Spanning Tree, that all provide planar and connected graph structures. We use here the RNG of Toussaint [19] defined below.

Definition: Let V be a set of points in a real space R^p (with p the number of attributes). The Relative Neighborhood Graph (RNG) of V is a graph with vertices set V , and the set of edges of the RNG of V are exactly those pairs (a, b) of points for which $d(a, b) \leq \max(d(a, c), d(b, c)) \forall c, c \neq a, b$, where $d(u, v)$ denotes the distance between two points u and v in R^p .

This definition means that the *lune* $L_{(u,v)}$ —constituted by the intersections of hypercircles centered on u and v with range the edge (u, v) —is empty. For example, on Fig. 1 (a), vertices 13 and 15 are connected because there is no vertex on the *lune* $L_{(13,15)}$.

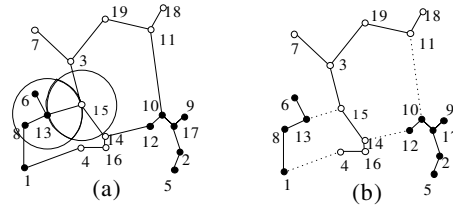


Fig. 1. RNG and clusters with two classes: the black and the white points

According to Zighed and Sebban [23] we introduce the concept of “cluster” to express that a set of close points have the same class. We call *cluster* a connected sub-graph of the neighborhood graph where all vertices belong to the same class. To build all clusters required for characterizing the structures of the scattered data points, we proceed in two steps:

1. we generate the geometrical neighborhood graph on the learning set;

2. we remove the edges connecting two vertices belonging to different classes, obtaining connected sub-graphs where all vertices belong to the same class.

The number of generated clusters gives a partial information on the class separability. If a number of clusters is low –at least the number of classes–, the classes are well separable and we can find a learning method capable of exhibit the model that underlies the particular group structure. For example on Fig. 1 (b), after cutting the four edges connecting vertices of different colors (in dotted line), we obtain three clusters for the two classes. But if this number tends to increase, closely to the number of clusters that we could have in a random situation, the classes could no longer be learned cause to the lack of a non random geometrical structure.

Actually, this number of clusters cannot ever characterize some little situations that seems intuitively different. For the same number of clusters, the situation can be very different depending on whether the clusters are easily isolated in the neighborhood graph or not. As soon as $p > 1$, rather than studying the number of clusters, we prefer to take an interest in the edges cut for building the clusters and we will calculate the relative weight of these edges in the edge set. In our example on Fig. 1 (b), we have cut four edges for isolating three clusters.

3 Cut Edge Weight Statistic

In a common point between supervised classification and spatial analysis, we consider a spatial contiguity graph which plays the role of the neighborhood graph [4]. The vertices of this graph are colored with k distinct colors. The color plays the role of the class Y . The matter is (1) to describe the link between the adjacency of two vertices and the fact they have the same color, and (2) to test the hypothesis of non significance. This would take us to test the hypothesis of no spatial autocorrelation between the values taken by a categorical variable over spatial units. In the case of a neighborhood graph, this would be the results for testing the hypothesis that the class Y cannot be learned from neighborhood-based methods.

3.1 Statistical Framework

Notations and Abbreviations

- Number of nodes in the graph: n
- Connection matrix: $V = (v_{ij}), i = 1, 2, \dots, n; j = 1, 2, \dots, n$; where $v_{ij} = 1$ if i and j are linked by an edge
- Weight matrix: $W = (w_{ij}), i = 1, 2, \dots, n; j = 1, 2, \dots, n$; where w_{ij} is the weight of edge (i, j) . Let w_{i+} and w_{+j} be the sums of row i and column j . We consider that W matrix is symmetrical. If we have to work with a non symmetrical matrix W' , which is very interesting for neighborhood graphs, we will go back to the symmetrical case without loss of generality calculating: $w_{ij} = \frac{1}{2} (w'_{ij} + w'_{ji})$.

- Number of edges: a
- Proportion of vertices corresponding to the class y_r : π_r , $r = 1, 2, \dots, k$

According to Cliff and Ord [4], we adopt the simplified notations below:

Notations	Definition	Case : $W = V$
$\sum_2 w_{ij}$	$\sum_{i=1}^n \sum_{j=1, i \neq j}^n w_{ij}$	$2a$
S_0	$\sum_2 w_{ij}$	$2a$
S_1	$\frac{1}{2} \sum_2 (w_{ij} + w_{ji})^2$	$4a$
S_2	$\sum_{i=1}^n (w_{i+} + w_{+i})^2$	$4 \sum_{i=1}^n v_{i+}^2$

Definition of the Cut Edge Weight Statistic In order to take into consideration a possible weighting of the edges, we deal with the symmetrized weights matrix W which is reduced to the connection matrix V if all the weights are equal to 1. We consider both the symmetrical weights based upon the distances and non symmetrical weights based upon the ranks. In the case of distances, we choose $w_{ij} = (1 + d_{ij})^{-1}$, while in the case of ranks we choose $w_{ij} = \frac{1}{r_j}$, where r_j is the rank of the vertex j among the neighbors of the vertex i .

Edges linking two vertices of the same class (non cut edges) have to be distinguished from those linking two vertices of different classes (cut edges in order to obtain clusters).

Let us denote by I_r the sum of weights relative to edges linking two vertices of class r , and by $J_{r,s}$ the sum of weights relative to edges linking a vertex of class r and a vertex of class s . Statistics I and J are defined as it follows.

non cut edges	cut edges
$I = \sum_{r=1}^k I_r$	$J = \sum_{r=1}^{k-1} \sum_{s=r+1}^k J_{r,s}$

In so far as I and J are connected by the relation $I + J = \frac{1}{2} S_0$, we have only to study J statistic or its normalization $\frac{J}{I+J} = \frac{2J}{S_0}$. Both give the same result after standardization. We may observe that I generalizes the test of runs in 2 dimensions and k groups [13,21].

Random Framework Like Jain and Dubes [8], we consider binomial sampling in which null hypothesis is defined by:

H_0 : the vertices of the graph are labelled independently of each other, according to the same probability distribution (π_r) where π_r denotes the probability of the class r , $r = 1, 2, \dots, k$.

We could consider hypergeometric sampling by adding into null hypothesis the constraint to have n_r vertices of the class r , $r = 1, 2, \dots, k$.

Rejecting null hypothesis means either the classes are non independently distributed or the probability distribution of the classes is not the same for the different vertices. In order to test the null hypothesis H_0 using statistic J (or I), we had first to study the distribution of these statistics under H_0 .

3.2 Distribution of I and J under Null Hypothesis

To test H_0 with the statistic J , we will use two-sided test if we are surprised at once by abnormally small values of J (great separability of the classes) and by abnormally great values (deterministic structuration or pattern presence). Hypothesis H_0 is rejected when J produce an outstanding value taking into account its distribution under H_0 . So, we have to establish the distribution of J under H_0 in order to calculate the p-value associated with the observed value of J as well as to calculate the critical value of J at the significance level α_0 . This calculation can be done either by simulation or by normal approximation. In the last case, we have to calculate the mean and the variance of J under H_0 .

Boolean Case The two classes defined by Y are noted 1 and 2. According to Moran [14], $U_i = 1$ if the class of the i^{th} vertex is 1 and $U_i = 0$ if the class is 2, $i = 1, 2, \dots, n$. We denote π_1 the vertex proportion of class 1 and π_2 the vertex proportion of class 2. Thus:

$$J_{1,2} = \frac{1}{2} \sum_2 w_{ij} (U_i - U_j)^2 = \frac{1}{2} \sum_2 w_{ij} Z_{ij}$$

where U_i are independently distributed according to Bernoulli distribution of parameter π_1 , noted $B(1, \pi_1)$. It must be noticed that the variables $Z_{ij} = (U_i - U_j)^2$ are distributed according to the distribution $B(1, 2\pi_1\pi_2)$, but are not independent. Actually, the covariances $Cov(Z_{ij}, Z_{kl})$ are null only if the four indices are different. Otherwise, when there is a common index, one can obtain:

$$Cov(Z_{ij}, Z_{il}) = \pi_1\pi_2(1 - 4\pi_1\pi_2)$$

The table below summarizes the different results related to the statistic $J_{1,2}$:

Variable	Mean	Variance
U_i	π_1	$\pi_1\pi_2$
$Z_{ij} = (U_i - U_j)^2$	$2\pi_1\pi_2$	$2\pi_1\pi_2(1 - 2\pi_1\pi_2)$
$J_{1,2}$	$S_0\pi_1\pi_2$	$S_1\pi_1^2\pi_2^2 + S_2\pi_1\pi_2\left(\frac{1}{4} - \pi_1\pi_2\right)$
$J_{1,2}$ si $w_{ij} = v_{ij}$	$2a\pi_1\pi_2$	$4a\pi_1^2\pi_2^2 + \pi_1\pi_2(1 - 4\pi_1\pi_2) \sum_{i=1}^n v_{i+}^2$

The p-value of $J_{1,2}$ is calculated from standard normal distribution after centering and reducing its observed value. The critical values for $J_{1,2}$ at the significance level α_0 are:

$$J_{1,2;\alpha_0/2} = S_0\pi_1\pi_2 - u_{1-\alpha_0/2} \sqrt{S_1\pi_1^2\pi_2^2 + S_2\pi_1\pi_2\left(\frac{1}{4} - \pi_1\pi_2\right)}$$

$$J_{1,2;1-\alpha_0/2} = S_0\pi_1\pi_2 + u_{1-\alpha_0/2} \sqrt{S_1\pi_1^2\pi_2^2 + S_2\pi_1\pi_2\left(\frac{1}{4} - \pi_1\pi_2\right)}$$

By simulation, the most convenient is to calculate the p-value associated with the observed value of $J_{1,2}$. To simulate a realization of $J_{1,2}$, one only has to simulate a realization of $B(1, \pi_1)$ for each example, which requires n random numbers between 0 and 1, and then to apply the formula which defines $J_{1,2}$. After having repeated N times the operation, one calculates the p-value associated with the observed value of $J_{1,2}$ by calculating the proportion of simulated values of $J_{1,2}$ which are less or equal to the observed value of $J_{1,2}$.

Multiclass Case To extend these results to the multiclass case, according to Cliff and Ord [4], we reason with I and J statistics already defined. These statistics are:

$$I = \sum_{r=1}^k I_r = \frac{1}{2} \sum_2 w_{ij} T_{ij} \quad J = \sum_{r=1}^{k-1} \sum_{s=r+1}^k J_{r,s} = \frac{1}{2} \sum_2 w_{ij} Z_{ij}$$

where T_{ij} and Z_{ij} are random boolean variables which indicate if the vertices i and j have the same class (T_{ij}) or not (Z_{ij}).

From previous results, we easily obtain the mean of I and J :

Test statistic	Mean
$I = \sum_{r=1}^k I_r$	$\frac{1}{2} S_0 \sum_{r=1}^k \pi_r^2$
$J = \sum_{r=1}^{k-1} \sum_{s=r+1}^k J_{r,s}$	$S_0 \sum_{r=1}^{k-1} \sum_{s=r+1}^k \pi_r \pi_s$

Because I and J are connected by the relation $I + J = \frac{1}{2} S_0$, these two variables have the same variance, denoted $\sigma^2 = Var(I) = Var(J)$. The calculation of σ^2 is complicated due to the necessity of taking the covariances into consideration. In accordance with Cliff and Ord [4], we obtain the following results for binomial sampling:

$$4\sigma^2 = S_2 \sum_{r=1}^{k-1} \sum_{s=r+1}^k \pi_r \pi_s + (2S_1 - 5S_2) \sum_{r=1}^{k-2} \sum_{s=r+1}^{k-1} \sum_{t=s+1}^k \pi_r \pi_s \pi_t + 4(S_1 - S_2) \left[\sum_{r=1}^{k-1} \sum_{s=r+1}^k \pi_r^2 \pi_s^2 - 2 \sum_{r=1}^{k-3} \sum_{s=r+1}^{k-2} \sum_{t=s+1}^{k-1} \sum_{u=t+1}^k \pi_r \pi_s \pi_t \pi_u \right]$$

3.3 Complexity of the Test

Differents steps are into consideration: computing the matrix distance is in $O(p \times n^2)$, with n the number of examples and p the attributes, and building the neighborhood graph in R^p is in $O(n^3)$. Because the number of attributes p is very small compared to the number of instances n , the test is in $O(n^3)$.

We point out that all the complete database is not needed for the test. A sample, particularly a stratified sample, can be enough to reveal a good idea of the class separability of the database.

4 From Numerical Attributes to Categorical Attributes

We have introduced the test of weighted cut edges for Y is a categorical variable and the attributes $X_1, X_2, \dots, X_p$ are numerical. One notice that in order to apply such a test in a supervised learning, we only need to build the neighborhood

graph which summarizes the information brought by the attributes. To the extent that the building of this neighborhood graph only requires the dissimilarity matrix between examples, we may consider a double enlargement of the weighted cut edge test.

The first enlargement corresponds to the situation of categorical attributes $X_j, j = 1, 2, \dots, p$, which often exists in the real world. In such a case, it is enough to construct a dissimilarity matrix from the data.

We have to use a dissimilarity measure suited to the nature of attributes (cf. Chandon and Pinson [3], Esposito et al. [6]).

In the case of boolean data, there is a set of similarity indices between examples relying on the number of matching “1” (noted by a) or “0” (d) and the number of mismatching “1-0” (b) or “0-1” (c). A general formula for similarity ($s_{\theta_1\theta_2}$) and dissimilarity ($d_{\theta_1\theta_2}$) indices taking their values between 0 and 1, is:

$$s_{\theta_1\theta_2} = \frac{a + \theta_1 d}{a + \theta_1 d + \theta_2(b + c)} = 1 - d_{\theta_1\theta_2}$$

Most known indices are mentioned in the table above.

Table 1. Main similarity indices

θ_1	θ_2	Name
1	1	Sokal and Michener, 1958
1	2	Rogers and Tanimoto, 1960
1	0.5	not named
0	1	Jaccard, 1900
0	2	Sokal and Sneath, 1963
0	0.5	Czekanowski, 1913; Dice, 1945

In the case of categorical data, there are two main methods:

- either to generalize the previously quoted indices when it is possible. For example, Sokal and Michener index is the proportion of matching categorical attributes. It is possible to weight the attributes according to their number of categorical components.
- or to rewrite each categorical attribute as a set of boolean attributes in order to use indices for boolean data. In this case, all the examples have the same number of “1”, namely p . Then, according to Lerman (1970), all the indices mentioned in Table 1 lead to the same ordering on the set of example’s pairs. Applying Minkowski distance of parameter 1 or 2 to such a matrix is equivalent to the generalization of Sokal and Michener index.

Lastly, when variables are of different types, one can use a linear weighted combination of dissimilarity measures adapted to each type of variable or reduce the data to the same type [6].

The second enlargement deals with the situation where only a dissimilarity matrix D is known and not the original data X . This situation arises for instance in the case of input-output tables (e.g., Leontiev Input-output table) or when the collected information is directly dissimilarity matrix (e.g., in marketing or psychology trials).

5 Experiments

5.1 Cut Weighted Edge Approach for Numerical Attributes

Values of the Cut Weighted Edge Test The weighted edge test has been experimentally studied on 13 benchmarks from the UCI Machine Learning Repository [2]. These databases have been chosen for having only numerical attributes and a symbolic class. For each base, we build a relative neighborhood graph [19] on the n instances of the learning set. In Table 1, the results show the number of instances n , the number of attributes p and the number of classes k . We present also information characterizing the geometrical graph: number of obtained edges for constructing the graph (*edges*) and the number of cluster obtained after cutting the edges linking two vertices of different classes (*clusters*).

Table 2. Cut weighted edge test values on 13 benchmarks

General information							without weighting			weighting: distance			weighting: rank		
Domain name	n	p	k	clust.	edges	error r.	$J / (I + J)$	J^s	p-value	$J / (I + J)$	J^s	p-value	$J / (I + J)$	J^s	p-value
Wine recognition	178	13	3	9	281	0.0389	0.093	-19.32	0	0.054	-19.40	0	0.074	-19.27	0
Breast Cancer	683	9	2	10	7562	0.0409	0.008	-25.29	0	0.003	-24.38	0	0.014	-25.02	0
Iris (Bezdek)	150	4	3	6	189	0.0533	0.090	-16.82	0	0.077	-17.01	0	0.078	-16.78	0
Iris plants	150	4	3	6	196	0.0600	0.087	-17.22	0	0.074	-17.41	0	0.076	-17.14	0
Musk "Clean1"	476	166	2	14	810	0.0650	0.167	-17.53	0	0.115	-7.69	2E-14	0.143	-18.10	0
Image seg.	210	19	7	27	268	0.1238	0.224	-29.63	0	0.141	-29.31	0	0.201	-29.88	0
Ionosphere	351	34	2	43	402	0.1397	0.137	-11.34	0	0.046	-11.07	0	0.136	-11.33	0
Waveform	1000	21	3	49	2443	0.1860	0.255	-42.75	0	0.248	-42.55	0	0.248	-42.55	0
Pima Indians	768	8	2	82	1416	0.2877	0.310	-8.74	2E-18	0.282	-9.86	0	0.305	-8.93	4E-19
Glass Ident.	214	9	6	52	275	0.3169	0.356	-12.63	0	0.315	-12.90	0	0.342	-12.93	0
Haberman	306	3	2	47	517	0.3263	0.331	-1.92	0.0544	0.321	-2.20	0.028	0.331	-1.90	0.058
Bupa	345	6	2	50	581	0.3632	0.401	-3.89	0.0001	0.385	-4.33	1E-05	0.394	-4.08	5E-05
Yeast	1484	8	10	401	2805	0.4549	0.524	-27.03	0	0.512	-27.18	0	0.509	-28.06	0

On Table 2, for each base, we present the relative cut edge weight $\frac{J}{I+J}$, the standardized cut weighted edge test J^s with its p-value in three cases: when the test is done without weighting, when the edges are weighted by the inverse of the distance between the vertices, and when the edges are weighted by the inverse of the number of the rank of a vertex to the others of the graph. For each base and weighting method, the p-values are extremely low, this shows that the null hypothesis of a random distribution of the labels on the vertices of a neighborhood graph is very strong.

For information, the empirical evaluation of the CPU time needed for the test (distance matrix computation, graph construction, edges cut, test statistic

calculation) is between a little less than 1 second for *Iris* (150 instances) and 200 seconds for *Yeast* (about 1,500 instances) on a 450 MHz PC. We present only the results obtained with a RNG graph of Toussaint (the results with a Gabriel Graph or a Minimal Spanning Tree are very close to them).

Weight of the Cut Edges and Error Rate in Machine Learning The 13 benchmarks have been tested on the following different machine learning methods: instance-based learning method (the nearest neighborhood: 1-NN [12]), decision tree (C4.5 [15]), induction graph (Sipina [22]), artificial neural networks (Perceptron [17], Multi-Layer Perceptron with 10 neurons on one hidden layer [12]) and the Naive Bayes [12]. On Table 3 we present the error rates obtained by these methods on a 10 cross validation with the benchmarks and the statistical values previously calculated (without weighting). The rate errors for the different learning methods, and particularly the mean of these methods, are well correlated with the relative cut edge weight ($J/(I + J)$). We can see on Fig. 2 the linear relation between the relative cut edge weight and the mean of the error rate for the 13 benchmarks.

Table 3. Error rates and statistical values of the 13 benchmarks

Domain name	General information					Statistical value			Error rate						
	n	p	k	clust.	edges	$J/(I+J)$	J^2	p-value	1-NN	C4.5	Sipina	Perc.	MLP	N. Bayes	Mean
Breast Cancer	683	9	2	10	7562	0.008	-25.29	0	0.041	0.059	0.050	0.032	0.032	0.026	0.040
BUFA liver	345	6	2	50	581	0.401	-3.89	0.0001	0.363	0.369	0.347	0.305	0.322	0.380	0.348
Glass Ident.	214	9	6	52	275	0.356	-12.63	0	0.317	0.289	0.304	0.350	0.448	0.401	0.352
Haberman	306	3	2	47	517	0.331	-1.92	0.0544	0.326	0.310	0.294	0.241	0.275	0.284	0.288
Image seg.	210	19	7	27	268	0.224	-29.63	0	0.124	0.124	0.152	0.119	0.114	0.605	0.206
Ionosphere	351	34	2	43	402	0.137	-11.34	0	0.140	0.074	0.114	0.128	0.131	0.160	0.124
Iris (Bezdek)	150	4	3	6	189	0.090	-16.82	0	0.053	0.060	0.067	0.060	0.053	0.087	0.063
Iris plants	150	4	3	6	196	0.087	-17.22	0	0.060	0.033	0.053	0.067	0.040	0.080	0.056
Musk "Clean1"	476	166	2	14	810	0.167	-17.53	0	0.065	0.162	0.232	0.187	0.113	0.227	0.164
Pima Indians	768	8	2	82	1416	0.310	-8.74	2.4E-18	0.288	0.283	0.270	0.231	0.266	0.259	0.266
Waveform	1000	21	3	49	2443	0.255	-42.75	0	0.186	0.260	0.251	0.173	0.169	0.243	0.214
Wine recognition	178	13	3	9	281	0.093	-19.32	0	0.039	0.062	0.073	0.011	0.017	0.186	0.065
Yeast	1484	8	10	401	2805	0.524	-27.03	0	0.455	0.445	0.437	0.447	0.446	0.435	0.444
Mean									0.189	0.195	0.203	0.181	0.187	0.259	0.202
$R^2 (J/(I+J) ; \text{error rate})$									0.933	0.934	0.937	0.912	0.877	0.528	0.979
$R^2 (J^2 ; \text{error rate})$									0.076	0.020	0.019	0.036	0.063	0.005	0.026

5.2 Complementary Experiments

Cut Weighted Test and Categorical Attributes To show how to deal with categorical attributes, we have applied the cut weighted edge test on the benchmark *Flag* of the UCI Repository [2] that contains such predictors (Table 4). The categorical attributes have been rewritten as a set of boolean attributes and the neighborhood graph is build with all standardized attributes. The test indicates that this base is separable, related to the mean error rate of 0.36 for 6 classes to learn.

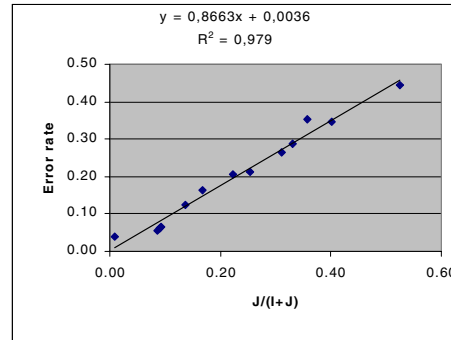


Fig. 2. Relative cut edge weight and mean of the error rates

Size Effect of the Database We point out the fact that J^s , the standardized cut weighted edge statistic, and then the p-value depend strongly of the size of the learning set. The same observed deviation in the null hypothesis is more significant, because of the learning set size. This fact is illustrated by the results of experiments conducted on the benchmark waves, for different size of learning set ($n=20, 50, 100, 1000$). The results of the tests are shown on Table 4. The error rates are decreasing but we do not present their values in the different learning methods because of the great variability due to the small size of the learning set. The p-value is not significant for $n=20$, and it is more and more significant when n increases. Concurrently, we notice that $\frac{J}{I+J}$ decreases as well as the error rate.

Table 4. Error rates and statistical values of the other databases

Domain name	n	p	k	clust.	edges	$J/(I+J)$	J^s	p-value	1-NN	C4.5	Sipina	Perc.	MLP	N.Bayes	Mean
Flag	194	67	6	46	327	0.489	-13.91	0	0.366	0.346	0.371	0.310	0.428	0.340	0.360
Waves-20	20	21	3	6	25	0.400	-0.44	0.6635							
Waves-50	50	21	3	11	72	0.375	-4.05	5.0E-05							
Waves-100	100	21	3	12	156	0.301	-8.44	3.3E-17							
Waves-1000	1000	21	3	49	2443	0.255	-42.75	0							

6 Conclusion

In this paper that proceeds the research of Zighed and Sebban [23], our results outcome a strict framework that permits to take into consideration the weight of the edges for numerical or categorical attributes. Furthermore we can use this framework to detect outliers and improve classification [10].

The construction of the test is based on the existence of a neighborhood graph. To build this graph, the dissimilarity matrix is only needed. This char-

acteristic gives to our approach a very general dimension to estimate the class separability, however the instance representation may be known or not.

Our perspectives are to describe the procedures of implementing and identifying the application fields, in order to make tests on real applications.

References

1. S. Aivazian, I. Enukov, and L. Mechalkine. *Eléments de modélisation et traitement primaire des données*. MIR, Moscou, 1986. 476
2. C. L. Blake and C. J. Merz. UCI repository of machine learning databases. Irvine, CA: University of California, Department of Information and Computer Science [http://www.ics.uci.edu/~mlearn/MLRepository.html], 1998. 483, 484
3. J. L. Chandon and S. Pinson. *Analyse Typologique, Théories et Applications*. Masson, 1981. 482
4. A. D. Cliff and J. K. Ord. *Spatial processes, models and applications*. Pion Limited, London, 1986. 476, 478, 479, 481
5. F. N. David. Measurement of diversity. In *Proceedings of the Sixth Berkeley Symposium on Mathematical Statistics and Probability*, pages 109–136, Berkeley, USA, 1971. 476
6. F. Esposito, D. Malerba, V. Tamma, and H. H. Bock. Similarity and dissimilarity measures: classical resemblance measures. In H. H. Bock and E. Diday, editors, *Analysis of Symbolic data*, pages 139–152. Springer-Verlag, 2000. 482
7. R. C. Geary. The contiguity ratio and statistical mapping. *The Incorporated Statistician*, 5:115–145, 1954. 476
8. A. K. Jain and R. C. Dubes. *Algorithms for clustering data*. Prentice Hall, 1988. 479
9. P. V. A. Krishna Iyer. The first and second moments of some probability distribution arising from points on a lattice, and their applications. In *Biometrika*, number 36, pages 135–141, 1949. 476
10. S. Lallich, F. Muhlenbach, and D. A. Zighed. Improving classification by removing or relabeling mislabeled instances. In *Proceedings of the XIIIth Int. Symposium on Methodologies for Intelligent Systems (ISMIS)*, 2002. To appear in LNAI. 485
11. L. Lebart. Data anlysis. In W. Gaul, O. Opitz, and M. Schader, editors, *Contiguity analysis and classification*, pages 233–244, Berlin, 2000. Springer. 476
12. T. Mitchell. *Machine Learning*. McGraw Hill, 1997. 484
13. A. Mood. The distribution theory of runs. *Ann. of Math. Statist.*, 11:367–392, 1940. 479
14. P. A. P. Moran. The interpretation of statistical maps. In *Journal of the Royal Statistical Society*, serie B, pages 246–251, 1948. 476, 480
15. J. R. Quinlan. *C4.5: Program for Machine Learning*. Morgan Kaufmann, San Mateo, Ca, 1993. 484
16. C. R. Rao. *Linear statistical inference and its applications*. Wiley, New-York, 1972. 475
17. F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–408, 1958. 484
18. M. Sebban. *Modèles théoriques en reconnaissance des formes et architecture hybride pour machine perceptive*. PhD thesis, Université Lyon 2, 1996. 476
19. G. Toussaint. The relative neighborhood graph of a finite planar set. *Pattern recognition*, 12:261–268, 1980. 476, 477, 483

20. V. Vapnik. *Statistical Learning Theory*. John Wiley, NY, 1998. 475
21. A. Wald and J. Wolfowitz. On a test whether two samples are from the same population. *Ann. of Math. Statist.*, 11:147–162, 1940. 479
22. D. A. Zighed, J. P. Auray, and G. Duru. *SIPINA : Méthode et logiciel*. Lacassagne, 1992. 484
23. D. A. Zighed and M. Sebban. Sélection et validation statistique de variables et de prototypes. In M. Sebban and G. Venturini, editors, *Apprentissage automatique*. Hermès Science, 1999. 476, 477, 485

Finding Hidden Factors Using Independent Component Analysis

Erkki Oja

Helsinki University of Technology
Neural Networks Research Centre
P.O.B. 5400, 02015 HUT, Finland
`Erkki.Oja@hut.fi`

Abstract. Independent Component Analysis (ICA) is a computational technique for revealing hidden factors that underlie sets of measurements or signals. ICA assumes a statistical model whereby the observed multivariate data, typically given as a large database of samples, are assumed to be linear or nonlinear mixtures of some unknown latent variables. The mixing coefficients are also unknown. The latent variables are non-gaussian and mutually independent, and they are called the independent components of the observed data. By ICA, these independent components, also called sources or factors, can be found. Thus ICA can be seen as an extension to Principal Component Analysis and Factor Analysis. ICA is a much richer technique, however, capable of finding the sources when these classical methods fail completely.

In many cases, the measurements are given as a set of parallel signals or time series. Typical examples are mixtures of simultaneous sounds or human voices that have been picked up by several microphones, brain signal measurements from multiple EEG sensors, several radio signals arriving at a portable phone, or multiple parallel time series obtained from some industrial process. The term blind source separation is used to characterize this problem.

The lecture will first cover the basic idea of demixing in the case of a linear mixing model and then take a look at the recent nonlinear demixing approaches.

Although ICA was originally developed for digital signal processing applications, it has recently been found that it may be a powerful tool for analyzing text document data as well, if the documents are presented in a suitable numerical form. A case study on analyzing dynamically evolving text is covered in the talk.

Reasoning with Classifiers^{*}

Dan Roth

Department of Computer Science
University of Illinois at Urbana-Champaign
danr@cs.uiuc.edu

Abstract. Research in machine learning concentrates on the study of learning *single* concepts from examples. In this framework the learner attempts to learn a single hidden function from a collection of examples, assumed to be drawn independently from some unknown probability distribution. However, in many cases – as in most natural language and visual processing situations – decisions depend on the outcomes of several different but mutually dependent classifiers. The classifiers’ outcomes need to respect some constraints that could arise from the sequential nature of the data or other domain specific conditions, thus requiring a level of inference on top the predictions.

We will describe research and present challenges related to *Inference with Classifiers* – a paradigm in which we address the problem of using the outcomes of several different classifiers in making coherent inferences – those that respect constraints on the outcome of the classifiers. Examples will be given from the natural language domain.

The emphasis of the research in machine learning has been on the study of learning *single* concepts from examples. In this framework the learner attempts to learn a single hidden function from a collection of examples, assumed to be drawn independently from some unknown probability distribution, and its performance is measured when classifying future examples.

In the context of natural language, for example, work in this direction has allowed researchers and practitioners to address the robust learnability of predicates such as “the part-of-speech of the word **can** in the given sentence is **noun**”, “the semantic sense of the word “plant” in the given sentence is “an industrial plant”, or determine, in a given sentence, the word that starts a noun phrase. In fact, a large number of disambiguation problems such as part-of speech tagging, word-sense disambiguation, prepositional phrase attachment, accent restoration, word choice selection in machine translation, context-sensitive spelling correction, word selection in speech recognition and identifying discourse markers have been addressed using machine learning techniques – in each of these problems it is necessary to disambiguate two or more [semantically, syntactically or structurally]-distinct forms which have been fused together into the same representation in some medium; a stand alone classifier can be learned to perform these task quite successfully [10].

^{*} Paper written to accompany an invited talk at ECML’02. This research is supported by NSF grants IIS-99-84168, ITR-IIS-00-85836 and an ONR MURI award.

However, in many cases – as in most natural language and visual processing situations – higher level decisions depend on the outcomes of several different but mutually dependent classifiers. Consider, for example, the problem of *chunking* natural language sentences where the goal is to identify several kinds of phrases (e.g. noun (NP), verb (VP) and prepositional (PP) phrases) in sentences, as in:

[NP He] [VP reckons] [NP the current account deficit] [VP will narrow
] [PP to] [NP only \$ 1.8 billion] [PP in] [NP September] .

A task of this sort involves multiple predictions that interact in some way. For example, one way to address the problem is to utilize two classifiers for each type of phrase, one of which recognizes the beginning of the phrase, and the other its end. Clearly, there are constraints over the predictions; for instance, phrases cannot overlap and there may also be probabilistic constraints over the order of phrases and over their lengths. The goal is to minimize some global measure of accuracy, not necessarily to maximize the performance of each individual classifier involved in the decision [8].

As a second example, consider the problem of recognizing the *kill* (KFJ, Oswald) relation in the sentence “J. V. Oswald was murdered at JFK after his assassin, R. U. KFJ...”. This task requires making several local decisions, such as identifying named entities in the sentence, in order to support the relation identification. For example, it may be useful to identify that Oswald and KFJ are *people*, and JFK is a *location*. In addition, it is necessary to identify that the action *kill* is described in the sentence. All of this information will help to discover the desired relation and identify its arguments. At the same time, the relation *kill* constrains its arguments to be *people* (or at least, not to be *locations*) and, in turn, helps to enforce that Oswald and KFJ are likely to be *people*, while JFK is not.

Finally, consider the challenge of designing a free-style natural language user interface that allows users to request in-depth information from a large collection of on-line articles, the web, or other semi-structured information sources. Specifically, consider the computational processes required in order to “understand” a simple question of the form “**what is the fastest automobile in the world?**”, and respond correctly to it. A straight forward key-word search may suggest that the following two passages contain the answer:

... will stretch Volkswagen’s lead in *the world’s fastest* growing *vehicle* market. Demand for *cars* is expected to soar...

... the Jaguar XJ220 is the dearest (415,000 pounds), *fastest* (217mph) and most sought after *car in the world*.

However, “understanding” the question and the passages to a level that allows a decision as to which in fact contains the correct answer, and extracting it, is a very challenging task.

Traditionally, the tasks described above have been viewed as inferential tasks [4, 7]; the hope was that stored knowledge about the language and the world will

allow inferring the syntactic and semantic analysis of the question and the candidate answers; background knowledge (e.g., Jaguar is a car company; automobile is synonymous to car) will then be used to choose the correct passage and to extract the answer. However, it has become clear that many of the difficulties in this task involve problems of context-sensitive ambiguities. These are abundant in natural language and occur at various levels of the processing, from syntactic disambiguation (is “demand” a Noun or a Verb?), to sense and semantic class disambiguation (what is a “Jaguar”?), phrase identification (importantly, “the world’s fastest growing vehicle market” is a noun phrase in the passage above) and others. Resolving any of these ambiguities require a lot of knowledge about the world and the language, but knowledge that cannot be written “explicitly” ahead of time. It is widely accepted today that any robust computational approach to these problems has to rely on a significant component of statistical learning, used both to acquire knowledge and to perform low level predictions of the type mentioned above.

The inference component is still very challenging. This view suggests, however, that rather than a deterministic collection of “facts” and “rules”, the inference challenge stems from the interaction of the large number of learned predictors involved. Inference of this sort is needed at the level of determining an answer to the question. An answer to the abovementioned question needs to be a name of a car company (predictor 1: identify the sought after entity; predictor 2: determine if the string Z represents a name of a car company) but also the subject of a sentence (predictor 3) in which a word equivalent to “fastest” (predictor 4) modifies (predictor 5) a word equivalent to “automobile” (predictor 6). Inferences of this sort are necessary also at other, lower levels of the process, as in the abovementioned problem of identifying noun phrases in a given sentence.

Thus, decisions typically depend on the outcomes of several predictors and they need to be made in ways that provide coherent inferences that satisfy some constraints. These constraints might arise from the sequential nature of the data, from semantic or pragmatic considerations or other domain specific conditions.

The examples described above exemplify the need for a unified theory of learning and inference. The purpose of this talk is to survey research in this direction, present progress and challenges.

Earlier works in this direction have developed the *Learning to Reason* framework - an integrated theory of learning, knowledge representation and reasoning within a unified framework [2, 9, 12]. This framework addresses an important aspect of the fundamental problem of unifying learning and reasoning - it proves the benefits of performing reasoning on top of learned hypotheses. And, by incorporating learning into the inference process it provides a way around some knowledge representation and comprehensibility issues that have traditionally prevented efficient solutions.

The work described here – on *Inference with Classifiers* – can be viewed as a concrete instantiation of the Learning to Reason framework; it addresses a second important aspect of a unified theory of learning and reasoning, the one which stems from the fact that, inherently, inferences in some domains involve

a large number of predictors that interact in different ways. The fundamental issue addressed is that of systematically combine, chain and perform inferences with the outcome of a large number of mutually dependent learned predictors.

We will discuss several well known inference paradigms, and show how to use those for inference with classifiers. Namely, we will use these inference paradigms to develop inference algorithms that take as input outcomes of classifiers and provide coherent inferences that satisfy some domain or problem specific constraints. Some of the inference paradigms used are hidden Markov models (HMMs), conditional probabilistic models [8, 3], loopy Bayesian networks [6, 11], constraint satisfaction [8, 5] and Markov random fields [1].

Research in this direction may offer several benefits over direct use of classifiers or simply using traditional inference models. One benefit is the ability to directly use powerful classifiers to represent domain variables that are of interest in the inference stage. Advantages of this view have been observed in the speech recognition community when neural network based classifiers were combined within an HMM based inference approach, and have been quantified also in [8]. A second key advantage stems from the fact that only a few of the domain variables are actually of any interest at the inference stage. Performing inference with outcomes of classifiers allows for abstracting away a large number of the domain variables (which will be used only to define the classifiers' outcomes) and will be beneficial also computationally.

Research in this direction offers several challenges to AI and Machine Learning researchers. One of the key challenges of this direction from the machine learning perspective is to understand how the presence of constraints on the outcomes of classifiers can be systematically analyzed and exploited in order to derive better learning algorithms and for reducing the number of labeled examples required for learning.

References

1. D. Heckerman, D. M. Chickering, C. Meek, R. Rounthwaite, and C. M. Kadie. Dependency networks for inference, collaborative filtering, and data visualization. *Journal of Machine Learning Research*, 1:49–75, 2000.
2. R. Khardon and D. Roth. Learning to reason. *Journal of the ACM*, 44(5):697–725, Sept. 1997.
3. J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning*, Williamstown, MA, 2001.
4. J. McCarthy. Programs with common sense. In R. Brachman and H. Levesque, editors, *Readings in Knowledge Representation, 1985*. Morgan-Kaufmann, 1958.
5. M. Munoz, V. Punyakanok, D. Roth, and D. Zimak. A learning approach to shallow parsing. In *EMNLP-VLC'99, the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 168–178, June 1999.
6. K. P. Murphy, Y. Weiss, and M. I. Jordan. Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of Uncertainty in AI*, pages 467–475, 1999.

7. N. J. Nilsson. Logic and artificial intelligence. *Artificial Intelligence*, 47:31–56, 1991.
8. V. Punyakanok and D. Roth. The use of classifiers in sequential inference. In *NIPS-13; The 2000 Conference on Advances in Neural Information Processing Systems*, pages 995–1001. MIT Press, 2001.
9. D. Roth. Learning to reason: The non-monotonic case. In *Proc. of the International Joint Conference on Artificial Intelligence*, pages 1178–1184, 1995.
10. D. Roth. Learning to resolve natural language ambiguities: A unified approach. In *Proc. of the American Association of Artificial Intelligence*, pages 806–813, 1998.
11. D. Roth and W.-T. Yih. Probabilistic reasoning for entity and relation recognition. In *COLING 2002, The 19th International Conference on Computational Linguistics*, 2002.
12. L. G. Valiant. Robust logic. In *Proceedings of the Annual ACM Symp. on the Theory of Computing*, 1999.

A Kernel Approach for Learning from Almost Orthogonal Patterns

Bernhard Schölkopf¹, Jason Weston¹, Eleazar Eskin², Christina Leslie², and William Stafford Noble^{2,3}

¹ Max-Planck-Institut für biologische Kybernetik, Spemannstr. 38,
D-72076 Tübingen, Germany

{bernhard.schoelkopf, jason.weston}@tuebingen.mpg.de

² Department of Computer Science

Columbia University, New York

{eeskin, cleslie, noble}@cs.columbia.edu

³ Columbia Genome Center

Columbia University, New York

Abstract. In kernel methods, all the information about the training data is contained in the Gram matrix. If this matrix has large diagonal values, which arises for many types of kernels, then kernel methods do not perform well. We propose and test several methods for dealing with this problem by reducing the dynamic range of the matrix while preserving the positive definiteness of the Hessian of the quadratic programming problem that one has to solve when training a Support Vector Machine.

1 Introduction

Support Vector Machines (SVM) and related kernel methods can be considered an approximate implementation of the structural risk minimization principle suggested by Vapnik (1979). To this end, they minimize an objective function containing a trade-off between two goals, that of minimizing the training error, and that of minimizing a regularization term. In SVMs, the latter is a function of the margin of separation between the two classes in a binary pattern recognition problem. This margin is measured in a so-called feature space $\mathcal{H}$ which is a Hilbert space into which the training patterns are mapped by means of a map

$$\Phi : \mathcal{X} \rightarrow \mathcal{H}. \quad (1)$$

Here, the input domain $\mathcal{X}$ can be an arbitrary nonempty set. The art of designing an SVM for a task at hand consist of selecting a feature space with the property that dot products between mapped input points, $\langle \Phi(x), \Phi(x') \rangle$, can be computed in terms of a so-called *kernel*

$$k(x, x') = \langle \Phi(x), \Phi(x') \rangle \quad (2)$$

which can be evaluated efficiently. Such a kernel necessarily belongs to the class of *positive definite kernels* (e.g. Berg et al. (1984)), i.e., it satisfies

$$\sum_{i,j=1}^m a_i a_j k(x_i, x_j) \geq 0 \quad (3)$$

for all $a_i \in \mathbb{R}, x_i \in \mathcal{X}, i = 1, \dots, m$. The kernel can be thought of as a nonlinear similarity measure that corresponds to the dot product in the associated feature space. Using k , we can carry out all algorithms in $\mathcal{H}$ that can be cast in terms of dot products, examples being SVMs and PCA (for an overview, see Schölkopf and Smola (2002)).

To train a hyperplane classifier in the feature space,

$$f(x) = \text{sgn}(\langle \mathbf{w}, \Phi(x) \rangle + b), \quad (4)$$

where $\mathbf{w}$ is expanded in terms of the points $\Phi(x_j)$,

$$\mathbf{w} = \sum_{j=1}^m a_j \Phi(x_j), \quad (5)$$

the SVM pattern recognition algorithm minimizes the quadratic form⁴

$$\|\mathbf{w}\|^2 = \sum_{i,j=1}^m a_i a_j K_{ij} \quad (6)$$

subject to the constraints

$$y_i [\langle \Phi(x_i), \mathbf{w} \rangle + b] \geq 1, \text{ i.e., } y_i \left[\sum_{j=1}^m a_j K_{ij} + b \right] \geq 1 \quad (7)$$

and

$$y_i a_i \geq 0 \quad (8)$$

for all $i \in \{1, \dots, m\}$. Here,

$$(x_1, y_1), \dots, (x_m, y_m) \in \mathcal{X} \times \{\pm 1\} \quad (9)$$

are the training examples, and

$$K_{ij} := k(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle \quad (10)$$

is the Gram matrix.

Note that the regularizer (6) equals the squared length of the weight vector $\mathbf{w}$ in $\mathcal{H}$. One can show that $\|\mathbf{w}\|$ is inversely proportional to the margin of

⁴ We are considering the zero training error case. Nonzero training errors are incorporated as suggested by Cortes and Vapnik (1995). Cf. also Osuna and Girosi (1999).

separation between the two classes, hence minimizing it amounts to maximizing the margin. Sometimes, a modification of this approach is considered, where the regularizer

$$\sum_{i=1}^m a_i^2 \quad (11)$$

is used instead of (6). Whilst this is no longer the squared length of a weight vector in the feature space $\mathcal{H}$, it is instructive to re-interpret it as the squared length in a different feature space, namely in $\mathbb{R}^m$.

To this end, we consider the feature map

$$\Phi_m(x) := (k(x, x_1), \dots, k(x, x_m))^\top, \quad (12)$$

sometimes called the *empirical kernel map* (Tsuda, 1999; Schölkopf and Smola, 2002). In this case, the SVM optimization problem consists in minimizing

$$\|\mathbf{a}\|^2 \quad (13)$$

subject to

$$y_i [\langle \Phi_m(x_i), \mathbf{a} \rangle + b] \geq 1 \quad (14)$$

for all $i \in \{1, \dots, m\}$, where $\mathbf{a} = (a_1, \dots, a_m)^\top \in \mathbb{R}^m$. In view of (12), however, the constraints (14) are equivalent to $y_i \left[\sum_{j=1}^m a_j K_{ij} + b \right] \geq 1$, i.e. to (7), while the regularizer $\|\mathbf{a}\|^2$ equals (11).

Therefore, using the regularizer (11) and the original kernel essentially⁵ corresponds to using a standard SVM with the empirical kernel map. This SVM operates in an m -dimensional feature space with the standard SVM regularizer, i.e., the squared weight of the weight vector in the feature space. We can thus train a classifier using the regularizer (11) simply by using an SVM with the kernel

$$k_m(x, x') := \langle \Phi_m(x), \Phi_m(x') \rangle, \quad (15)$$

and thus, by definition of Φ_m , using the Gram matrix

$$K_m = K K^\top, \quad (16)$$

where K denotes the Gram matrix of the original kernel. The last equation shows that when employing the empirical kernel map, it is not necessary to use a positive definite kernel. The reason is that no matter what K is, the Gram matrix $K K^\top$ is always positive definite,⁶ which is sufficient for an SVM.

The remainder of the paper is structured as follows. In Section 2, we introduce the problem of large diagonals, followed by our proposed method to handle it (Section 3). Section 4 presents experiments, and Section 5 summarizes our conclusions.

⁵ disregarding the positivity constraints (8)

⁶ Here, as in (3), we allow for a nonzero null space in our usage of the concept of positive definiteness.

2 Orthogonal Patterns in the Feature Space

An important feature of kernel methods is that the input domain $\mathcal{X}$ does not have to be a vector space. The inputs might just as well be discrete objects such as strings. Moreover, the map Φ might compute rather complex features of the inputs. Examples thereof are polynomial kernels (Boser et al., 1992), where Φ computes all products (of a given order) of entries of the inputs (in this case, the inputs are vectors), and string kernels (Watkins, 2000; Haussler, 1999; Lodhi et al., 2002), which, for instance, can compute the number of common substrings (not necessarily contiguous) of a certain length $n \in \mathbb{N}$ of two strings x, x' in $O(n|x||x'|)$ time. Here, we assume that x and x' are two finite strings over a finite alphabet Σ . For the string kernel of order n , a basis for the feature space consists of the set of all strings of length n , Σ^n . In this case, Φ maps a string x into a vector whose entries indicate whether the respective string of length n occurs as a substring in x . By construction, these will be rather sparse vectors — a large number of *possible* substrings do not occur in a given string. Therefore, the dot product of two *different* vectors will take a value which is much smaller than the dot product of a vector with itself. This can also be understood as follows: any string shares *all* substrings with itself, but relatively few substrings with another string. Therefore, it will typically be the case that we are faced with *large diagonals*. By this we mean that, given some training inputs $x_1, \dots, x_m$, we have⁷

$$k(x_i, x_i) \gg |k(x_i, x_j)| \text{ for } x_i \neq x_j, i, j \in \{1, \dots, m\}. \quad (17)$$

In this case, the associated Gram matrix will have large diagonal elements.⁸

Let us next consider an innocuous application which is rather popular with SVMs: handwritten digit recognition. We suppose that the data are handwritten characters represented by images in $[0, 1]^N$ (here, $N \in \mathbb{N}$ is the number of pixels), and that only a small fraction of the images is ink (i.e. few entries take the value 1). In that case, we typically have $\langle x, x \rangle > \langle x, x' \rangle$ for $x \neq x'$, and thus the polynomial kernel (which is what most commonly is used for SVM handwritten digit recognition)

$$k(x, x') = \langle x, x' \rangle^d \quad (18)$$

satisfies $k(x, x) \gg |k(x, x')|$ already for moderately large d — it has large diagonals.

Note that as in the case of the string kernel, one can also understand this phenomenon in terms of the sparsity of the vectors in the feature space. It is

⁷ The diagonal terms $k(x_i, x_i)$ are necessarily nonnegative for positive definite kernels, hence no modulus on the left hand side.

⁸ In the machine learning literature, the problem is sometimes referred to as *diagonal dominance*. However, the latter term is used in linear algebra for matrices where the absolute value of each diagonal element is greater than the sum of the absolute values of the other elements in its row (or column). Real diagonally dominant matrices with positive diagonal elements are positive definite.

known that the polynomial kernel of order d effectively maps the data into a feature space whose dimensions are spanned by all products of d pixels. Clearly, if some of the pixels take the value zero to begin with, then an even larger fraction of all possible products of d pixels (assuming $d > 1$) will be zero. Therefore, the sparsity of the vectors will increase with d .

In practice, it has been observed that SVMs do not work well in this situation. Empirically, they work much better if the images are scaled such that the individual pixel values are in $[-1, 1]$, i.e., that the background value is -1 . In this case, the data vectors are less sparse and thus further from being orthogonal.

Indeed, large diagonals correspond to approximate orthogonality of any two different patterns mapped into the feature space. To see this, assume that $x \neq x'$ and note that due to $k(x, x) \gg |k(x, x')|$,

$$\begin{aligned} \cos(\angle(\Phi(x), \Phi(x'))) &= \frac{\langle \Phi(x), \Phi(x') \rangle}{\sqrt{\langle \Phi(x), \Phi(x) \rangle \langle \Phi(x'), \Phi(x') \rangle}} \\ &= \frac{k(x, x')}{\sqrt{k(x, x)k(x', x')}} \approx 0 \end{aligned}$$

In some cases, an SVM trained using a kernel with large diagonals will *memorize* the data. Let us consider a simple toy example, using X as data matrix and Y as label vector, respectively:

$$X = \begin{pmatrix} 1 & 0 & 0 & 9 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 8 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 9 & 0 & 0 \\ 0 & 0 & 9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 8 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 9 & 0 \end{pmatrix}, \quad Y = \begin{pmatrix} +1 \\ +1 \\ +1 \\ -1 \\ -1 \\ -1 \end{pmatrix}$$

The Gram matrix for these data (using the linear kernel $k(x, x') = \langle x, x' \rangle$) is

$$K = \begin{pmatrix} 82 & 1 & 1 & 0 & 0 & 0 \\ 1 & 65 & 1 & 0 & 0 & 0 \\ 1 & 1 & 82 & 0 & 0 & 0 \\ 0 & 0 & 0 & 81 & 0 & 0 \\ 0 & 0 & 0 & 0 & 64 & 0 \\ 0 & 0 & 0 & 0 & 0 & 81 \end{pmatrix}.$$

A standard SVM finds the solution $f(x) = \text{sgn}(\langle w, x \rangle + b)$ with

$$w = (0.04, 0, -0.11, 0.11, 0, 0.12, -0.12, 0.11, 0, -0.11)^\top, \quad b = -0.02.$$

It can be seen from the coefficients of the weight vector w that this solution has but memorized the data: all the entries which are larger than 0.1 in absolute value correspond to dimensions which are nonzero only for *one* of the training points. We thus end up with a look-up table. A *good* solution for a linear classifier, on the other hand, would be to just choose the first feature, e.g., $f(x) = \text{sgn}(\langle w, x \rangle + b)$, with $w = (2, 0, 0, 0, 0, 0, 0, 0, 0, 0)^\top$, $b = -1$.

3 Methods to Reduce Large Diagonals

The basic idea that we are proposing is very simple indeed. We would like to use a nonlinear transformation to reduce the size of the diagonal elements, or, more generally, to reduce the dynamic range of the Gram matrix entries. The only difficulty is that if we simply do this, we have no guarantee that we end up with a Gram matrix that is still positive definite. To ensure that it is, we can use methods of functional calculus for matrices. In the experiments we will mainly use a simple special case of the below. Nevertheless, let us introduce the general case, since we think it provides a useful perspective on kernel methods, and on the transformations that can be done on Gram matrices.

Let K be a symmetric $m \times m$ matrix with eigenvalues in $[\lambda_{\min}, \lambda_{\max}]$, and f a continuous function on $[\lambda_{\min}, \lambda_{\max}]$. Functional calculus provides a unique symmetric matrix, denoted by $f(K)$, with eigenvalues in $[f(\lambda_{\min}), f(\lambda_{\max})]$. It can be computed via a Taylor series expansion in K , or using the eigenvalue decomposition of K : If $K = S^\top D S$ (with D diagonal and S unitary), then $f(K) = S^\top f(D) S$, where $f(D)$ is the diagonal matrix with $f(D)_{ii} = f(D_{ii})$.

The convenient property of this procedure is that we can treat functions of symmetric matrices just like functions on $\mathbb{R}$; in particular, we have, for $\alpha \in \mathbb{R}$, and real continuous functions f, g defined on $[\lambda_{\min}, \lambda_{\max}]$,⁹

$$\begin{aligned} (\alpha f + g)(K) &= \alpha f(K) + g(K) \\ (fg)(K) &= f(K)g(K) = g(K)f(K) \\ \|f\|_{\infty, \sigma(K)} &= \|f(K)\| \\ \sigma(f(K)) &= f(\sigma(K)). \end{aligned}$$

In technical terms, the C^* -algebra generated by K is isomorphic to the set of continuous functions on $\sigma(K)$.

For our problems, functional calculus can be applied in the following way. We start off with a positive definite matrix K with large diagonals. We then reduce its dynamic range by elementwise application of a nonlinear function, such as $\varphi(x) = \log(x + 1)$ or $\varphi(x) = \text{sgn}(x) \cdot |x|^p$ with $0 < p < 1$. This will lead to a matrix which may no longer be positive definite. However, it is still symmetric, and hence we can apply functional calculus. As a consequence of $\sigma(f(K)) = f(\sigma(K))$, we just need to apply a function f which maps to $\mathbb{R}_0^+$. This will ensure that all eigenvalues of $f(K)$ are nonnegative, hence $f(K)$ will be positive definite. One can use these observations to design the following scheme.

For positive definite K ,

1. compute the positive definite matrix $A := \sqrt{K}$
2. reduce the dynamic range of the entries of A by applying an elementwise transformation φ , leading to a symmetric matrix A_φ
3. compute the positive definite matrix $K' := (A_\varphi)^2$ and use it in subsequent processing. The entries of K' will be the “effective kernel,” which in this case is no longer given in analytic form.

⁹ Below, $\sigma(K)$ denotes the spectrum of K .

Note that in this procedure, if φ is the identity, then we have $K = K'$.

Experimentally, this scheme works rather well. However, it has one downside: since we no longer have the kernel function in analytic form, our only means of evaluating it is to include all test inputs (not the test labels, though) into the matrix K . In other words, K should be the Gram matrix computed from the observations $x_1, \dots, x_{m+n}$ where $x_{m+1}, \dots, x_{m+n}$ denote the test inputs. We thus need to know the test inputs already during training. This setting is sometimes referred to as *transduction* (Vapnik, 1998).

If we skip the step of taking the square root of K , we can alleviate this problem. In that case, the only application of functional calculus left is a rather trivial one, that of computing the square of K . The $m \times m$ submatrix of K^2 which in this case would have to be used for training then equals the Gram matrix when using the empirical kernel map

$$\Phi_{m+n}(x) = (k(x, x_1), \dots, k(x, x_{m+n}))^\top. \quad (19)$$

For the purposes of computing dot products, however, this can approximately be replaced by the empirical kernel map in terms of the training examples only, i.e., by (12). The justification for this is that for large $r \in \mathbb{N}$, $\frac{1}{r} \langle \Phi_r(x), \Phi_r(x') \rangle \approx \int_{\mathcal{X}} k(x, x'') k(x', x'') dP(x'')$, where P is assumed to be the distribution of the inputs. Therefore, we have $\frac{1}{m} \langle \Phi_m(x), \Phi_m(x') \rangle \approx \frac{1}{m+n} \langle \Phi_{m+n}(x), \Phi_{m+n}(x') \rangle$. Altogether, the procedure then boils down to simply training an SVM using the empirical kernel map in terms of the training examples and the transformed kernel function $\varphi(k(x, x'))$. This is what we will use in the experiments below.¹⁰

4 Experiments

4.1 Artificial Data

We first constructed a set of artificial experiments which produce kernels exhibiting large diagonals. The experiments are as follows: a string classification problem, a microarray cancer detection problem supplemented with extra noisy features and a toy problem whose labels depend upon hidden variables; the visible variables are nonlinear combinations of those hidden variables.

String Classification We considered the following classification problem. Two classes of strings are generated with equal probability by two different Markov models. Both classes of strings consist of letters from the same alphabet of $a = 20$ letters, and strings from both classes are always of length $n = 20$. Strings from the negative class are generated by a model where transitions from any letter to any other letter are equally likely. Strings from the positive class are generated by a model where transitions from one letter to itself (so the next letter is the same as the last) have probability 0.43, and all other transitions have probability 0.03. For both classes the starting letter of any string is equally likely to be any

¹⁰ For further experimental details, cf. Weston and Schölkopf (2001).

letter of the alphabet. The task then is to predict which class a given string belongs to. To map these strings into a feature space, we used the string kernel described above, computing a dot product product in a feature space consisting of all subsequences of length l . In the present application, the subsequences are weighted by an exponentially decaying factor λ of their full length in the text, hence emphasizing those occurrences which are close to contiguous. A method of computing this kernel efficiently using a dynamic programming technique is described by Lodhi et al. (2002). For our problem we chose the parameters $l = 3$ and $\lambda = \frac{1}{4}$.

We generated 50 such strings and used the string subsequence kernel with $\lambda = 0.25$.¹¹ We split the data into 25 for training and 25 for testing in 20 separate trials. We measured the success of a method by calculating the mean classification loss on the test sets. Figure 1 shows four strings from the dataset and the computed kernel matrix for these strings¹². Note that the diagonal entries are much larger than the off-diagonals because a long string has a large number of subsequences that are shared with no other strings in the dataset apart from itself. However, information relevant to the classification of the strings is contained in the matrix. This can be seen by computing the mean kernel value between two examples of the positive class which is equal to 0.0003 ± 0.0011 , whereas the mean kernel value between two examples of opposite classes is 0.00002 ± 0.00007 . Although the numbers are very small, this captures that the positive class have more in common with each other than with random strings (they are more likely to have repeated letters).

string	class	
qqbqqnshrtktfhhaahhh	+ve	$K = \begin{pmatrix} 0.6183 & 0.0133 & 0.0000 & 0.0000 \\ 0.0133 & 1.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.4692 & 0.0002 \\ 0.0000 & 0.0000 & 0.0002 & 0.4292 \end{pmatrix}$
abajahnaajjjiiittt	+ve	
sdolncqniflmpcrioog	-ve	
reaqhcoigegalqjdsdgs	-ve	

Fig. 1. Four strings and their kernel matrix using the string subsequence kernel with $\lambda = 0.25$. Note that the diagonal entries are much larger than the off-diagonals because a long string has a large number of subsequences that are shared with no other strings in the dataset apart from itself.

If the original kernel is denoted as a dot product $k(x, y) = \langle \Phi(x), \Phi(y) \rangle$, then we employ the kernel $k(x, y) = \langle \Phi(x), \Phi(y) \rangle^p$ where $0 < p < 1$ to solve the diagonal dominance problem. We will refer to this kernel as a *subpolynomial* one. As this kernel may no longer be positive definite we use the method described in

¹¹ We note that introducing nonlinearities using an RBF kernel with respect to the distances generated by the subsequence kernel can improve results on this problem, but we limit our experiments to ones performed in the linear space of features generated by the subsequence kernel.

¹² Note, the matrix was rescaled by dividing by the largest entry.

Table 1. Results of using the string subsequence kernel on a string classification problem (top row). The remaining rows show the results of using the subpolynomial kernel to deal with the large diagonal.

kernel method		classification loss
original k , $k(x, y) = \langle \Phi(x), \Phi(y) \rangle$		0.36 ± 0.13
$k_{emp}(x, y) = \langle \Phi(x), \Phi(y) \rangle^p$	p=1	0.30 ± 0.08
	p=0.9	0.25 ± 0.09
	p=0.8	0.20 ± 0.10
	p=0.7	0.15 ± 0.09
	p=0.6	0.13 ± 0.07
	p=0.5	0.14 ± 0.06
	p=0.4	0.15 ± 0.07
	p=0.3	0.15 ± 0.06
	p=0.2	0.17 ± 0.07
	p=0.1	0.21 ± 0.09

Section 1, employing the empirical kernel map to embed our distance measure into a feature space. Results of using our method to solve the problem of large diagonals is given in Table 1. The method provides, with the optimum choice of the free parameter, a reduction from a loss of 0.36 ± 0.13 with the original kernel to 0.13 ± 0.07 with $p=0.6$. Although we do not provide methods for choosing this free parameter, it is straight-forward to apply conventional techniques of model selection (such as cross validation) to achieve this goal.

We also performed some further experiments which we will briefly discuss. To check that the result is a feature of kernel algorithms, and not something peculiar to SVMs, we also applied the same kernels to another algorithm, kernel 1-nearest neighbor. Using the original kernel matrix yields a loss of 0.43 ± 0.06 whereas the subpolynomial method again improves the results, using $p = 0.6$ yields 0.22 ± 0.08 and $p = 0.3$ (the optimum choice) yields 0.17 ± 0.07 . Finally, we tried some alternative proposals for reducing the large diagonal effect. We tried using Kernel PCA to extract features as a pre-processing to training an SVM. The intuition behind using this is that features contributing to the large diagonal effect may have low variance and would thus be removed by KPCA. KPCA did improve performance a little, but did not provide results as good as the subpolynomial method. The best result was found by extracting 15 features (from the kernel matrix of 50 examples) yielding a loss of 0.23 ± 0.07 .

Microarray Data With Added Noise We next considered the microarray classification problem of Alon et al. (1999) (see also Guyon et al. (2001) for a treatment of this problem with SVMs). In this problem one must distinguish between cancerous and normal tissue in a colon cancer problem given the expression of genes measured by microarray technology. In this problem one does not encounter large diagonals, however we augmented the original dataset with extra noisy features to simulate such a problem. The original data has 62 ex-

amples (22 positive, 40 negative) and 2000 features (gene expression levels of the tissues samples). We added a further 10,000 features to the dataset, such that for each example a randomly chosen 100 of these features are chosen to be nonzero (taking a random value between 0 and 1) and the rest are equal to zero. This creates a kernel matrix with large diagonals. In Figure 2 we show the first 4×4 entries of the kernel matrix of a linear kernel before and after adding the noisy features.

The problem is again an artificial one demonstrating the problem of large diagonals, however this time the feature space is rather more explicit rather than the implicit one induced by string kernels. In this problem we can clearly see the large diagonal problem is really a special kind of feature selection problem. As such, feature selection algorithms should be able to help improve generalize ability, unfortunately most feature selection algorithms work on explicit features rather than implicit ones induced by kernels.

Performance of methods was measured using 10 fold cross validation, which was repeated 10 times. Due to the unbalanced nature of the number of positive and negative examples in this data set we measured the error rates using a balanced loss function with the property that chance level is a loss of 0.5, regardless of the ratio of positive to negative examples. On this problem (with the added noise) an SVM using the original kernel does not perform better than chance. The results of using the original kernel and the subpolynomial method are given in Table 2. The subpolynomial kernel leads to a large improvement over using the original kernel. Its performance is close to that of an SVM on the original data without the added noise, which in this case is 0.18 ± 0.15 .

Hidden Variable Problem We then constructed an artificial problem where the labels can be predicted by a linear rule based upon some hidden variables. However, the visible variables are a nonlinear combination of the hidden variables combined with noise. The purpose is to show that the subpolynomial kernel is not only useful in the case of matrices with large diagonals: it can also improve results in the case where a *linear* rule already overfits. The data are generated as follows. There are 10 hidden variables: each class $y \in \{\pm 1\}$ is generated by a 10 dimensional normal distribution $N(\mu, \sigma)$ with variance $\sigma^2 = 1$, and mean $\mu = y(0.5, 0.5, \dots, 0.5)$. We then add 10 more (noisy) features for each example, each generated with $N(0, 1)$. Let us denote the 20-dimensional vector obtained

$$K = \begin{pmatrix} 1.00 & 0.41 & 0.33 & 0.42 \\ 0.41 & 1.00 & 0.17 & 0.39 \\ 0.33 & 0.17 & 1.00 & 0.61 \\ 0.42 & 0.39 & 0.61 & 1.00 \end{pmatrix}, \quad K' = \begin{pmatrix} 39.20 & 0.41 & 0.33 & 0.73 \\ 0.41 & 37.43 & 0.26 & 0.88 \\ 0.33 & 0.26 & 31.94 & 0.61 \\ 0.73 & 0.88 & 0.61 & 35.32 \end{pmatrix}$$

Fig. 2. The first 4×4 entries of the kernel matrix of a linear kernel on the colon cancer problem before (K) and after (K') adding 10,000 sparse, noisy features. The added features are designed to create a kernel matrix with a large diagonal.

Table 2. Results of using a linear kernel on a colon cancer classification problem with added noise (top row). The remaining rows show the results of using the subpolynomial kernel to deal with the large diagonal.

kernel method	balanced loss
original k , $k(x, y) = \langle x, y \rangle$	0.49 ± 0.05
$k_{emp}(x, y) = \text{sgn} \langle x, y \rangle \cdot \langle x, y \rangle ^p$ p=0.95	0.35 ± 0.17
p=0.9	0.30 ± 0.17
p=0.8	0.25 ± 0.18
p=0.7	0.22 ± 0.17
p=0.6	0.23 ± 0.17
p=0.5	0.25 ± 0.19
p=0.4	0.28 ± 0.19
p=0.3	0.29 ± 0.18
p=0.2	0.30 ± 0.19
p=0.1	0.31 ± 0.18

this way for example i as h_i . The visible variables x_i are then constructed by taking all monomials of degree 1 to 4 of h_i . It is known that dot products between such vectors can be computed using polynomial kernels (Boser et al., 1992), thus the dot product between two visible variables is

$$k(x_i, x_j) = (\langle h_i, h_j \rangle + 1)^4.$$

We compared the subpolynomial method to a linear kernel using balanced 10-fold cross validation, repeated 10 times. The results are shown in Table 3. Again, the subpolynomial kernel gives improved results.

One interpretation of these results is that if we know that the visible variables are polynomials of some hidden variables, then it makes sense to use a subpolynomial transformation to obtain a Gram matrix closer to the one we could compute if we were given the hidden variables. In effect, the subpolynomial kernel can (approximately) extract the hidden variables.

4.2 Real Data

Thrombin Binding Problem In the thrombin dataset the problem is to predict whether a given drug binds to a target site on thrombin, a key receptor in blood clotting. This dataset was used in the KDD (Knowledge Discovery and Data Mining) Cup 2001 competition and was provided by DuPont Pharmaceuticals.

In the training set there are 1909 examples representing different possible molecules (drugs), 42 of which bind. Hence the data is rather unbalanced in this respect. Each example has a fixed length vector of 139,351 binary features (variables) in $\{0, 1\}$ which describe three-dimensional properties of the molecule. An important characteristic of the data is that very few of the feature entries are nonzero (0.68% of the 1909×139351 training matrix, see (Weston et al., 2002) for

Table 3. Results of using a linear kernel on the hidden variable problem (top row). The remaining rows show the results of using the subpolynomial kernel to deal with the large diagonal.

kernel method	classification loss
original k , $k(x, y) = \langle x, y \rangle$	0.26 ± 0.12
$k_{emp}(x, y) = \text{sgn} \langle x, y \rangle \cdot \langle x, y \rangle ^p$ p=1	0.25 ± 0.12
p=0.9	0.23 ± 0.13
p=0.8	0.19 ± 0.12
p=0.7	0.18 ± 0.12
p=0.6	0.16 ± 0.11
p=0.5	0.16 ± 0.11
p=0.4	0.16 ± 0.11
p=0.3	0.18 ± 0.11
p=0.2	0.20 ± 0.12
p=0.1	0.19 ± 0.13

further statistical analysis of the dataset). Thus, many of the features somewhat resemble the noisy features that we added on to the colon cancer dataset to create a large diagonal in Section 4.1. Indeed, constructing a kernel matrix of the training data using a linear kernel yields a matrix with a mean diagonal element of 1377.9 ± 2825 and a mean off-diagonal element of 78.7 ± 209 . We compared the subpolynomial method to the original kernel using 8-fold balanced cross validation (ensuring an equal number of positive examples were in each fold). The results are given in Table 4. Once again the subpolynomial method provides improved generalization. It should be noted that feature selection and transduction methods have also been shown to improve results, above that of a linear kernel on this problem (Weston et al., 2002).

Table 4. Results of using a linear kernel on the thrombin binding problem (top row). The remaining rows show the results of using the subpolynomial kernel to deal with the large diagonal.

kernel method	balanced loss
original k , $k(x, y) = \langle x, y \rangle$	0.30 ± 0.12
$k_{emp}(x, y) = \langle x, y \rangle^p$ p=0.9	0.24 ± 0.10
p=0.8	0.24 ± 0.10
p=0.7	0.18 ± 0.09
p=0.6	0.18 ± 0.09
p=0.5	0.15 ± 0.09
p=0.4	0.17 ± 0.10
p=0.3	0.17 ± 0.10
p=0.2	0.18 ± 0.10
p=0.1	0.22 ± 0.15

Table 5. Results of using a linear kernel on the Lymphoma classification problem (top row). The remaining rows show the results of using the subpolynomial kernel to deal with the large diagonal.

kernel method	balanced loss
original k , $k(x, y) = \langle x, y \rangle$	0.043 ± 0.08
$k_{emp}(x, y) = \text{sgn} \langle x, y \rangle \cdot \langle x, y \rangle ^p$ $p=1$	0.037 ± 0.07
$p=0.9$	0.021 ± 0.05
$p=0.8$	0.016 ± 0.05
$p=0.7$	0.015 ± 0.05
$p=0.6$	0.022 ± 0.06
$p=0.5$	0.022 ± 0.06
$p=0.4$	0.042 ± 0.07
$p=0.3$	0.046 ± 0.08
$p=0.2$	0.083 ± 0.09
$p=0.1$	0.106 ± 0.09

Lymphoma Classification We next looked at the problem of identifying large B-Cell Lymphoma by gene expression profiling (Alizadeh et al., 2000). In this problem the gene expression of 96 samples is measured with microarrays to give 4026 features. Sixty-one of the samples are in classes "DLCL", "FL" or "CLL" (malignant) and 35 are labelled "otherwise" (usually normal). Although the data does not induce a kernel matrix with a very large diagonal it is possible that the large number of features induce overfitting even in a linear kernel. To examine if our method would still help in this situation we applied the same techniques as before, this time using balanced 10-fold cross validation, repeated 10 times, and measuring error rates using the balanced loss. The results are given in Table 5. The improvement given by the subpolynomial kernel suggests that overfitting in linear kernels when the number of features is large may be overcome by applying special feature maps. It should be noted that (explicit) feature selection methods have also been shown to improve results on this problem, see e.g. Weston et al. (2001).

Protein Family Classification We then focussed on the problem of classifying protein domains into superfamilies in the Structural Classification of Proteins (SCOP) database version 1.53 (Murzin et al., 1995). We followed the same problem setting as Liao and Noble (2002): sequences were selected using the Astral database (astral.stanford.edu cite), removing similar sequences using an E-value threshold of 10^{-25} . This procedure resulted in 4352 distinct sequences, grouped into families and superfamilies. For each family, the protein domains within the family are considered positive test examples, and the protein domains outside the family but within the same superfamily are taken as positive training examples. The data set yields 54 families containing at least 10 family members (positive training examples). Negative examples are taken from outside of the positive sequence's fold, and are randomly split into train and test sets in

the same ratio as the positive examples. Details about the various families are listed in (Liao and Noble, 2002), and the complete data set is available at www.cs.columbia.edu/compbio/svm-pairwise. The experiments are characterized by small positive (training and test) sets and large negative sets. Note that this experimental setup is similar to that used by Jaakkola et al. (2000), except the positive training sets do not include additional protein sequences extracted from a large, unlabeled database, which amounts to a kind of “transduction” (Vapnik, 1998) algorithm.¹³

An SVM requires fixed length vectors. Proteins, of course, are variable-length sequences of amino acids and hence cannot be directly used in an SVM. To solve this task we used a sequence kernel, called the spectrum kernel, which maps strings into a space of features which correspond to every possible k -mer (sequence of k letters) with at most m mismatches, weighted by prior probabilities (Leslie et al., 2002). In this experiment we chose $k = 3$ and $m = 0$. This kernel is then normalized so that each vector has length 1 in the feature space; i.e.,

$$k(x, x') = \frac{\langle x, x' \rangle}{\sqrt{\langle x, x \rangle \langle x', x' \rangle}}. \quad (20)$$

An asymmetric soft margin is implemented by adding to the diagonal of the kernel matrix a value $0.02 \cdot \rho$, where ρ is the fraction of training set sequences that have the same label as the current sequence (see Cortes and Vapnik (1995); Brown et al. (2000) for details). For comparison, the same SVM parameters are used to train an SVM using the Fisher kernel (Jaakkola and Haussler (1999); Jaakkola et al. (2000), see also Tsuda et al. (2002)), another possible kernel choice. The Fisher kernel is currently considered one of the most powerful homology detection methods. This method combines a generative, profile hidden Markov model (HMM) and uses it to generate a kernel for training an SVM. A protein’s vector representation induced by the kernel is its gradient with respect to the profile hidden Markov model, the parameters of which are found by expectation-maximization.

For each method, the output of the SVM is a discriminant score that is used to rank the members of the test set. Each of the above methods produces as output a ranking of the test set sequences. To measure the quality of this ranking, we use two different scores: receiver operating characteristic (ROC) scores and the median rate of false positives (RFP). The ROC score is the normalized area under a curve that plots true positives as a function of false positives for varying classification thresholds. A perfect classifier that puts all the positives at the top of the ranked list will receive an ROC score of 1, and for these data, a random classifier will receive an ROC score very close to 0. The median RFP score is the fraction of negative test sequences that score as high or better

¹³ We believe that it is this transduction step which may be responsible for much of the success of using the methods described by Jaakkola et al. (2000)). However, to make a fair comparison of kernel methods we do not include this step which could potentially be included in any of the methods. Studying the importance of transduction remains a subject of further research.

Table 6. Results of using the spectrum kernel with $k = 3$, $m = 0$ on the SCOP dataset (top row). The remaining rows (apart from the last one) show the results of using the subpolynomial kernel to deal with the large diagonal. The last row, for comparison, shows the performance of an SVM using the Fisher kernel.

kernel method		RFP	ROC
original k , $k(\Phi(x), \Phi(y)) = \langle x, y \rangle$		0.1978	0.7516
$k_{emp}(x, y) = \langle \Phi(x), \Phi(y) \rangle^p$	p=0.5	0.1697	0.7967
	p=0.4	0.1569	0.8072
	p=0.3	0.1474	0.8183
	p=0.2	0.1357	0.8251
	p=0.1	0.1431	0.8213
	p=0.05	0.1489	0.8156
SVM-FISHER		0.2946	0.6762

than the median-scoring positive sequence. RFP scores were used by Jaakkola *et al.* in evaluating the Fisher-SVM method. The results of using the spectrum kernel, the subpolynomial kernel applied to the spectrum kernel and the fisher kernel are given in Table 6. The mean ROC and RFP scores are superior for the subpolynomial kernel. We also show a family-by-family comparison of the subpolynomial spectrum kernel with the normal spectrum kernel and the Fisher kernel in Figure 3. The coordinates of each point in the plot are the ROC scores for one SCOP family. The subpolynomial kernel uses the parameter $p = 0.2$. Although the subpolynomial method does not improve performance on every single family over the other two methods, there are only a small number of cases where there is a loss in performance.

Note that explicit feature selection cannot readily be used in this problem, unless it is possible to integrate the feature selection method into the construction of the spectrum kernel, as the features are never explicitly represented. Thus we do not know of another method that can provide the improvements described here. Note though that the improvements are not as large as reported in the other experiments (for example, the toy string kernel experiment of Section 4.1). We believe this is because this application does not suffer from the large diagonal problem as much as the other problems. Even without using the subpolynomial method, the spectrum kernel is already superior to the Fisher kernel method. Finally, note that while these results are rather good, they do not represent the record results on this dataset: in (Liao and Noble, 2002), a different kernel (Smith-Waterman pairwise scores)¹⁴ is shown to provide further improvements (mean RFP: 0.09, mean ROC: 0.89). It is also possible to choose other parameters of the spectrum kernel to improve its results. Future work will continue to investigate these kernels.

¹⁴ The Smith-Waterman score technique is closely related to the empirical kernel map, where the (non-positive definite) effective “kernel” is the Smith-Waterman algorithm plus p -value computation.

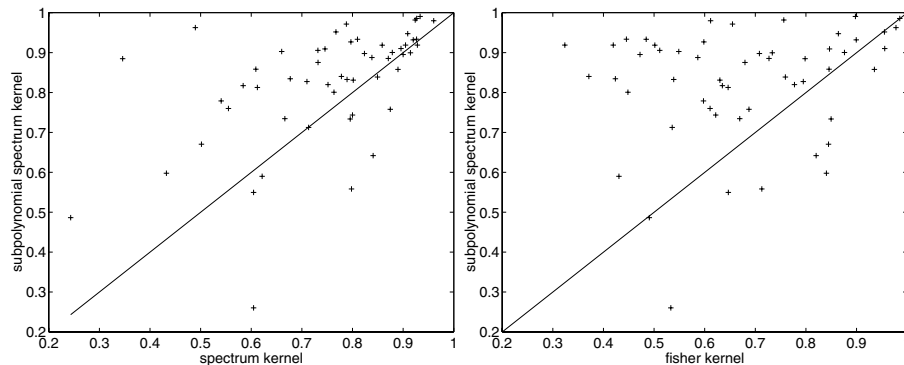


Fig. 3. Family-by-family comparison of the subpolynomial spectrum kernel with: the normal spectrum kernel (left), and the Fisher kernel (right). The coordinates of each point in the plot are the ROC scores for one SCOP family. The spectrum kernel uses $k = 3$ and $m = 0$, and the subpolynomial kernel uses $p=0.2$. Points above the diagonal indicate problems where the subpolynomial kernel performs better than the other methods.

5 Conclusion

It is a difficult problem to construct useful similarity measures for non-vectorial data types. Not only do the similarity measures have to be positive definite to be useable in an SVM (or, more generally, conditionally positive definite, see e.g. Schölkopf and Smola (2002)), but, as we have explained in the present paper, they should also lead to Gram matrices whose diagonal values are not overly large. It can be difficult to satisfy both needs simultaneously, a prominent example being the much celebrated (but so far not too much used) string kernel. However, the problem is not limited to sophisticated kernels. It is common to all situations where the data are represented as sparse vectors and then processed using an algorithm which is based on dot products.

We have provided a method to deal with this problem. The method's upside is that it turns kernels such as string kernels into kernels that work very well on real-world problems. Its main downside so far is that the precise role and the choice of the function we apply to reduce the dynamic range has yet to be understood.

Acknowledgements We would like to thank Olivier Chapelle and André Elisseeff for very helpful discussions. We moreover thank Chris Watkins for drawing our attention to the problem of large diagonals.

Bibliography

- A. A. Alizadeh et al. Distinct types of diffuse large b-cell lymphoma identified by gene expression profiling. *Nature*, 403:503–511, 2000. Data available from <http://llmpp.nih.gov/lymphoma>.
- U. Alon, N. Barkai, D. Notterman, K. Gish, S. Ybarra, D. Mack, and A. Levine. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon cancer tissues probed by oligonucleotide arrays. *Cell Biology*, 96:6745–6750, 1999.
- C. Berg, J. P. R. Christensen, and P. Ressel. *Harmonic Analysis on Semigroups*. Springer-Verlag, New York, 1984.
- B. E. Boser, I. M. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In D. Haussler, editor, *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, pages 144–152, Pittsburgh, PA, July 1992. ACM Press.
- M. P. S. Brown, W. N. Grundy, D. Lin, N. Cristianini, C. Sugnet, T. S. Furey, M. Ares, and D. Haussler. Knowledge-based analysis of microarray gene expression data using support vector machines. *Proceedings of the National Academy of Sciences*, 97(1):262–267, 2000.
- C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20: 273–297, 1995.
- I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 2001.
- D. Haussler. Convolutional kernels on discrete structures. Technical Report UCSC-CRL-99-10, Computer Science Department, University of California at Santa Cruz, 1999.
- T. S. Jaakkola, M. Diekhans, and D. Haussler. A discriminative framework for detecting remote protein homologies. *Journal of Computational Biology*, 7: 95–114, 2000.
- T. S. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. In M. S. Kearns, S. A. Solla, and D. A. Cohn, editors, *Advances in Neural Information Processing Systems 11*, Cambridge, MA, 1999. MIT Press.
- C. Leslie, E. Eskin, and W. S. Noble. The spectrum kernel: A string kernel for SVM protein classification. *Proceedings of the Pacific Symposium on Biocomputing*, 2002. To appear.
- L. Liao and W. S. Noble. Combining pairwise sequence similarity and support vector machines for remote protein homology detection. *Proceedings of the Sixth International Conference on Computational Molecular Biology*, 2002.
- H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins. Text classification using string kernels. *Journal of Machine Learning Research*, 2: 419–444, 2002.
- A. G. Murzin, S. E. Brenner, T. Hubbard, and C. Chothia. SCOP: A structural classification of proteins database for the investigation of sequences and structures. *Journal of Molecular Biology*, pages 247:536–540, 1995.

- E. Osuna and F. Girosi. Reducing the run-time complexity in support vector machines. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*, pages 271–284, Cambridge, MA, 1999. MIT Press.
- B. Schölkopf and A. J. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.
- K. Tsuda. Support vector classifier with asymmetric kernel function. In M. Verleysen, editor, *Proceedings ESANN*, pages 183–188, Brussels, 1999. D Facto.
- K. Tsuda, M. Kawanabe, G. Rätsch, S. Sonnenburg, and K.R. Müller. A new discriminative kernel from probabilistic models. In T.G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems*, volume 14. MIT Press, 2002. To appear.
- V. Vapnik. *Estimation of Dependences Based on Empirical Data [in Russian]*. Nauka, Moscow, 1979. (English translation: Springer Verlag, New York, 1982).
- V. Vapnik. *Statistical Learning Theory*. John Wiley and Sons, New York, 1998.
- C. Watkins. Dynamic alignment kernels. In A. J. Smola, P. L. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 39–50, Cambridge, MA, 2000. MIT Press.
- J. Weston, A. Elisseeff, and B. Schölkopf. Use of the ℓ_0 -norm with linear models and kernel methods. *Biowulf Technical report*, 2001. <http://www.conclu.de/~jason/>.
- J. Weston, F. Pérez-Cruz, O. Bousquet, O. Chapelle, A. Elisseeff, and B. Schölkopf. Feature selection and transduction for prediction of molecular bioactivity for drug design, 2002. <http://www.conclu.de/~jason/kdd/kdd.html>.
- J. Weston and B. Schölkopf. Dealing with large diagonals in kernel matrices. In *New Trends in Optimization and Computational algorithms (NTOC 2001)*, Kyoto, Japan, 2001.

Learning with Mixture Models: Concepts and Applications

Padhraic Smyth

Information and Computer Science, University of California
Irvine, CA 92697-3425, USA
smyth@ics.uci.edu

Abstract. Probabilistic mixture models have been used in statistics for well over a century as flexible data models. More recently these techniques have been adopted by the machine learning and data mining communities in a variety of application settings. We begin this talk with a review of the basic concepts of finite mixture models: what can they represent? how can we learn them from data? and so on. We will then discuss how the traditional mixture model (defined in a fixed dimensional vector space) can be usefully generalized to model non-vector data, such as sets of sequences and sets of curves. A number of real-world applications will be used to illustrate how these techniques can be applied to large-scale real-world data exploration and prediction problems, including clustering of visitors to a Web site based on their sequences of page requests, modeling of sparse high-dimensional "market basket" data for retail forecasting, and clustering of storm trajectories in atmospheric science.

Author Index

Abe, Kenji	1	Hall, Håkan	27
Agartz, Ingrid	27	He, Xiaofeng	112
Agarwal, Ramesh C.	237	Hirano, Shoji	188
Angiulli, Fabrizio	15	Hüllermeier, Eyke	200
Aref, Walid G.	51	Jaroszewicz, Szymon	212
Arikawa, Setsuo	1	Jeudy, Baptiste	225
Arimura, Hiroki	1	Jönsson, Erik	27
Arnborg, Stefan	27	Joshi, Mahesh V.	237
Asai, Tatsuya	1	Kargupta, Hillol	250
Asker, Lars	338	Kawasoe, Shinji	1
Atallah, Mikhail	51	Kemp, Charles	263
Bailey, James	39	Kindermann, Jörg	373
Berberidis, Christos	51	Klösgen, Willi	275
Blockeel, Hendrik	299	Knobbe, Arno J.	287
Boström, Henrik	338	Kosala, Raymond	299
Boulicaut, Jean-François	225	Koyutürk, Mehmet	311
Brain, Damien	62	Kumar, Vipin	237
Bruynooghe, Maurice	299	Lallich, Stéphane	475
Bussche, Jan Van den	299	Larson, Martha	373
Calders, Toon	74	Lavrač, Nada	163
Carvalho, Marcio	435	Leng, Paul	99
Choki, Yuta	86	Leopold, Edda	373
Coenen, Frans	99	Leslie, Christina	494
Cunningham, Pádraig	449	Li, Jinyan	325
Ding, Chris	112	Lidén, Per	338
Domeniconi, Carlotta	125	Ma, Sheng	125
Eickeler, Stefan	373	Maedche, Alexander	348
Elmagarmid, Ahmed K.	51	Mamitsuka, Hiroshi	361
Eskin, Eleazar	494	Manco, Giuseppe	175
Felty, Amy	138	Manoukian, Thomas	39
Forman, George	150	Marseille, Bart	287
Gamberger, Dragan	163	Matwin, Stan	138
Ghosh, Samiran	250	May, Michael	275
Giannotti, Fosca	175	Morishita, Shinichi	410
Goethals, Bart	74	Muhlenbach, Fabrice	475
Gozzi, Cristian	175	Oja, Erkki	488
Grama, Ananth	311	Paaß, Gerhard	373
Gusmão, Bruno	435	Palaniswami, Marimuthu	385

- Park, Laurence A.F. 385
 Parthasarathy, Srini 435
 Perng, Chang-shing 125
 Pizzuti, Clara 15
 Ramakrishnan, Naren 311
 Ramamohanarao, Kotagiri 39,
 263, 385
 Roth, Dan 489
 Scheffer, Tobias 397
 Schölkopf, Bernhard 494
 Sedvall, Göran 27
 Sese, Jun 410
 Siebes, Arno 287
 Sillén, Anna 27
 Simon, Horst 112
 Simovici, Dan A. 212
 Sivakumar, Krishnamoorthy ... 250
 Smyth, Padhraic 512
 Spiliopoulou, Myra 461
 Stafford Noble, William 494
 Suzuki, Einoshin 86
 Tsumoto, Shusaku 188, 423
 Veloso, Adriano 435
 Vilalta, Ricardo 125
 Vlahavas, Ioannis 51
 Wagner, Meira Jr., ... 435
 Wall, Robert 449
 Walsh, Paul 449
 Webb, Geoffrey I. 62
 Weston, Jason 494
 Winkler, Karsten 461
 Wong, Limsoon 325
 Wrobel, Stefan 397
 Zacharias, Valentin 348
 Zaki, Mohammed 435
 Zha, Hongyuan 112
 Zighed, Djamel A. 475