Vorlesung Neuroinformatik

R. Der
Universität Leipzig
Institut für Informatik

19. Juni 1998

Inhaltsverzeichnis

1 Grundlagen **7**

1.1 Neurobiologische Grundlagen 7

1.1.1 Allgemeines 7

1.1.2 Das Gehirn als System 7

1.1.3 Unterschiede Hirn – Computer 13

1.2 Grundlegende elektrophysiologische Mechanismen 14

1.2.1 Das Ruhepotential 16

1.2.2 Aktionspotential und Reizleitung 17

1.2.3 Synaptische Kopplungen 18

1.3 Das mathematische Modell des Neurons 20

1.3.1 Neuronale Informationskodierung 20

1.3.2 Funktionelles Schema des stationären Neurons 22

1.3.3 Das mathematische Modell des stationären Neurons . . . 22

2 Das Neuron als lernender Elementarprozessor **27**

2.1 Das Neuron als lernender Funktionsapproximator 27

2.1.1 On–line Lernen vs. batch Lernen 29

2.1.2 Beipiele 30

2.2 Das Neuron als lernender Klassifikator 36

2.3 Das Perzeptron 39

2.4 Grenzen des Neurons – das XOR Problem 41

3 Feed–Forward Netze **43**

3.1 Informationsverarbeitung mit Feed–Forward Netzen 43

3.2 Der Backpropagation–Algorithmus 45

3.2.1 Die Update–Regel 45

3.2.2 Anhang: Herleitung des Backprop-Algorithmus für Feed-Forward Netzwerke 46

3.3 Einige Anwendungen 47
3.3.1 NETTALK 47
3.3.2 Zeichenerkennung 48
3.3.3 Bottleneck–Netze zur Dimensionsreduktion und adaptiven Datenkompression 49
3.3.4 Emergenz getrennter Verarbeitungskanäle für Objekt– und Positionsinformation im visuellen System 52
3.4 Architekturen neuronaler Netze 57
3.4.1 *XOR* Netzwerke 58
3.4.2 Erlernen Boolscher Funktionen 58
3.4.3 Selbstorganisierende Architekturen 59
3.5 Nichtlineare Regression 60
3.5.1 Feed–Forward Netze als universelle Funktionsapproximatoren 60
3.5.2 Prognosegenauigkeit, Generalisierungsfähigkeit, Overfitting 61
3.5.3 Verfahren zur Verbesserung der Generalisierungsfähigkeit 65
3.6 Eigenschaften und Varianten des Backprop-Verfahrens 66
3.6.1 Die Austauschbarkeit von Steilheit und Lernrate 66
3.6.2 Variable Schrittlänge 67
3.6.3 Backpropagation mit Massenterm. 67
3.6.4 Quickprop 67

4 Neuronale Netze als Modelle von Zeitreihen 69
4.1 Allgemeines 69
4.1.1 Beispiele für Zeitreihen: 70
4.2 Prognose von Zeitreihen mit neuronalen Netzen 72
4.2.1 Ein *ad hoc* Ansatz 72
4.2.2 Feed-Forward-Netze als universelle Modelle von Zeitreihen 74
4.2.3 Qualitätskriterien 76
4.3 Rekurrente Netze 77
4.4 Zeitreihenprognose mit selbstorganisierenden Merkmalskarten (Kohonen-Netze) 77
4.4.1 Das Funktionsprinzip 77
4.4.2 Der Kohonenalgorithmus zur Zeitreihenvorhersage 78
4.5 Übungen 80

5 Partiell rekurrene Netze 83

6 Attraktornetzwerke **85**

6.1 Architektur und Dynamik des Netzwerkes 85

6.1.1 Definition des Netzwerkes 85

6.1.2 Formale Eigenschaften 87

6.1.3 Optimierung frustrierter Systeme als Netzwerkdynamik . 88

6.2 Das Hopfieldnetzwerk 90

6.2.1 Die Hebbsche Lernregel 90

6.2.2 Eigenschaften des Hopfieldmodells 91

6.2.3 Stochastische Netzwerke – das Hopfield Modell bei endlichen Temperaturen 98

6.3 Informationstheorie und Statistische Mechanik 103

6.3.1 Informationstheorie 103

6.3.2 Statistische Mechanik 104

7 Neuronale Netze und abstrakte Automaten **111**

7.1 Abstrakte Automaten 111

7.2 Neuronale Netze als abstrakte Automaten 112

7.3 Stochastische Automaten und Netze 114

8 Selbstorganisierende Karten **115**

8.1 Der Vektorquantisierer 116

8.1.1 Rezeptive Felder und Voronoiparkettierung des Inputraumes 116

8.1.2 Der Vektorquanisierer (VQ) 117

8.1.3 Wettbewerbslernen 119

8.1.4 Der überwacht lernende Vektorquantisierer (LVQ) 120

8.2 Selbstorganisation topografischer Abbildungen – der Algorithmus von Kohonen 122

8.3 Anwendungen der selbstorganisierenden Karten 128

8.3.1 Datenentrauschung 128

8.3.2 Selbstorganisierende Karten und kognitive Prozesse – Ein Modell des '*perceptual magnet*' Effektes 128

8.3.3 Lösung eines Optimierungsproblems – der reisende Handelsmann 129

8.3.4 WEBSOM – Ein automatisches Verfahren zur semantischen Clusterung von Internetdokumenten 129

8.4 Messung der topologischen Eigenschaften der Karten 129

Kapitel 1

Grundlagen

1.1 Neurobiologische Grundlagen

1.1.1 Allgemeines

Gegenstand der Neuroinformatik sind künstliche neuronale Netze (KNN) sowohl (i) als Modelle kognitiver Prozesse im Hirn als auch (ii) als neuartige Medien der Informationsverarbeitung. Wunschziel der Neuroinformatik ist die Entwicklung von Algorithmen oder letztendlich Maschinen (Computer) zur Verarbeitung von Informationen nach dem Vorbild Gehirn. Die Umsetzung dieser weit in die Zukunft reichenden Aufgabe erfordert als unabdingbare Voraussetzung ein hinreichend weit gediehenes Verständnis der Organisations- und Funktionsprinzipien der perzeptuellen und kognitiven Prozesse einschließlich ihrer neuronalen "Implementierung". Davon sind wir aber noch beliebig weit entfernt. Zwar ist einerseits die Anatomie des außerordentlich komplexen Systems Gehirn bis hinab auf die zelluläre Ebene erstaunlich gut bekannt. Andererseits wurde, besonders durch die stürmische Entwicklung bildgebender Verfahren in jüngster Zeit, die funktionelle Architektur in vielen wesentlichen Einzelheiten aufgeklärt. Die neuronale Verankerung der höheren kognitiven Prozesse muß aber bisher als weitgehend unbekannt angesehen werden. Weiterführende Literatur [30, 29].

1.1.2 Das Gehirn als System

Das Gehirn kann als das komplexeste uns bekannte System betrachtet werden. Seine Komplexität hat in der Evolution mit der Herausbildung und Entwicklung des Neocortex bei den Säugern beständig zugenommen. Dieser überdeckt beim Menschen die phylogenetisch früheren Teile des Hirns wie das limbische System

und das Stammhirn und zeichnet sich besonders durch seine Furchungen aus,

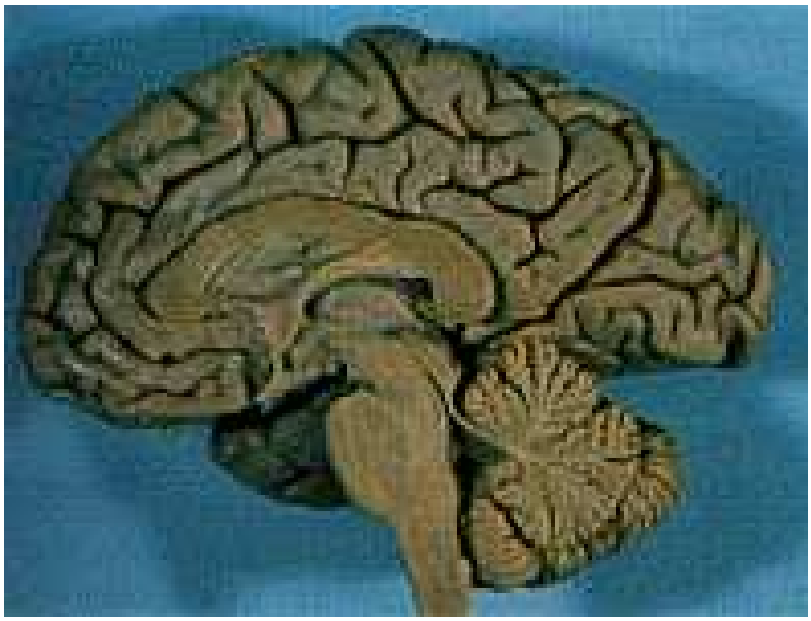


Abbildung 1.1: Senkrechter Schnitt durch das menschliche Hirn mit Hirnstamm, Kleinhirn, limbischem System und Hirnrinde (Neocortex) deutlich erkenntlich an seiner starken Furchung, die der Vergrößerung der Oberfläche dient.

die eine größere Oberfläche für den Neocortex schaffen.

Hinter der Vielzahl hochkomplexer Prozesse, die zielsicher auf dem neuronalen Substrat ablaufen, könnte man eine Vielfalt unterschiedlicher "Bauelemente" vermuten, die, geeignet zusammengeschaltet, die kognitiven Funktionen hervorbringen. Gesicherte neuroanatomische Erkenntnisse belegen aber genau das Gegenteil. In der Tat besteht das Hirn nur aus einigen wenigen Grundtypen von Nervenzellen oder Neuronen, die als die funktionellen Einheiten des Systems zu betrachten sind. Neuere Schätzungen gehen dabei für das menschliche Hirn von ca. 100 Mrd. Nervenzellen (Neuronen) aus. Zusätzlich gibt es im Hirn noch ca. 500 – 1000 Mrd Gliazellen. Die Nervenzellen sind untereinander über die Synapsen verschaltet, wobei jedes Neuron bis zu 1000, teilweise bis zu 10 000 Verbindungen zu anderen Zellen unterhält. [1] In den Abbildungen sind einige Zelltypen zu sehen. Eine ausführliche Zusammenstellung von Daten über das gesamte Nervensystem findet sich im Internet, s. [7].

Neuronen und Synapsen

In der Neuroinformatik werden die biologischen Nervenzellen durch idea-

[1] Purkinjezellen kö nnen im Kleinhirn sogar bis zu 200 000 Verbindungen ausbilden.

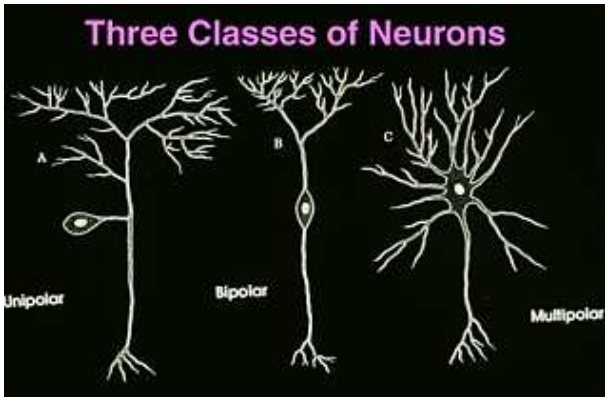


Abbildung 1.2: Einteilung der Zellen nach der Zahl der von ihnen ausgebildeten Pole. Unipolare haben nur einen Fortsatz, der vom Soma ausgeht. Sie überwiegen bei den Avertebraten (Wirbellosen). Bipolare Zellen sind vor allem in der Retina anzutreffen, wo sie die Schaltstelle zwischen den Photorezeptoren und den nachgeschalteten amakrinen und Ganglien–Zellen bilden. Multipolar heißen Zellen mit mehreren Fortsätzen. Überwiegender Zelltyp im Zentralnervensystem der Vertebraten.

lisierte mathematische Modelle repräsentiert. Vorbild für diese Modelle sind Zellen des in Abbildung 1.3 dargestellten Typs (Pyramidenzelle).

Diese Zellen wirken erregend aufeinander ein. Dieser Selbststimulierung der neuronalen Aktivität muß eine ausreichend starke Hemmung entgegenstehen. Diese wird vornehmlich von den Interneuronen aber auch z. B. von den sog. Purkinje–Zellen ausgeübt, s. Abb. 1.4.

Für die Visualisierung der zellularen Struktur des Hirngewebes stehen eine Reihe von Färbungsverfahren zur Verfügung, s. Abb. 1.5 und 1.6.

Von einem funktionell-neuroanatomischen Standpunkt aus gesehen besteht das Neuron aus dem Zellkörper (Soma) mit seinen dendritischen Fortsätzen und dem Axon. Das Neuron empfängt über synaptische Kontakte entweder direkt über sein Soma oder über seine Dendriten elektrische Signale x_i von anderen Neuronen $i = 1, 2, ...$ oder den Muskeln bzw. Sinnesorganen. Diese werden aufsummiert und sobald ihre summarische Wirkung einen gewissen Schwellwert überschreitet, sendet das Neuron selbst ein Aktionspotential y (s.u.) über sein Axon ab. Dieses kann die Signale über eine vergleichsweise lange Strecke weiter-

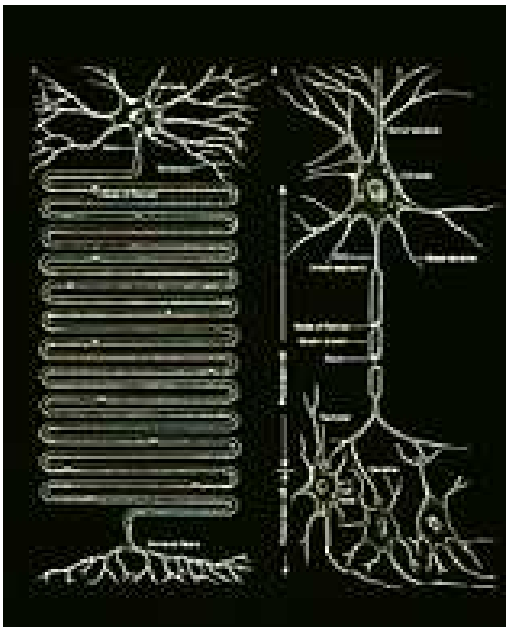


Abbildung 1.3: Schematische Darstellung einer typischen multipolaren Nervenzelle des Zentralnervensystems (links). Zellkörper und dendritische Fortsätze sind deutlich zu erkennen. Zur Veranschaulichung der natürlichen Längenverhältnisse ist das Axon in seiner vollen Länge dargestellt. Die Endverzweigungen des Axons dienen der Kontaktierung anderer Neuronen oder von Muskelzellen (im Fall von Motorneuronen). Rechts im Bild ist die Verschaltung mit weiteren Nervenzellen schematisch dargestellt.

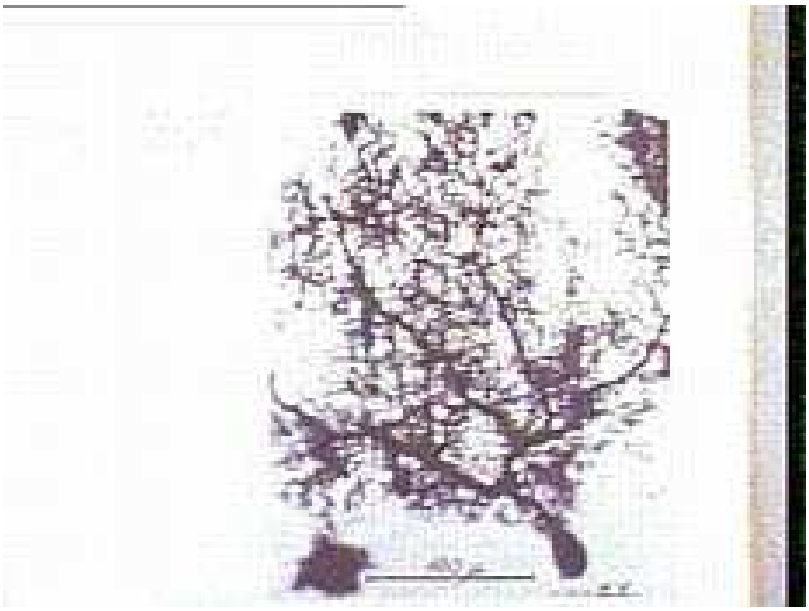


Abbildung 1.4: Eine Purkinje–Zelle. Dieser Zelltyp ist ähnlich wie die Interneurone für hemmende Schaltkreise wichtig. Purkinje–Zellen werden unabhängig von ihrer eigenen Erregung durch Korbzellen gehemmt (Vorwärtshemmung). Andererseits bilden die Purkinje–Zell–Axone hemmende Synapsen auf den Kleinhirnkernen aus. Die ausgelöste Hemmung ist direkt der Aktivität der Purkinje–Zellen proportional (tonische Hemmung).

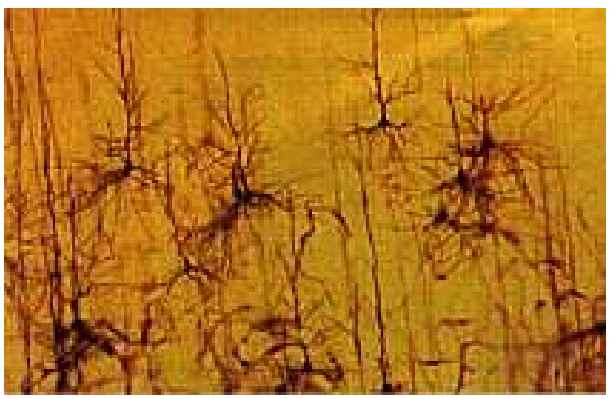


Abbildung 1.5: Durch Golgi-Färbung sichtbar gemachte Nervenzellen.

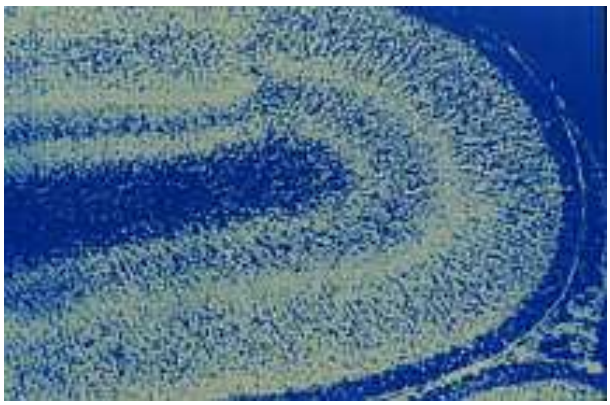
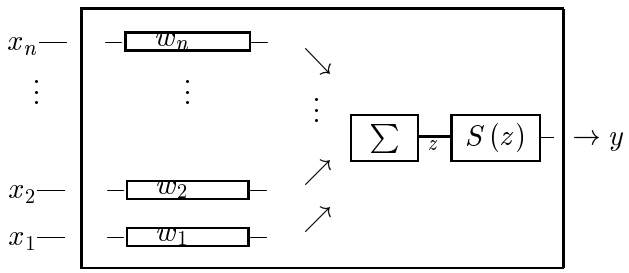


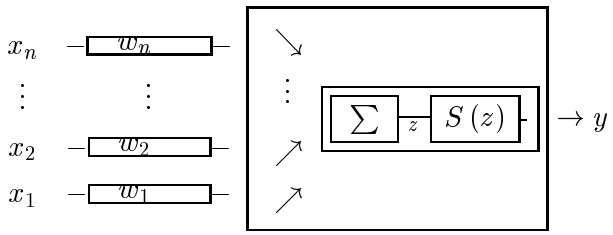
Abbildung 1.6: Die Nissl-Färbung macht die Zytoarchitektur des visuellen Kortex sichtbar.

leiten und kontaktiert dann durch seine Endverzweigungen andere Neuronen:



(1.1)

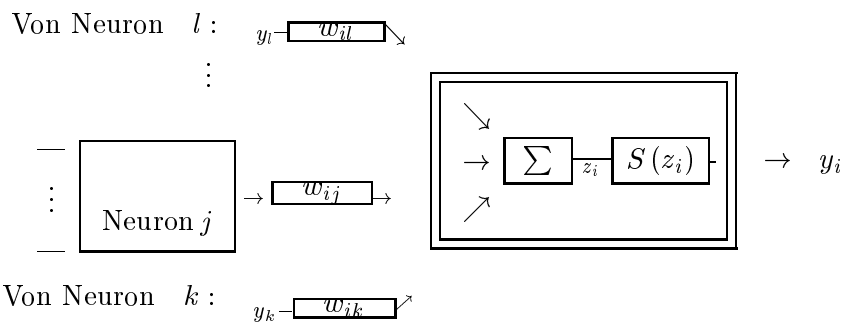
wobei die w_i die Effektivitäten bzw. Leitfähigkeiten der Synapsen (Kontakte) symbolisieren. Wenn wir stärker das Soma als eine Einheit betrachten wollen, dann stellen wir das Neuron einschließlich seiner synaptischen Kontakte so dar:



(1.2)

Funktionell (und stark vereinfachend) kann das Neuron also wie ein elektrisches Bauelelement mit einem definierten wenn auch komplexen Input–Output–Verhalten betrachtet werden. Das System Gehirn ist in diesem Sinne ein komplexes Netzwerk bestehend aus Neuronen, die über die synaptischen Kontakte

miteinander wechselwirken können, ausschnittsweise



(1.3)

Bei der Einordnung dieses Systems ist aber zu beachten, daß wir es hier mit einem offenen, aktiven System zu tun haben, das nicht nur von der Umwelt stark beeinflußt wird sondern das auch gezielt auf seine Umwelt einwirken und diese modifizieren kann, was zu hochkomplexen Rückkopplungseffekten führt. Eine weitere Besonderheit gegenüber den etwa in der Physik betrachteten komplexen Systemen ist die Tatsache, daß das System Gehirn (bzw. das Teilsystem KNN) im Prozeß des Lernens (d. h. der Adaption an die Umwelt) die Wechselwirkungen (Effektivitäten der synaptischen Kontakte) zwischen seinen Elementen (Neuronen) selbst verändern kann, so daß sich die Systemeigenschaften dauernd ändern. Damit ist das Hirn der Prototyp eines adaptiven Systems in seiner höchstentwickelten Form.

1.1.3 Unterschiede Hirn – Computer

Der Unterschied zwischen einem herkömmlichen Computer und dem Hirn folgt aus dem Vergleich von Verarbeitungsprinzipien und Leistungen. Interessant ist etwa ein Vergleich zwischen dem Hirn eines Spatzen und einem heutigen Supercomputer. Wir betrachten ersteres als ein elektronisches System mit ca. 1 Milliarde Bauelementen (Neuronen), die bestenfalls alle 10 ms einen verarbeitungsrelevanten Schaltvorgang vornehmen können (Übergang von einer hohen zu einer niedrigen Feuerrate oder umgekehrt). Dann führt dieses System maximal 10^{11} identifizierbare Schaltvorgänge pro Sekunde aus, liegt also noch unterhalb des Bereiches, in dem moderne Supercomputer operieren.

Obgleich in der reinen Rechenleistung durchaus vergleichbar, sind die kognitiven Leistungen eines Spatzenhirns denen eines jeden Supercomputers weit überlegen. Das betrifft insbesondere alle Fähigkeiten zur Mustererkennung und –Verarbeitung, Lernfähigkeit, die zielgerichtete Steuerung der Bewegungen und

allgemein die Überlebensfähigkeit in einer komplexen Umgebung, die immer neue Anpassungsleistungen erfordert.

Wichtige Charakteristika, die das Hirn vom Computer unterscheiden und die zumindest teilweise den trotz der außerordentlich geringen Geschwindigkeit der neuronalen Schaltkreise erheblichen Performanzvorteil des "Neurocomputers" Gehirn erklären, sind

- Hochgradige Parallelverarbeitung.
- Integration von "Spezialhardware" in Form funktionsspezifischer Neuromodule.
- Keine Trennung von Hard– und Software. Die Funktion des "Neurocomputers" folgt aus seiner Architektur und wird großenteils in der Auseinandersetzung mit der Umwelt erworben.
- Weitere unbekannte Architektur– und Funktionsprinzipien.

Der Unterschied zwischen Hirn und Computer kommt aber insbesondere auch in der Robustheit des Hirns gegenüber Störungen aller Art zum Ausdruck. Eines der markantesten Beispiele sind Hirnverletzungen. Besonders bekannt geworden ist der Fall des Phineas Gage, dessen Verletzungen durch einen Arbeitsunfall in der Computerrekonstruktion in Abb. 1.7 deutlich werden.

Trotz seiner furchtbaren Verletzungen genas Gage nach etwa zwei Monaten und erlangte seine grundlegenden geistigen, sprachlichen und beruflichen Fähigkeiten zurück. Allerdings war nach den Worten seines behandelnden Arztes Dr. Harlow "das Gleichgewicht zwischen seinen geistigen Fähigkeiten und seinen animalischen Neigungen" entscheidend gestört. Diese fatale Charakteränderung bewirkte, daß er auf seinen wechselnden Arbeitsstellen nicht lange geduldet wurde, so daß er im Verlaufe seines immer noch fast zwanzig Jahre währenden Lebens in große persönliche Schwierigkeiten geriet.

1.2 Grundlegende elektrophysiologische Mechanismen

Physiologische Flüssigkeiten (wässrige Salzlösungen) wie $NaCl$ und KCl liegen intra– und extrazellulär in dissoziierter Form, d. h. in Form der anionischen Cl^- und der kationischen Na^+ bzw. K^+ Ionen vor. Die Zellmembran, die die

Abbildung 1.7: Der berühmt gewordene Fall des Phineas Gage zeigt besonders eindringlich die Unvergleichbarkeit von Hirn und Computer. Im Bild eine Rekonstruktion der Verletzungen, die er durch einen Arbeitsunfall erlitt. Obwohl die eingedrungene Eisenstange große Teile seines Hirns zerstörte, erlangte er nach seiner physischen Genesung auch seine grundlegenden geistigen, sprachlichen und beruflichen Fähigkeiten zurück.

äußere Umhüllung des Neurons einschließlich seiner Dendriten und des Axons darstellt, trennt elektrisch diese beiden gut leitenden Medien voneinander. Das System

$$\begin{array}{c} \underline{\quad intrazelluläres\ Medium \quad} \\ \times\times\times\times Membran \times\times\times\times\times\times \\ \overline{\quad extrazelluläres\ Medium \quad} \end{array} \tag{1.4}$$

bildet folglich einen Kondensator mit der Membran als Dielektrikum. Diese ist allerdings teilweise für die Ladungen (Ionen) durchlässig – feinste, kompliziert gebaute Poren in der Zellmembran ermöglichen sowohl passive als auch aktive Transportprozesse durch die ca. 6 nm (≈ 60 Å) dicke Zellmembran hindurch. Für die Na^+ Ionen wirken spezielle Poren wie eine Art Ionenpumpe, die diese Ionen in eine Vorzugsrichtung (von innen nach außen) auch gegen elektrische Kräfte durch die Membran hindurchbefördert. Die erforderliche Energie wird durch Stoffwechselvorgänge bereitgestellt. Die passiven Kanäle sind nur für die K^+ und bei Nervenzellen in geringerem Maße auch für die Cl^-– Ionen durchlässig, deren Hydrathüllen kleiner als die der Na^+ Ionen sind.

1.2.1 Das Ruhepotential

Das Ruhepotential wird durch die ungleiche Konzentration der An– und Kationen auf beiden Seiten der Membran erzeugt. Träger des Potentials ist vornehmlich das $K^+ - Cl^-$ System gemeinsam mit auschließlich in der Zelle vorhandenen kationischen Eiweißmolekülen A^- ($\ominus$), die im Gegensatz zu den K^+ Ionen ($\oplus$) **nicht** hinausdiffundieren können. Unter Vernachlässigung der Cl^- Ionen ergibt sich schematisch folgendes Bild

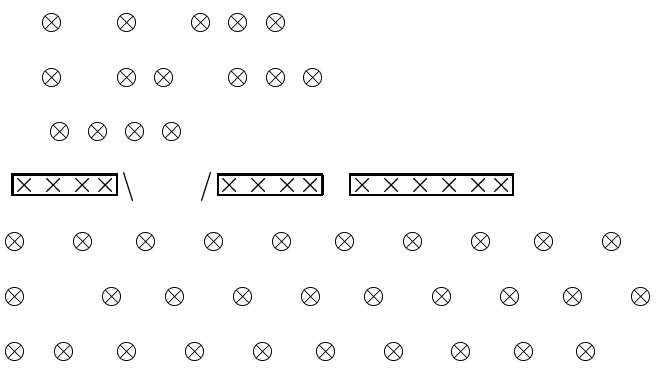
$$\begin{array}{ll} \ominus\ \oplus\ \ominus\ \oplus\ \ominus\ \oplus\ \ominus\ \oplus\ \ominus\ \oplus & \\ \ominus\ \oplus\ \ominus\ \oplus\ \ominus\ \ominus\ \oplus\ \ominus\ \oplus\ \ominus & \text{(intrazellulär)} \\ \oplus\ \ominus\ \oplus\ \ominus\ \oplus\ \ominus\ \ominus\ \oplus\ \ominus\ \oplus & \\ \boxed{\times\times\times\times\times\times} \quad \boxed{\times\times\times\times\times\times\times\times\times} & \\ \text{Pore} \ \nearrow & \\ \oplus & \text{(extrazellulär)} \end{array} \tag{1.5}$$

Die Diffusion entfernt positive Ionen aus dem Inneren, was zu einem Überwiegen der negativen Ladungen und damit zu einem Absenken (Negativierung) des intrazellulären Potentials führt. Allerdings würde dieses Potential durch die ebenfalls diffusionsfähigen Cl^- Ionen wieder aufgehoben. Die Stabilisierung des

Ruhepotentials auf seinen beobachteten Wert bewirken letztendlich die aktiven Transportprozesse durch die Ionenpumpen, die Na^+ Ionen aus dem Inneren der Zelle entfernen. Im Ergebnis der so erzeugten ungleichen Ladungsverteilung[2] erscheint die Zelle **polarisiert**, das Innere der Zelle liegt typischerweise auf einem Potential von ca. -70 mV. (Potentialangaben sind immer relativ zum extrazellulären Raum, dessen Potential also als Null angenommen wird).

1.2.2 Aktionspotential und Reizleitung

Die Wirkung der Ionenpumpen bzw. der Permeabilität der Membran für bestimmte Ionen hängt entscheidend von der anliegenden Potentialdifferenz ab. Wird etwa durch einen externen, überschwelligen Reiz (von einem anderen Neuron) die Membran lokal depolarisiert (Potentialdifferenz unter einen Schwellwert von ca. -50 mV vermindert), dann wird zunächst die Membran kurzzeitig für die Na^+ Ionen durchlässig, diese strömen also massiv solange in das Innere der Zelle ein, bis die negative Ladung lokal kompensiert ist. Kurz danach bewirken einströmende K^+ Ionen einen weiteren **positiven** Potentialaufbau. Diese Abweichung vom Ruhepotential wird als Aktionspotential bezeichnet. In 1.6 ist schematisch das Eindringen der Kationen gezeigt, wobei die dadurch verursachte Depolarisation auch der Umgebung gut zu erkennen sein sollte (dargestellt sind nur die Na^+ Ionen $\otimes$)



(1.6)

Dieser Transportvorgang ist außerordentlich schnell, seine Zeitkonstane liegt bei 0.2 mS. Die Depolarisation strahlt in die Umgebung aus, denn der Zusammenbruch des Potentialgefälles erhöht auch die Na^+ Leitfähigkeit der umliegenden Regionen, was auch dort zu einem Einströmen von Na^+ Ionen führt. Somit

[2] Das Konzentrationsverhältnis (intra:extrazellulär) der K^+ Ionen ist typischerweise $50:1$, das der Na^+ Ionen etwa $1:10$. Die Konzentrationsverteilung der Cl^- Ionen ist ziemlich genau umgekehrt gleich dem der K^+ Ionen.

pflanzt sich die ursprüngliche lokale Potentialstörung über die gesamte Neuronenmembran fort.

Die durch die Depolarisation ausgelöste Durchlässigkeit der Membran ist nicht irrreversibel. Komplizierte Prozesse in der Membran sorgen nach wenigen Millisekunden für deren Inaktivierung und führen damit zur baldigen Wiederherstellung der alten Verhältnisse. Einmal ausgelöste Aktionspotentiale leben lokal also nur eine bestimmte Zeit, so daß die Zelle nach der sog. Refraktärphase (ca. $1 - 3$ mS) für die Aufnahme eines neuen Reizes bereit ist. Die Aktionspotentiale breiten sich über das Axon mit einer Geschwindigkeit von ca. 100 m/S aus. Die Zeitspanne für das Aktionspotential an einer festen Stelle ist i. a. nicht größer als 1/2 mS, d. h. das Aktionspotential hat einen impulsförmigen Zeitverlauf und wird deshalb i. a. als **Spike** bezeichnet. Bei der Ausbreitung behalten diese Impulse ihre Form bei. Folgen von solchen Spikes stellen den Output des Neurons dar.

Spikes

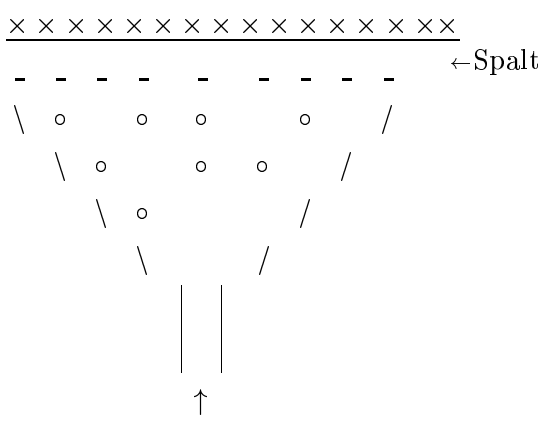
Neben dieser Form der Ausbreitung gibt es noch die passive Ausbreitung elektrischer Reize, die dabei aber abgeschwächt und in ihrem Zeitverlauf verzerrt werden. Diese Art der Reizleitung bestimmt im wesentlichen die Ausbreitung der Inputsignale. Im Gegensatz zu den Spikes haben diese lokalen Potentiale beliebige Stärken, so daß sie auf diese Weise die Stärke der Inputs (z. B. sensorische Reizung) kodieren können. Von dieser Natur sind auch die postsynaptischen Potentiale (s. u.).

1.2.3 Synaptische Kopplungen

Die Übertragung der elektrischen Signale zwischen den Neuronen erfolgt über die Synapsen. Zum synaptischen Bereich rechnen wir den präsynaptischen Endkolben, den ca. $0,1\,\mu m$ schmalen synaptischen Spalt und die sub– oder postsynaptische Membran als den von der Synapse überdeckten Bereich der Zellmembran des kontaktierten Neurons. Die Synapsen haben i. a. eine Ventilfunktion in dem Sinne, daß die Übertragung der Signale immer nur in der Richtung auf

Ventilfunktion der Synapsen

das kontaktierte Neuron hin erfolgen kann (Pfeilrichtung in (1.7)).



(1.7)

Die sog. synaptischen Bläschen oder Vesikel ∘ (Durchmesser ca. 50 nm) im präsynaptischen Endkolben enthalten jeweils eine ziemlich genau definierte Menge[3] (Quantum) einer speziellen chemischen Substanz, den Neurotransmitter[4]. Ein afferenter (ankommender) elektrischer Reiz bewirkt *via* Depolarisation der präsynaptischen Membran die Freisetzung und in Folge die Ausschüttung in den synaptischen Spalt dieses Überträgerstoffes. Einmal im Spalt bewirken die Transmitterstoffe ihrerseits eine Depolarisation der subsynaptischen Membran und lösen dadurch ein Aktionspotential in der kontaktierten (postsynaptischen) Zelle aus. Der gesamte Prozeß ist sehr schnell, die synaptische Latenz beträgt ca. 0.2 mS. Die Menge der freigesetzten Quanten (genauer die Wahrscheinlichkeit für ihre Freisetzung) hängt durchaus von der Stärke des präsynaptischen Aktionspotentials ab, so daß diese Information weitergegeben werden kann. Ähnlich der automatischen Inaktivierung der Zellmembran werden auch die Transmitter nach erfolgter Signalübertragung innerhalb kürzester Zeit wieder von dem synaptischen Endkolben resorbiert, wodurch die Synapse in ihren Ausgangszustand zurückkehrt.

Neurotransmitter

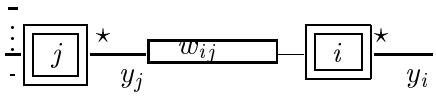
Entscheidend für die Funktion neuronaler Netze ist die Tatsache, daß sich die synaptische Effektivitäten, d. h. die Übertragungseigenschaften der Synapsen, modifizieren lassen. Sie übertragen bei häufiger Benutzung besser als bei seltener. Das ist auch der wesentliche Inhalt der Hebbschen Lernregel. Diese führt das Lernen auf eine erregungsproportionale Verstärkung der synaptischen Verbindungen zwischen gleichzeitig aktiven Neuronen zurück. Betrachten wir

Synaptische Effektivitäten

[3] Man schätzt, daß ein Vesikel ca. 1000 bis 10 000 Transmittermoleküle (z. B. Acetylcholin) enthält und daß pro Synapse 200 bis 2000 Quanten freigesetzt werden.

[4] Wir besprechen hier nur die chemischen Synapsen. Die selteneren elektrischen Synapsen funktionieren anders.

zwei Neuronen wie in (1.3)



(1.8)

Hebbsches Lernen

Die Veränderung Δw_{ij} der synaptischen Effektivität [5] w_{ij} ist nach Hebb

$$\Delta w_{ij} = \epsilon y_i y_j \tag{1.9}$$

Sind nun, wie in 1.8 durch den Stern $\star$ symbolisiert, die Outputs beider Neuronen i, j auf einem hohen Niveau, so wird die synaptische Verbindung w_{ij} gemäß 1.9 deutlich verstärkt. Der typische Wertebereich für die Lernrate ist $0.01 < \epsilon < 0.1$.

Auf der Plastizität der Synapsen beruht allgemein die Lern– und Gedächtnisfunktion neuronaler Netze. Trotz ihrer Einfachheit und ihrer nur lokalen Einflußnahme (Δw_{ij} ist nur von den unmittelbar beteiligten Neuronen abhängig) ist die Hebbsche Lerndynamik ein grundlegendes Wirkprinzip für das Erlernen globaler Fähigkeiten durch das neuronale Netz.

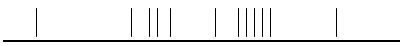
1.3 Das mathematische Modell des Neurons

1.3.1 Neuronale Informationskodierung

Die impulsförmigen Aktionspotentiale anderer Neuronen erreichen die Dendriten oder das Soma (Zellkörper) eines Empfängerneurons über die synaptischen Kontakte. Diese elektrischen Signale werden passiv über die Dendriten und aktiv nach dem in Abschnitt 1.2 beschriebenen Mechanismus über die gesamte Zellmembran fortgeleitet. Das Neuron integriert diese Potentiale mit einer sehr kleinen Zeitkonstanten auf. Überschreitet deren summarischer Effekt am Axonhügel eine bestimmte Schwelle, so sendet das Neuron über sein Axon ebenfalls ein Aktionspotential (Spike) aus, das seinerseits andere Neuronen aktivieren kann. Systemtheoretisch gesehen ist das Neuron also durch ein bestimmtes Input–Output–Verhalten gekennzeichnet, das eine komplexe Zeitstruktur aufweist. Als Element des informationsverarbeitenden Systems Gehirn betrachtet ist zu fragen, in welcher Weise die Information in einer solchen funktionellen Architektur kodiert ist und wie das Neuron diese Information lesen und verarbeiten kann.

[5] Die betrachtete Synapse kontaktiert das Neuron i (Zielneuron). Wir schreiben w_{ij} für die Kopplungsstärke (Effektivität) der Synapse, d.h. der erste Index bezieht sich immer auf das Zielneuron.

Es gibt hier eine Vielzahl von Ansätzen, von denen wir nur die zwei grundlegenden nennen wollen. Der erste geht davon aus, daß die Zeitstruktur der Signale wesentlich ist. Im Extremfall kann man annehmen, daß die Information in einer Spikefolge durch den zeitlichen Abstand zwischen aufeinanderfolgenden Spikes (die alle etwa die gleiche Amplitude haben) kodiert ist, schematisch



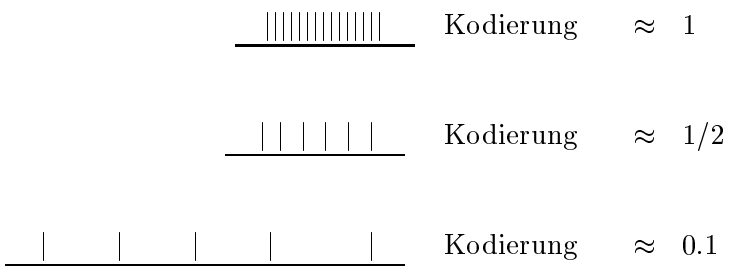
(1.10)

Dem entspricht in der Nachrichtentechnik die Puls**phasen**modulation. Für deren Realisierung als neuronale "Sprache" gibt es bisher aber relativ wenig empirische Anhaltspunkte, da sie eine recht genaue Messung der zeitlichen Abstände voraussetzt und sie demzufolge nicht die für biologische Systeme erforderliche Robustheit aufweist. Andererseits ermöglicht diese Zeitstruktur der neuronalen Signale Synchronisationsphänomene zwischen verschiedenen Neuronen, was als ein Lösungsansatz zum Verständnis des Binding–Problems allgemein akzeptiert wird.

Pulsphasen - Modulation

Grundlegend für die in dieser Vorlesung zu besprechenden künstlichen neuronalen Netze ist der zweite Ansatz, daß nämlich die Information, ähnlich der Puls**frequenz**modulation in der Nachrichtentechnik, einzig durch die Feuerrate (das ist die Frequenz mit der die Spikes abgesendet werden) kodiert ist. Damit die Feuerrate als robuster Informationsträger fungieren kann, müssen diese zeitlichen Abstände dann über einen gewissen Minimalzeitraum hinreichend konstant bleiben (der Minimalzeitraum ist frequenzproportional). Die Feuerraten lassen sich durch kontinuierliche Zahlen abbilden,

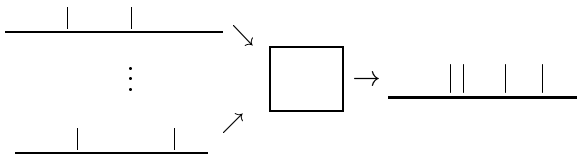
Ratenkodierung



die geeignet skaliert werden, so daß sie beispielweise zwischen 0 (Neuron feuert nicht) und 1 (Neuron feuert mit maximaler Frequenz) liegen. Diese Neuronen werden auch als stationäre Neuronen bezeichnet.

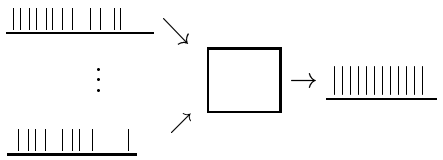
Im Extremfalle des McCulloch–Pitts Neurons (s.u.) werden nur zwei Situa-

tionen unterschieden, der Fall niedriger Aktivität



(1.11)

wird durch den Output $y = 0$, der Fall hoher Aktivität

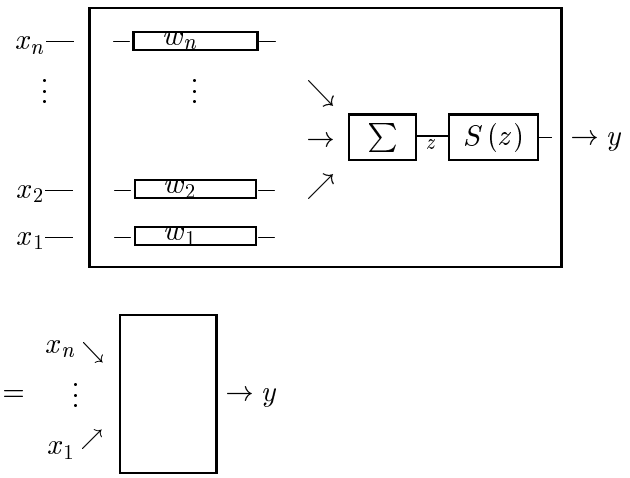


(1.12)

durch $y = 1$ kodiert. In 1.11 und 1.12 bezeichnet $|||\ |||\ ||$ wieder eine zeitliche Abfolge von Spikes.

1.3.2 Funktionelles Schema des stationären Neurons

Für die Untersuchung künstlicher neuronaler Netze fassen wir das in 1.1 eingeführte Neuron als ein kompaktes Bauelement auf,



(1.13)

das mathematisch durch seine Transferfunktion modelliert wird. Wie in Abschnitt 1.3.1 besprochen, sind dabei die Inputs und Outputs als statische Größen aufzufassen (Ratenkodierung), von deren elektrophysiologischer Rolle oder Interpretation abstrahiert wird.

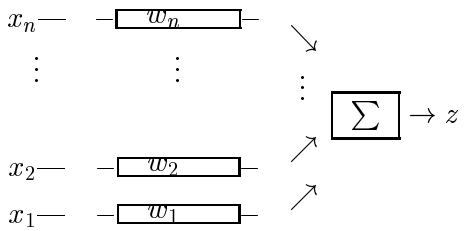
1.3.3 Das mathematische Modell des stationären Neurons

Das stationäre Neuron als funktionelle Einheit künstlicher neuronaler Netze wird modellhaft durch seine Transferfunktion

$$y = f(\vec{x}) \tag{1.14}$$

beschrieben, diese bildet die Inputs $\vec{x} = (x_1, ..., x_N)$ auf den Output y des Neurons ab. Wir setzen immer eine Ratenkodierung der neuronalen Information voraus, d.h. sowohl die Inputs x_i als auch der Output y kodieren jeweils eine Feuerrate, s. Abschnitt 1.3.1. Explizit berechnet das Neuron in idealisierter Form zunächst sein postsynaptisches Potential (PSP) z

Postsynaptisches Potential

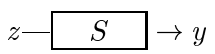


(1.15)

oder

$$z = \sum_{i=1}^{n} w_i x_i = \vec{w} \cdot \vec{x} \tag{1.16}$$

als mit den synaptischen Effektivitäten w_i gewichtete Summe seiner Inputs, vergleicht diese mit einer Schwelle U und bildet das Ergebnis dann nichtlinear auf das Intervall $[0, 1]$ ab, letzteres besorgt das Element



in 1.13, das die biologische Funktion des Axonhügels mathematisch modelliert, explizit

$$y = S\,(z - U) \tag{1.17}$$

Die Transferfunktion

Transferfunktion

$$y = f\,(\vec{x}) = f\,(\vec{x} \mid \vec{w}) \tag{1.18}$$

lautet folglich in expliziter Form

$$\boxed{y = S\,(\vec{w} \cdot \vec{x} - U)} \tag{1.19}$$

Für die feed–forward Netze ist S eine sigmoide Funktion, z. B.

$$S\,(z) = \frac{1}{1 + e^{-\beta z}} \tag{1.20}$$

wobei β die Steilheit des Anstieges parametrisiert, s. Abbildung 1.8. Die Ausgabefunktion S reflektiert den nichtlinearen Zusammenhang zwischen Input und Output, insbesondere in Form des Sättigungsverhaltens des biologischen

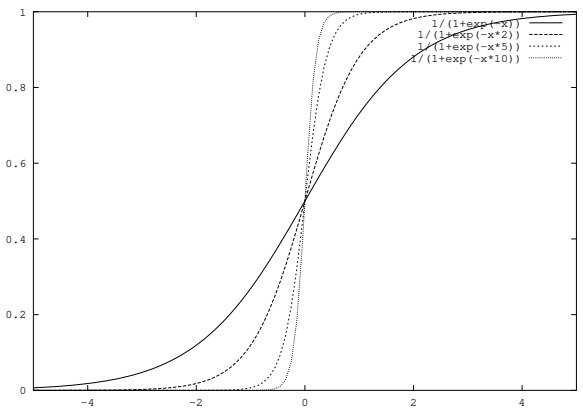


Abbildung 1.8: Die Ausgabefunktion für Parameter $\beta = 1$, $\beta = 2$, $\beta = 5$ und $\beta = 10$.

Neurons: Für sehr große PSP ist die Feuerrate praktisch nicht mehr vom PSP abhängig.

Der Trick mit der Schwelle

Die Schwelle U kann durch die Einführung eines fiktiven Inputs $x_0 = 1$ und $w_0 = -U$ und Definition des Skalarproduktes $\vec{x} \cdot \vec{w} = \sum_{i=0}^{n} w_i x_i$ berücksichtigt werden. Dann ist die allgemeine Transferfunktion $y = f(\vec{x})$

$$\boxed{f(\vec{x}) = f(\vec{x} \mid \vec{w}) = S(\vec{w} \cdot \vec{x})} \tag{1.21}$$

Wenn die Abhängigkeit von den Parametern explizit zum Ausdruck gebracht werden soll, schreiben wir $f(\vec{x})$ als $f(\vec{x} \mid \vec{w})$.

Ein Spezialfall der sigmoiden Ausgabefunktion ist die Sprungfunktion

$$\theta(z) = \begin{cases} 1 & \text{für} \quad z \geq 0 \\ \\ 0 & \text{für} \quad z < 0 \end{cases} \tag{1.22}$$

Binäres Neuron

wobei $\theta(z) = \lim_{\beta \to \infty} S(z)$. Dieses binäre Neuron wurde zuerst von McCulloch und Pitts eingeführt[28]. Für eine Reihe von Netzwerkmodellen hat es sich als günstig erwiesen, den Zustand minimaler Erregung des Neurons statt mit 0 mit -1 zu kodieren, d. h.

$$S(z) = \tanh(\beta z) = \frac{e^{\beta z} - e^{-\beta z}}{e^{\beta z} + e^{-\beta z}} \tag{1.23}$$

Bipolares Neuron

Dem McCulloch–Pitts Neuron entspricht dann das bipolare Neuron mit

$$S(z) = sign(z) = \begin{cases} 1 & \text{für} \quad z \geq 0 \\ \\ -1 & \text{für} \quad z < 0 \end{cases} \tag{1.24}$$

Bemerkungen: Für die Anwendungen mit feed–forward Netzwerken (s.u.) wird meist die Kodierung gemäß 1.20 gewählt. Die Ausgabefunktion 2.1 ist oft nützlich, wenn das Netz auch negative Werte ausgeben soll. Für die Steuerung eines mobilen Roboters kann das beispielweise interessant sein, wenn das Ausgabeneuron direkt die Radansteuerung vorgibt (negative Werte bei Rückwärtsfahrt).

Kapitel 2

Das Neuron als lernender Elementarprozessor

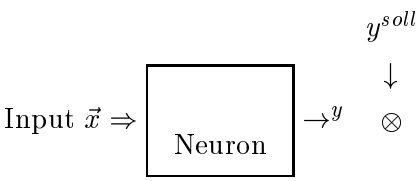
Wir untersuchen nun das oben eingeführte formale Neuron in seiner Eigenschaft als universeller Elementarprozessor neuronaler Netze.

2.1 Das Neuron als lernender Funktionsapproximator

Mathematisch ist das Neuron eine Funktion $y = f(\vec{x})$, die von einer Reihe von Parametern, zusammengefaßt im Parametervektor $\vec{w}$, abhängt, $f(\vec{x}) = f(\vec{x} \mid \vec{w})$. Das Neuron kann deshalb als Funktionsapproximator dienen, indem $\vec{w}$ so gewählt wird, daß die Funktion $f(\vec{x})$ möglichst wenig von der Zielfunktion $F(\vec{x})$ abweicht. Gesucht ist ein Lernalgorithmus, der den optimalen Parametervektor $\vec{w}$ findet. Das Lernen geschieht durch Präsentation von Instanzen der Zielfunktion $F(\vec{x})$, jede Instanz besteht aus dem Wertepaar

$$Z_{\vec{x}} = (\vec{x}, F(\vec{x})) = \left(\vec{x}, y^{soll}\right) \tag{2.1}$$

d. h. aus dem Solloutput y^{soll} des Neurons zu gegebenem Input $\vec{x}$, wobei sich im allgemeinen eine Abweichung zwischen dem Istwert $y = f(\vec{x})$ und dem Solloutput ergibt.



(2.2)

Eine geeignete Fehler– oder Kostenfunktion wie z. B.

$$E = E(\vec{x}|\vec{w}) = \frac{1}{2}\left(y^{soll} - y\right)^2 = \frac{1}{2}\left(f\left(\vec{x}\right) - F\left(\vec{x}\right)\right)^2 \tag{2.3}$$

mißt die Güte der Approximation bei gegebenem Parametersatz $\vec{w}$. Ziel ist die Minimierung dieses Fehlers durch Anpassung von $\vec{w}$ und zwar $\forall \vec{x} \in \mathbf{D}$ wobei $\mathbf{D}$ der Definitionsbereich von $F\left(\vec{x}\right)$ ist.

Im allgemeinen steht allerdings die Funktion $F\left(\vec{x}\right)$ nicht explizit zur Verfügung sondern etwa nur in Form gemessener Werte für eine definierte Auswahl von Argumenten $\vec{x}$. Sei Ω die Menge der Argumente für die die Funktionswerte $F\left(\vec{x}\right)$ bekannt sind (z. B. durch Messung). Diese Menge der bekannten Instanzen bildet die sog. Trainingsmenge oder Beispielmenge

$$\mathcal{T} = \{Z_{\vec{x}}\} = \{(\vec{x}, F\left(\vec{x}\right)) \mid \vec{x} \in \Omega\} \tag{2.4}$$

für die Bestimmung des optimalen Satzes von Parametern $\vec{w}$ durch Minimierung des Fehlers 2.3.

Die Suche nach dem besten $\vec{w}$ geht in einer Abfolge von **Lernschritten** vonstatten, wobei für jede Trainingsinstanz $\left(\vec{x}, y^{soll}\right)$, $\vec{x} \in \Omega$ der Parametervektor $\vec{w}$ um ein $\Delta\vec{w}$ (Update) gerade so verschoben wird, daß sich die Kosten verringern. Dazu muß längs jeder Achse des Koordinatensystems die Verschiebung umgekehrt proportional zur Ableitung der Kostenfunktion gewählt werden, der Lernschritt ist also

Lernschritt

$$\Delta w_i = -\epsilon \frac{\partial E}{\partial w_i} \quad \forall i \tag{2.5}$$

Der Vektor $\vec{G} = grad_{\vec{w}} E = \left(\frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, \ldots, \frac{\partial E}{\partial w_n}\right)$ ist der Gradient der Kostenfunktion, der in die Richtung des steilsten **Anstieges** von E zeigt. Das Verfahren 2.5 ist demzufolge (wegen des negativen Vorzeichens) ein Gradien-

Gradientenabstieg

tenabstieg**abstieg** auf der Landschaft der Kosten. Explizit gilt

$$\boxed{\Delta w_i = \epsilon dx_i} \tag{2.6}$$

oder vektoriell

$$\boxed{\Delta\vec{w} = \epsilon d\vec{x}}$$

wobei

$$d = \left(y^{soll} - y\right) S'\left(\vec{x} \cdot \vec{w}\right) \tag{2.7}$$

mit $S'(z) = dS(z)/dz$. Die sog. Fehleraktivität d ist proportional zur Abweichung zwischen dem aktuellen Output $y = f(\vec{x})$ des Neurons und der Sollvorgabe $y^{soll} = F(\vec{x})$ für den betrachteten Input $\vec{x}$.

Der Algorithmus konvergiert gegen ein mittleres (lokales) Minimum der Kostenfunktion wenn alle Instanzen hinreichend oft präsentiert werden und der Lernparameter ϵ geeignet "abgekühlt" wird. In der Praxis bedeutet das, daß ϵ im Laufe des Lernens von einem Startwert $\epsilon \approx 0.1$ beginnend auf Werte $\epsilon = 0.01$ bis zu $\epsilon = 0.001$ reduziert wird: Sehr kleine ϵ bedeuten sehr kleine Lernschritte, so daß nur im Mittel über viele Trainingsinstanzen eine merkliche Verschiebung von $\vec{w}$ zustande kommen kann. Die Bewegung von $\vec{w}$ ist demzufolge eine systematische Bewegung (hin zum Minimum) überlagert von kleinen Fluktuationen. Geht ϵ hinreichend langsam gegen Null, so mitteln sich die Fluktuationen immer besser heraus, ohne daß die systematische Bewegung abgebrochen wird. In der Grenze konvergiert der Algorithmus damit gegen das lokal optimale $\vec{w}$. Für die richtige Strategie zur Abkühlung gibt es eine genaue definierte Strategie, die streng aus dem Robins–Monroe–Theorem folgt, s. [9].

Dabei können die Instanzen als Ergebnisse von Messungen auch "verrauscht" sein, der Lernalgorithmus 2.6 konvergiert dann gegen ein $\vec{w}$ so, daß $f(\vec{x} \mid \vec{w})$ den Funktionszusammenhang im Mittel repräsentiert.

Ist $S(z) = z$ (lineares Neuron), so ist nach dem Lernen $f(\vec{x} \mid \vec{w})$ mit dem Ergebnis der Methode der linearen Regression identisch.

2.1.1 On–line Lernen vs. batch Lernen

Im Lernalgorithmus 2.7 wird für jeden Input $\vec{x}$ sofort der Lernschritt ausgeführt, deshalb ist das ein on–line Algorithmus. Der besondere Vorteil dieses Lernverfahrens besteht in seiner Echtzeitfähigkeit, er nähert sich inkrementell dem Optimum und kann deshalb nach jedem Einzelschritt abgebrochen werden, eventuell um den Preis eines suboptimalen Ergebnisses. Außerdem werden neu hinzukommende Trainingsbeispiele zwanglos integriert.

Das eigentliche Ziel ist aber natürlich, den Fehler im Mittel über alle Inputs, d. h. für alle $\vec{x} \in \Omega$ zu minimieren, die zugehörige Fehlerfunktion ist dann durch den Mittelwert aller Einzelfehler gegeben

$$E(\vec{w}) = \sum_{\vec{x} \in \Omega} E(\vec{x} \mid \vec{w}) P(\vec{x}) \tag{2.8}$$

wobei $P(\vec{x})$ die Wahrscheinlichkeit (relative Häufigkeit) für den Input $\vec{x}$ ist. Gradientenabstieg auf dieser Kostenfunktion entspricht einem sog. batch Algorithmus, bei dem also alle Inputs erst gesammelt werden müssen, um E berechnen und dann den Gradientenabstieg durchführen zu können. Der on–line Algorithmus ist die stochastische Approximation des batch Algorithmus. Bei geeigneter Abkühlungsstrategie für die Lernrate ϵ konvergiert der Algorithmus mit Wahrscheinlichkeit 1 gegen das Minimum von $E(\vec{w})$.

2.1.2 Beipiele

Konvergenzverhalten bei quadratischer Fehlerfunktion

Wir betrachten zunächst den einfachen Fall, daß die Kostenfunktion nicht von x abhängt und daß diese quadratisch sei

$$E(\vec{w}) = \sum_{ij} w_i A_{ij} w_j \tag{2.9}$$

wobei das w–Koordinatensystem willkürlich so gewählt wurde, daß am Minimum $\vec{w} = 0$. Die Form 2.9 der Fehlerfunktion ist allgemein in der Umgebung eines lokalen Minimums gültig. Sie ergibt sich aus einer Taylorentwicklung um das Minimum unter Berücksichtigung von $\partial E/\partial w_i = 0 \quad \forall i$ am Minimum. Dann ist $A_{ij} = A_{ji}$ so daß die quadratische Form diagonalisiert und damit das Problem auf die Betrachtung des eindimensionalen Falles reduziert werden kann, d. h. wir betrachten (mit $a > 0$)

$$E = \frac{1}{2} a w^2$$

und folglich

$$\Delta w = -\epsilon a w \tag{2.10}$$

wobei a die Öffnungsweite der Parabel ist. Dieser Parameter macht also eine Aussage über die Steilheit der Abhänge des Tales. Sei t mit $t = 0, 1, ...$ der Schrittzähler der Lernschritte, so entspricht Gleichung 2.10

$$w(t+1) = (1 - \epsilon a)\, w(t) \tag{2.11}$$

mit Lösung

$$w(t) = e^{-t/\tau} w(0) \quad \text{falls} \quad \epsilon a < 1 \tag{2.12}$$

wobei

$$\tau = -\frac{1}{\ln(1 - \epsilon a)} \tag{2.13}$$

Von einem beliebigen Wert $w(0)$ startend konvergiert für $\epsilon a < 1$ der Algorithmus offensichtlich exponentiell mit Zeitkonstante τ gegen das Minimum. Für hinreichend kleine ϵ so daß $\epsilon a << 1$ folgt durch Taylorentwicklung

Zeitkonstante des Lernens

$$\tau = \frac{1}{\epsilon a} \quad \text{für} \quad \epsilon a << 1 \tag{2.14}$$

Die Lernrate ϵ bestimmt somit direkt die Konvergenzgeschwindigkeit gegen das Minimum der Fehlerfunktion.

Formel 2.12 ist allerdings nur für $\epsilon a < 1$ gültig. Für $\epsilon a = 1$ wird das Minimum in einem Schritt erreicht. Das ist formal in 2.13 enthalten, da für $\epsilon a \to 1$ die Zeitkonstante $\tau \to 0$ konvergiert, was einer unendlich großen Konvergenzgeschwindigkeit des Algorithmus entspricht. Für $1 < a\epsilon < 2$ ergibt sich eine alternierende aber noch konvergente Folge und für $\epsilon a > 2$ divergiert der Lernalgorithmus schließlich.

Divergenz des Algorithmus bei zu großen Lernraten

Das Neuron als Lehrer und Schüler

Wir betrachten – der Einfachkeit halber für einen eindimensionalen Input – den Fall, daß ein Neuron mit fester synaptischer Kopplungsstärke $w = 1$ als Lehrer die Solloutputs für ein Schülerneuron vorgibt, d h.

$$F(x) = \frac{1}{1 + e^{-x}}$$

Wie aus der Abbildung 2.1 klar ersichtlich hat die Kostenfunktion

$$E = \frac{1}{2}\left(\frac{1}{1 + e^{-wx}} - \frac{1}{1 + e^{-x}}\right)^2$$

das erwartete globale Minimum bei $w = 1$, so daß das Schülerneuron durch Gradientenabstieg tatsächlich exakt das Verhalten des Lehrers erlernen kann. Dieses Minimum ist unabhängig von x, so daß der Lernalgorithmus auch für feste Werte von ϵ konvergiert.

Synaptische Divergenz und synaptischer Kollaps

Eine häufige "Fehlleistung" beim Lernen sind Divergenz oder Kollaps der synaptischen Kopplungen. Das tritt immer dann auf, wenn die Sollvorgaben mit der Transferfunktion des Neurons inkompatibel sind, wir betrachten z. B. den Fall $F(x) = 1$ mit Fehlerfunktion

$$E = \frac{1}{2}\left(\frac{1}{1 + e^{-xw}} - 1\right)^2 \tag{2.15}$$

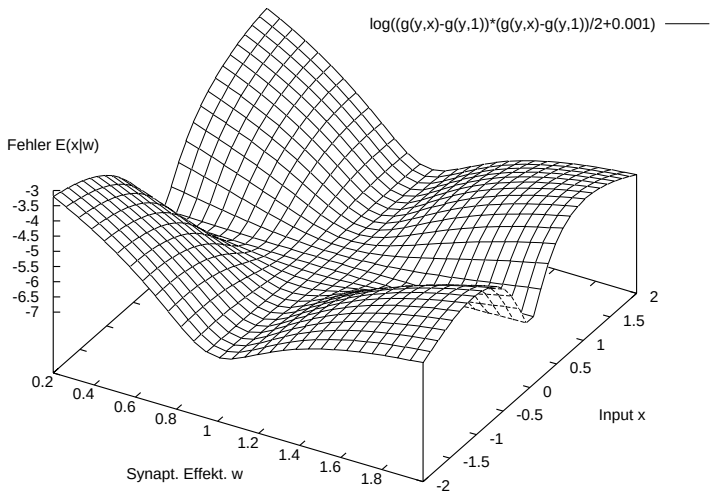


Abbildung 2.1: Die Fehlerfunktion für das Training eines Schülerneurons durch ein Lehrerneuron mit $w = 1$. Da die eigentliche Fehlerfunktion sehr flach ist, wurde zur besseren Sichtbarmachung des Minimums bei $w = 1$ die Darstellung $\log\left(E\left(x \mid w\right) + 0.001\right)$ für den Fehler gewählt.

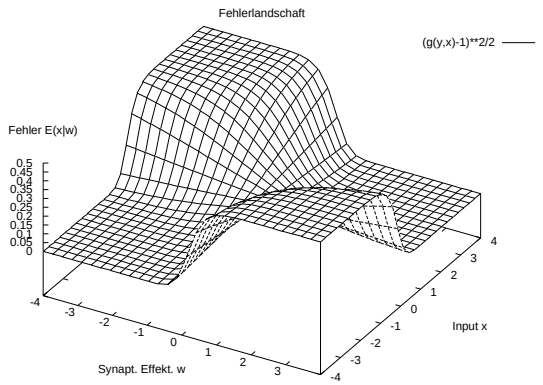


Abbildung 2.2: Die Fehlerlandschaft für das Erlernen der Zielfunktion $F = 1$.

Die Fehlerfunktion ist urpsrungssymmetrisch. Wie schon aus der Fehlerfunktion explizit ersichtlich treibt für $x > 0$ die Gradientenabstiegsdynamik $w \to -\infty$, für $x < 0$ folgt analog $w \to \infty$.Werden also ausschließlich Werte aus dem einen oder anderen Bereich von x angeboten, so divergiert die synaptische Kopplungsstärke. Werden andererseits Beispiele mit sowohl $x > 0$ als auch $x < 0$ angeboten, so konvergiert w gegen einen (kleinen) Wert. Sind die x Werte im Vorzeichen gleichverteilt, so konvergiert w gegen Null. Man kann hier vom Kollaps der Synapsen sprechen oder auch von der Selbstabschaltung des Neurons als Reaktion auf Überforderung durch die aus der Sicht des Neurons widersprüchlichen Sollvorgaben.

Synaptische Divergenz

Synaptischer Kollaps durch inkonsistente Lehrer

Beide Phänomene – synaptischer Kollaps wie synaptische Divergenz – treten leicht in den feed–forward Netzen (s.u.) auf, die nach der Methode der Fehlerrückpropagierung trainiert werden.. Danach erhält ein jedes Neuron die Sollvorgaben im Lernschritt von ihm nachgeordneten Neuronen zugewiesen. Ist die Architektur unglücklich gewählt, so können die Forderungen (Sollwerte y_j^{soll}), die an ein bestimmtes Neuron j im Netz auf diese Weise herangetragen werden, mit der Transferfunktion des Neurons inkompatibel sein. Im Umkehrschluß können aus dem Verhalten der Synapsen im Lernprozeß Aussagen über eventuell erforderliche Veränderungen der Architektur abgeleitet werden.

Synapsendynamik und Netzwerkarchitektur

Erlernen einer komplizierteren Funktion

Als Beispiel für eine kompliziertere Fehlerlandschaft betrachten wir die Approximation der sin–Funktion durch ein Neuron, d. h. wir setzen

$$F(x) = \frac{1}{2}(\sin x + 1) \tag{2.16}$$

so daß $y^{soll} \in [0,1]$ und

$$f(x) = \frac{1}{1 + e^{-wx}} \tag{2.17}$$

wie gehabt. Für $w \approx 2$ ergibt sich für $x \in [-2,2]$ eine recht gute Übereinstimmung zwischen f und F, s. Abbildung 2.3, für Werte außerhalb dieses Intervalls wird die Übereinstimmung aber beliebig schlecht.

Die Fehlerfunktion $E(x \mid w)$ ist durch eine recht komplizierte Struktur gekennzeichnet. Wir beobachten einerseits für $x \in [-2,2]$ um $w = 2$ einen relativ konzentrierten Minimumsbereich, was die recht gute Übereinstimmung zwischen f und F in diesem Bereich widerspiegelt, s. Abbildungen 2.3 und 2.5. Anderer-

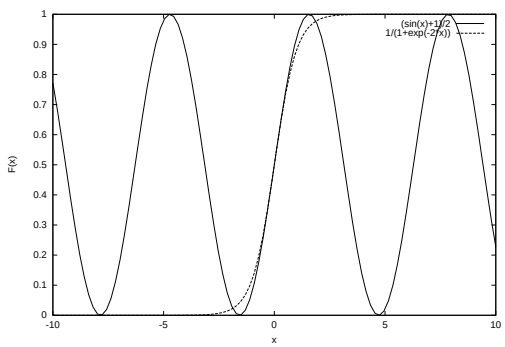


Abbildung 2.3: Die Funktion $F(x) = \frac{1}{2}(\sin x + 1)$ und die Transferfunktion $f(x) = 1/(1 + e^{-xw})$ für $w = 2$.

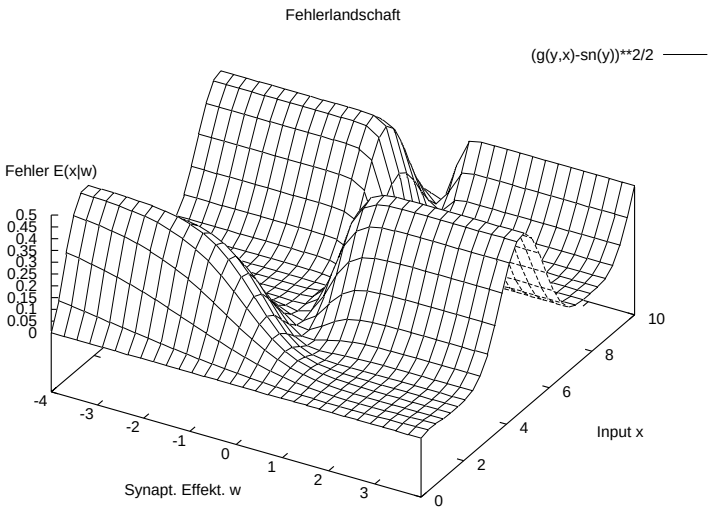


Abbildung 2.4: Die Fehlerfunktion für die Approximation der Funktion $F(x) = (\sin x + 1)/2$ durch ein Neuron mit Transferfunktion $f(x) = \frac{1}{1+e^{-wx}}$.

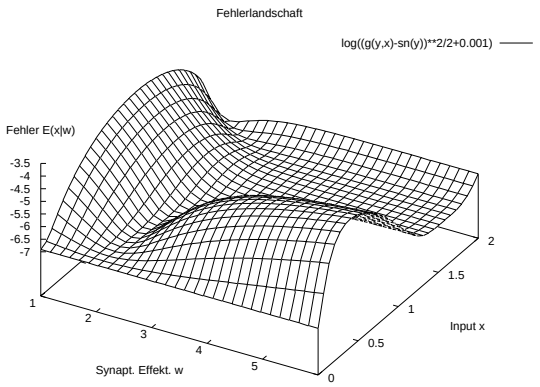


Abbildung 2.5: Die Fehlerfunktion für die Approximation der Funktion $F(x) = (\sin x+1)/2$ durch ein Neuron im Bereich $x \in [0, 2]$. Obwohl das Neuron mit $w \approx 2$ die Zielfunktion $F(x)$ in dem betrachteten Bereich gut approximiert hängt das Minimum von $E(x \mid w)$ deutlich von x ab. Da die eigentliche Fehlerfunktion sehr flach ist, wurde $\log(E(x \mid w) + 0.001)$ dargestellt.

seits hängt, wie aus Abbildung 2.6 ersichtlich, außerhalb dieses Bereiches das Minimum von E sehr stark von x ab.

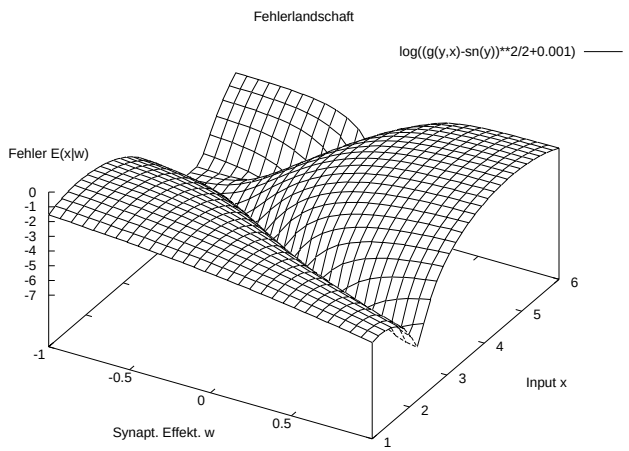


Abbildung 2.6: Die Fehlerfunktion für die Approximation der Funktion $F(x) = (\sin x + 1)/2$ durch ein Neuron im Bereich $x \in [2, 6]$. Das Neuron approximiert $F(x)$ in diesem Bereich sehr schlecht, was durch die komplizierte Fehlerlandschaft reflektiert wird. Dargestelt wurde wieder $\log(E(x \mid w) + 0.001)$.

2.2 Das Neuron als lernender Klassifikator

Sei $\mathcal{K}$ eine Menge von Datenpunkten, die in zwei Teilmengen (Klassen) $\mathcal{K}_1, \mathcal{K}_2$ zerfällt, mit $\mathcal{K}_1 \cap \mathcal{K}_2 = \emptyset$. Ein Klassifikator ist eine Funktion $x \longmapsto K(\vec{x})$ mit der Eigenschaft

$$K(\vec{x}) = \begin{cases} 1 & \text{für} \quad \vec{x} \in \mathcal{K}_1 \\ \\ 0 & \text{für} \quad \vec{x} \in \mathcal{K}_2 \end{cases} \tag{2.18}$$

Ein Neuron mit Transferfunktion $y = f(\vec{x})$ und einer sigmoiden Ausgabefunktion kann diese Funktion K nur repräsentieren, wenn wir zusätzlich eine Akzeptanzschwelle einführen, d. h. wir definieren (für den Fall der sigmoiden Ausgabefunktion) die Klassifikatorfunktion $K(\vec{x})$ in Termen der Transferfunktion $f(\vec{x})$ des Neurons als

$$K(\vec{x}) = \begin{cases} 1 & \text{falls} \quad f(\vec{x}) > 1 - k \qquad \text{(Klasse 1)} \\ \\ 0 & \text{falls} \quad f(\vec{x}) < k \qquad \text{(Klasse 2)} \end{cases} \tag{2.19}$$

wobei $0 < k < 1/2$. Die Menge der $\vec{x}$, für die $k \leq K(\vec{x}) \leq 1 - k$ gilt, bilden die sog. Rückweisungsklasse, da sie durch den Klassifikator nicht klassifiziert

werden. Ist speziell $k = 1/2$, dann ist

$$K = 1 \quad falls \quad \vec{w} \cdot \vec{x} > U \qquad \text{(Klasse 1)} \tag{2.20}$$

$$K = 0 \quad falls \quad \vec{w} \cdot \vec{x} < U \qquad \text{(Klasse 2)}$$

Das Klassifikationsproblem ist durch das Neuron lösbar, wenn die beiden Teilmengen linear separabel sind. Das ist allgemein dann der Fall, wenn eine Trennebene existiert, so daß jede Teilmenge auf jeweils einer Seite der Ebene liegt. Die Gleichung $f(\vec{x}) = S(\vec{w} \cdot \vec{x} - U) = 1/2$ bzw.

$$\vec{w} \cdot \vec{x} = U$$

oder

$$\sum_{i=1}^{n} w_i x_i = U \tag{2.21}$$

definiert eine zum Vektor $(w_1, ..., w_n)$ orthogonale (Hyper–)Ebene im Abstand $U/|\vec{w}|$ zum Ursprung des Koordinatensystems. Das Neuron kann damit ein beliebiges linear separables Klassifikationsproblem lösen, wenn nur $\vec{w}$ (und U) so gewählt ist, daß die Menge der Datenpunkte $\mathcal{K}_1$ oberhalb (in Richtung des Vektors $(w_1, ..., w_n)$ gesehen) und die der Menge $\mathcal{K}_2$ unterhalb dieser Trennebene liegt.[1]

Diese Eigenschaft kann das Neuron nach obiger Regel erlernen[2]. Als Trainigsinstanzen dienen dabei die Datenpunkte $\vec{x} \in \mathcal{K}$ mit der dazugehörigen Klassenzuweisung $y^{soll} = K(\vec{x}) \in \{0, 1\}$, d. h.

$$\mathcal{T} = \{(\vec{x}, K(\vec{x}))\}$$

Damit ist das Neuron ein lernender Klassifikator für jedes beliebige lineare Klassifikationsproblem. Umgekehrt kann das Neuron nichtlineare Klassifikationsprobleme bestenfalls nur näherungsweise lösen.

[1] Falls $k < 1/2$ gewählt wurde, die Rückweisungsklasse also nicht leer ist, wird die Grenze zwischen den klassifizierbaren Objekten durch einen Streifen markiert, dessen Breite eine Funktion von k ist.

[2] Dazu ist es günstig, die Schwelle durch die Einführung eines zusätzlichen Inputs einzubeziehen, wie in Kapitel 1 diskutiert. Dadurch wird die Schwelle in Form der synaptischen Verbindungsstärke zu diesem Input mitgelernt.

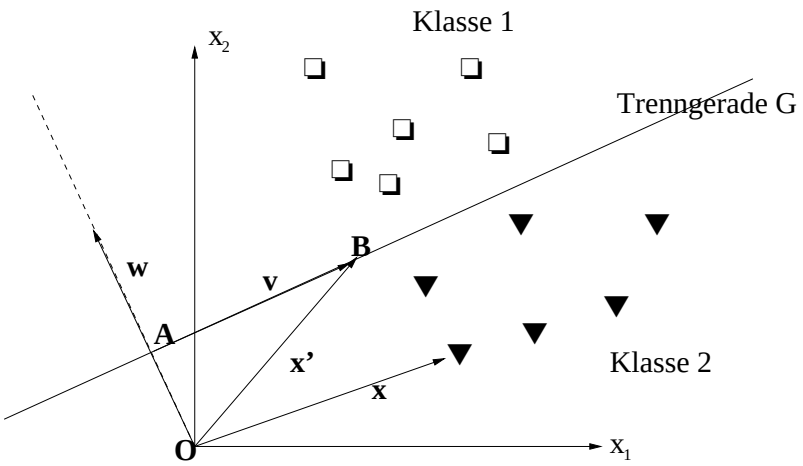


Abbildung 2.7: Die durch Gleichung 2.21 definierte Trenngerade. Elemente der Klassen 1 bzw. 2 sind durch Quadrate bzw. Dreiecke dargestellt. Jedes Element einer Klasse ist durch seinen Ortsvektor $\vec{x}$ repräsentiert. Der Abstand OA ist gleich $U/|\vec{w}|$, wobei $|\vec{w}|$ die Länge des Vektors $\vec{w}$ ist. Der Vektor $\vec{v}$ ist die gerichtete Strecke von A nach B. Für jeden Punkt B, (dargestellt durch seinen Ortsvektor $\vec{x}'$) auf der Trenngeraden gilt $\vec{v}\vec{w} = 0$ was gerade Gleichung 2.21 entspricht.

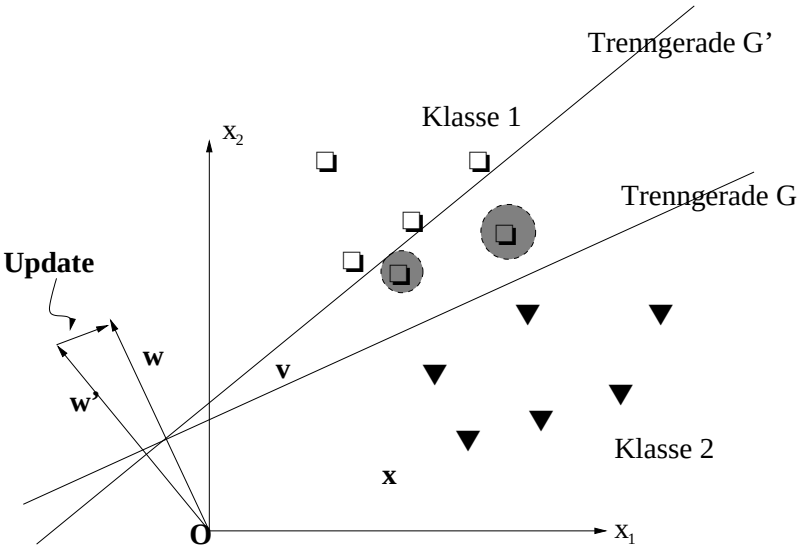


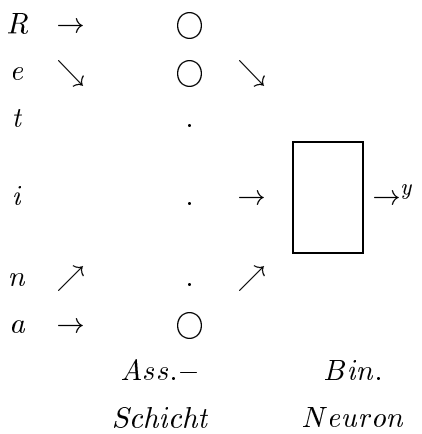
Abbildung 2.8: Fehlklassifikationen können durch Variation $\vec{w}' \rightarrow \vec{w}$ des Vektors $\vec{w}$ der synaptischen Effektivitäten korrigiert werden.

2.3 Das Perzeptron

Das Perzeptron spielte in der Frühphase des Studiums neuronaler Systeme als Medien der KI eine zentrale Rolle. Dieses besteht aus einer (künstlichen) Retina, die über synaptische Verbindungen die binären Neuronen der sog. Assoziationsschicht speist. Diese koppeln die binären Outputs $x_1, ..., x_n$, $x_i \in \{0, 1\}$ aus, die als Inputs eines binären Neurons mit Transferfunktion

$$y = \theta\left(\vec{w} \cdot \vec{x} - U\right) \tag{2.22}$$

dienen, das die Klassifikation der auf die Retina projizierten Muster vornimmt.



(2.23)

Die Vorverarbeitungsschicht sollte im wesentlichen die Umsetzung der retinalen Information in eine linear seperable Klassifikationsaufgabe bewerkstelligen, die dann durch ein formales Neuron lösbar wäre.

Damit ist das Perzeptron ebenfalls ein lernender Klassifikator, allerdings ist hier der oben abgeleitete Lernalgorithmus nicht einsetzbar, da die Ableitung $S'(z)$ für die unstetige Funtkion $S(z) = \theta(z)$ an $z = 0$ nicht existiert. Das richtige Inkrement von $\vec{w}$ im Lernschritt kann man sich durch die folgenden Überlegungen herleiten:

1. Falls das präsentierte $\vec{x}$ durch das Perzeptron richtig klassifiziert wurde, muß $\vec{w}$ nicht geändert werden, d. h. $\Delta\vec{w} = 0$.

2. Bei Fehlklassifikation gilt für $\vec{x} \in \mathcal{K}_1$, daß $z = \vec{w} \cdot \vec{x} < U$. [3] Die Wahl $\Delta\vec{w} = \epsilon\vec{x}$ (wobei $0 < \epsilon < 1$) bewirkt die Vergrößerung des PSP um die

[3] Wir nehmen an, daß die Schwelle richtig gewählt wurde, sonst kann diese wieder durch die Mitnahme eines fiktiven Zusatzinputs mitgelernt werden.

positive Größe $\Delta z = \epsilon \vec{x}^2$ und ist damit ein Schritt in die richtige Richtung, da ja bei richtiger Klassifikation $z > U$ sein muß vgl. Abb. 2.9. Analog gilt für $\vec{x} \in \mathcal{K}_2$, daß $\Delta \vec{w} = -\epsilon \vec{x}$ gewählt werden muß.

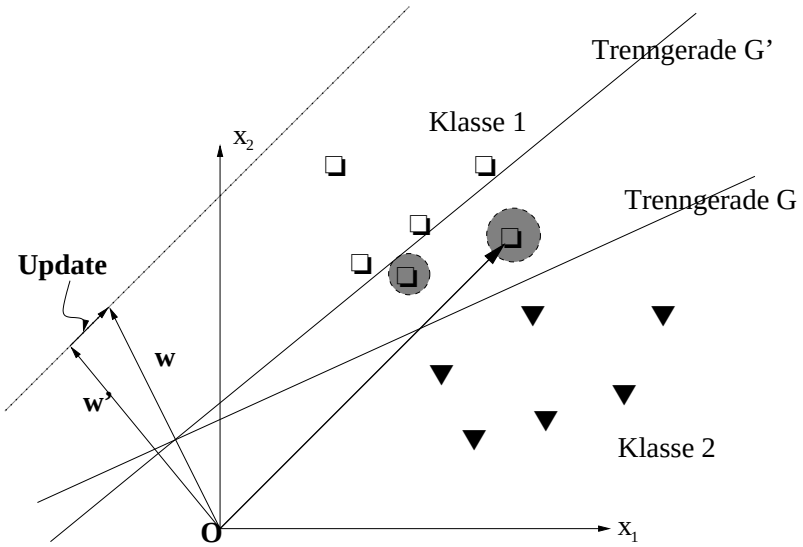


Abbildung 2.9: Zur Perzeptron–Lernregel: Der Input mit Vektor x ist fehlklassifiziert. Der Update $\Delta \vec{w} = \epsilon \vec{x}$ verschiebt $\vec{w}$ in die richtige Richtung. Die gepunktete Linie verläuft parallel zum Vektor $\vec{x}$. Durch den Update verändert sich nicht nur die Richtung sondern auch die Länge des Vektors $\vec{w}$, was einer Änderung der Schwelle entspricht. Das kann vermieden werden, wenn die Schwelle durch Einführung eines fiktiven Inputs x_0 wie eine zusätzliche Synapse behandelt werden kann.

Das kann in die Perzeptron–Lernregel

$$\boxed{\Delta \vec{w} = \epsilon d \vec{x}} \tag{2.24}$$

mit

$$\boxed{d = y^{soll} - y}$$

zusammengefaßt werden wobei $K(\vec{x}) = y^{soll} = 1$ falls $x \in \mathcal{K}_1$ und $y^{soll} = 0$ sonst. Gleichung 2.24 kann wieder im Sinne einer Hebbschen Regel interpretiert werden, wenn wir die Fehleraktivität $d = \left(y^{soll} - y\right)$ als Outputaktivität

des lernenden Neurons verstehen. Für diese Lernregel existiert ein Konvergenzbeweis, der zeigt, daß der Algorithmus nach einer **endlichen** Zahl von Schritten konvergiert, falls die Klassen linear trennbar sind und ϵ geeignet gewählt wurde.

2.4 Grenzen des Neurons – das XOR Problem

Binäre Neuronen mit binären Inputs $x_i \in \{0,1\}$, $i = 1, ..., n$ stellen eine Boolsche Funktion $\{0,1\}^n \mapsto \{0,1\}$ dar. Allerdings ist die Realisierung Boolscher Funktionen durch ein einzelnes Neuron in vielen Fällen unmöglich. Schon die wichtige zweistellige Boolsche Funktion XOR

$$\begin{array}{ccc} x_1 & x_2 & y \\ 0 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{array} \tag{2.25}$$

ist nicht mehr realisierbar. In der Tat, die Inputs $\vec{x}$ liegen auf den Eckpunkten des Einheitsquadrates $[0,1]\times[0,1]$ in der $x_1 - x_2$–Ebene wobei sich die Repräsentanten für jede der beiden Klassen diagonal gegenüberliegen. Sie sind deshalb durch eine Gerade nicht trennbar, so daß das Problem nicht linear separabel ist. Eine Lösung ergibt sich erst in der Zusammenschaltung mehrerer Neuronen zu einem neuronalen Netz, s. Kapitel 3.

Weiterführende Literatur: Geeignete Lehrbücher sind [34, 45, 31, 22], wobei in [31] eine besonders anschauliche Darstellungsweise gefunden wurde.

Kapitel 3

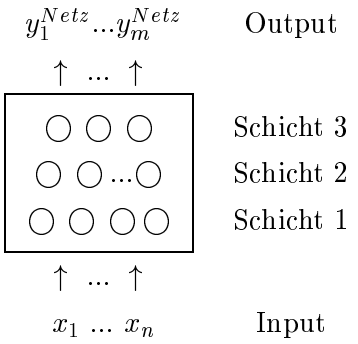
Feed–Forward Netze

Künstliche Neuronale Netze sind für viele Anwendungen in doppelter Hinsicht interessant. Einerseits werden sie als Werkzeuge zur Funktionsapproximation und zum Aufbau einer inneren Repräsentation der Außenwelt ausgenutzt. Andererseits dienen sie z. B. in der biologisch orientierten Roboterforschung als willkommene Medien zur Realisierung besonders biologienaher Roboter. Dabei sind die neuronalen Netze besonders wegen ihrer Lernfähigkeit attraktiv – sie können eine Aufgabe anhand von Beispielen erlernen statt aufwendig programmiert werden zu müssen. Zudem können sie direkt in Hardware realisiert werden (Neurocomputer), was zu extrem kurzen Verarbeitungszeiten führt.

3.1 Informationsverarbeitung mit Feed–Forward Netzen

Das einzelne Neuron kann nur eine pseudolineare Approximation einer beliebigen Funktion liefern. Will man komplizierte Funktionen gut approximieren, muß man mehrere Neuronen geeignet zusammenschalten. Feed-Forward Netze bestehen aus mehreren (sog. verborgenen) Schichten, die zwischen die Input–

und Output–Einheiten geschaltet sind.



(3.1)

Propagation der Information

Die Neuronen einer Schicht erhalten dabei ihre Inputs von den Neuronen der jeweils davorgeschalteten Schicht, daher der Name feed–forward Netzwerk. I. a. besteht ein Netz aus aus M Schichten $s = 1, ..., M$ zu jeweils $n^{(s)}$ Neuronen. Das Netz in 3.1 besteht aus $M = 3$ Schichten, wobei die $n^{(3)} = 3$ **Output**neuronen die Schicht 3 bilden und die Schichten $1, 2$ verborgen sind. Die Terminologie ist hier nicht ganz eindeutig geregelt, manche Autoren bezeichnen auch die Schicht 1 als Inputschicht, so daß im Beispiel 3.1 nur die Schicht 2 verborgen wäre. Wir schließen uns aber der ersteren Sprechweise an, d. h. alle Neuronen, die keine Outputneuronen sind, bilden die verborgenen Schichten. Wir schreiben

Architektur Notation

symbolisch die Architektur des Netzes in der Form $n - n^{(1)} - n^{(2)} - ... - n^{(M)}$ (wobei $n^{(M)} = m$) für ein Netz aus M Schichten mit n Inputs und m Outputs. Das Netz in 3.1 wird (mit $m = 3$) folglich als $n - 4 - n^{(2)} - 3$ geschrieben.

In jeder Schicht s ergibt sich das postsynaptische Potential (PSP) $z_i^{(s)}$des Neurons i aus der Linearkombination der Inputs $\vec{y}^{(s-1)}$ in diese Schicht mit den Komponenten des Gewichtsvektors $\vec{w}_i^{(s)}$ des Neurons

$$z_i^{(s)} = \vec{w}_i^{(s)} \cdot \vec{y}^{(s-1)} = \sum_{k=1}^{n^{(s)}} w_{i,k}^{(s)} \, y_k^{(s-1)} \tag{3.2}$$

Der **Output** desNeurons i in Schicht s ist

$$y_i^{(s)} = S\left(z_i^{(s)}\right) = S\left(\vec{w}_i^{(s)} \cdot \vec{y}^{(s-1)}\right) \tag{3.3}$$

wobei mit der Vereinbarung $\vec{y}^{(0)} = \vec{x}$ und $\vec{y}^{(M)} = \vec{y}$ die In– und Outputs mit eingeschlossen sind.

Transferfunktion

Die Transferfunktion des gesamten Netzes ist folglich eine verschachtelte Funktion, die rekursiv aus 3.4 bei den Output–Einheiten beginnend aufgebaut

werden kann. Ist $\vec{\mathbf{W}} = \left\{ \vec{w}_i^{(s)} \right\}$ die Menge aller Gewichtsvektoren dann ist die Transferfunktion des Netzes

$$\boxed{\vec{y} = \vec{f}^{Netz}\left(\vec{x} \mid \vec{\mathbf{W}}\right)} \tag{3.4}$$

Die FFN sind universelle Funktionsapproximatoren wie in Abschnitt 3.5.1 näher dargestellt.

3.2 Der Backpropagation–Algorithmus

3.2.1 Die Update–Regel

Zur Bestimmung des optimalen $\vec{\mathbf{W}}$ konstruieren wir analog zu **??** eine Kostenfunktion

$$\begin{aligned} E\left(\vec{x} \mid \vec{\mathbf{W}}\right) &= \tfrac{1}{2}\left(\vec{y}^{Netz} - \vec{y}^{soll}\right)^2 = \tfrac{1}{2}\sum_{i=1}^{M} \tfrac{1}{2}\left(y_i^{Netz} - y_i^{soll}\right)^2 \\ &= \tfrac{1}{2}\left(\vec{f}^{Netz}\left(\vec{x} \mid \vec{\mathbf{W}}\right) - \vec{F}\left(\vec{x}\right)\right)^2 \end{aligned} \tag{3.5}$$

Für den Lernschritt (*Update*) ergibt sich (Gradientenabstieg) für ein $\vec{x} \in \Omega$ für jedes Neuron k in jeder Schicht s das Inkrement zu

$$\Delta w_{ik}^{(s)} = -\epsilon \frac{\partial E}{\partial w_{ik}^{(s)}} \tag{3.6}$$

Die Ausführung der Differentiation nach der Kettenregel zeigt, daß dieses immer in der Form

Update Regel

$$\boxed{\Delta w_{ki}^{(s)} = -\epsilon\, d_k^{(s)} y_i^{(s-1)}} \tag{3.7}$$

analog zu 2.6 geschrieben werden kann, wobei die Fehleraktivität am Output wieder

$$d_k^{(M)} = \left(y_k^{Netz} - y_k^{soll}\right) S'\left(z_k^{(M)}\right) \tag{3.8}$$

ist, vgl. 2.7. Für die Neuronen in den verborgenen Schichten stehen explizit keine Sollvorgaben zur Verfügung. Diese können aber wie die Ableitung zeigt (s.u.) aus den Fehleraktivitäten der darüberliegenden Schichten rekursiv gemäß

Propagation der Fehleraktivitäten

$$\boxed{d_k^{(s)} = \sum_{i=1}^{n^{(s+1)}} w_{ik}^{(s+1)} d_i^{(s+1)}} \tag{3.9}$$

ermittelt werden. Die Fehleraktivität zur Korrektur der Gewichte des Neurons k ergibt sich als die gewichtete Summe der Fehleraktivitäten aller der Neuronen, die vom betrachteten Neuron gespeist werden. Formeln 3.7 und 3.9 bilden den Backpropagation–Algorithmus. Der Fehler wird durch die Rekursion von 3.9 durch das Netz hindurch rückpropagiert. Daher der Name des Algorithmus.

3.2.2 Anhang: Herleitung des Backprop-Algorithmus für Feed-Forward Netzwerke

Für die Ableitung der Kostenfunktion nach den synaptischen Vektoren einer beliebigen Schicht s gilt

$$\begin{aligned} &\frac{\partial}{\partial w_{k,l}^{(s)}} E\left(\vec{x}, \vec{W}\right) = \sum_{i=1}^{n^{(\mathcal{M})}} \left(y_i^{(\mathcal{M})} - y_i^{soll}\right) \frac{\partial}{\partial w_{k,l}^{(s)}} y_i^{(\mathcal{M})} = \\ &\sum_{i=1}^{n^{(\mathcal{M})}} \left(y_i^{(\mathcal{M})} - y_i^{soll}\right) S'\left(z_i^{(\mathcal{M})}\right) \frac{\partial}{\partial w_{k,l}^{(s)}} z_i^{(\mathcal{M})} = \sum_{i=1}^{n^{(\mathcal{M})}} d_i^{(\mathcal{M})} \frac{\partial}{\partial w_{k,l}^{(s)}} z_i^{(\mathcal{M})} \end{aligned} \tag{3.10}$$

wobei dfer modifizierte Output-Fehler

$$d_i^{(\mathcal{M})} = \left(y_i^{(\mathcal{M})} - y_i^{soll}\right) S'\left(z_i^{(\mathcal{M})}\right) \tag{3.11}$$

wie früher definiert wurde. Für die output-Schicht wird (3.10)

$$\frac{\partial}{\partial w_{k,l}^{(\mathcal{M})}} E\left(\vec{x}, \{\vec{w}\}\right) = d_k^{(\mathcal{M})} y_l^{(\mathcal{M}-1)} \tag{3.12}$$

Für die darunterliegende Schicht mit $s = \mathcal{M} - 1$ folgt

$$\begin{aligned} &\frac{\partial}{\partial w_{k,l}^{(s)}} E\left(\vec{x}, \{\vec{w}\}\right) = \sum_{i=1}^{n^{(\mathcal{M})}} d_i^{(\mathcal{M})} \frac{\partial}{\partial w_{k,l}^{(s)}} z_i^{(\mathcal{M})} \\ &= \sum_{i=1}^{n^{(\mathcal{M})}} \sum_{j=1}^{n^{(\mathcal{M}-\infty)}} d_i^{(\mathcal{M})} \frac{\partial}{\partial w_{k,l}^{(s)}} w_{i,j}^{(\mathcal{M})} y_j^{(\mathcal{M}-1)} \\ &= \sum_{i=1}^{n^{(\mathcal{M})}} \sum_{j=1}^{n^{(\mathcal{M}-\infty)}} d_i^{(\mathcal{M})} w_{i,j}^{(\mathcal{M})} \frac{\partial}{\partial w_{k,l}^{(s)}} S\left(\vec{w}_j^{(\mathcal{M}-1)} \cdot \vec{y}^{(\mathcal{M}-2)}\right) \\ &= \sum_{i=1}^{n^{(\mathcal{M})}} d_i^{(\mathcal{M})} w_{i,k}^{(\mathcal{M})} S'\left(z_k^{(\mathcal{M}-1)}\right) y_l^{(\mathcal{M}-2)} \end{aligned}$$

und damit

$$\frac{\partial}{\partial w_{k,l}^{(s)}} E\left(\vec{x}, \{\vec{w}\}\right) = d_k^{(\mathcal{M}-1)} y_l^{(\mathcal{M}-2)}, \qquad \text{für} \quad s = \mathcal{M} - 1$$

wobei die Fehleraktivität $d_k^{(\mathcal{M}-1)}$ am Ausgang von Neuron k der Schicht $\mathcal{M} - 1$ sich als Linearkombination der Fehler an der Schicht $\mathcal{M}$, d. h. der Folgeschicht (in feed-forward Richtung) der betrachteten Schicht, ergibt

$$d_k^{(\mathcal{M}-1)} = S'\left(z_k^{(\mathcal{M}-1)}\right) \sum_{i=1}^{n^{(\mathcal{M})}} d_i^{(\mathcal{M})} w_{i,k}^{(\mathcal{M})} \tag{3.13}$$

Der Fehler der Ausgabeschicht wird also auf diese Weise zurückpropagiert. Dieses Schema wiederholt sich in jeder Schicht.

Ganz allgemein gilt also für $0 < s < \mathcal{M}$

$$\boxed{\frac{\partial}{\partial w_{k,l}^{(s)}} E\left(\vec{x}, \{\vec{w}\}\right) = d_k^{(s)} y_l^{(s-1)}} \tag{3.14}$$

wobei

$$\boxed{d_k^{(s)} = S'\left(z_k^{(s)}\right) \sum_{i=1}^{n^{(s+1)}} d_i^{(s+1)} w_{i,k}^{(s+1)}} \tag{3.15}$$

Für den Lernschritt ergibt sich damit die Regel 3.7 mit 3.9.

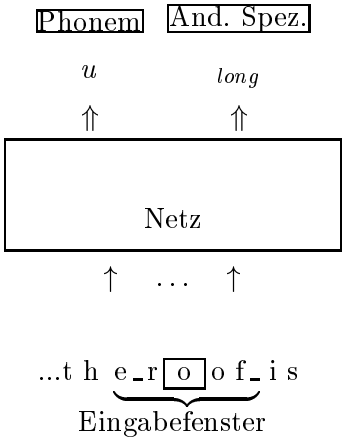
3.3 Einige Anwendungen

Im folgenden sollen einige Standardanwendungen neuronaler Netze wie NETTALK (das sprechende Netz) oder Zeichenerkennung vorgestellt werden, danach diskutieren wir einige Anwendungen mit Bottleneck–Netzen und untersuchen in 3.3.4 anhand eines Beispieles die Eignung von Feed–Forward–Netzen zur Modellierung kognitiver Prozesse. Weitere wichtige Anwendungen, vor allem für die Zeitreihenvorhersage, finden sich in späteren Kapiteln.

3.3.1 NETTALK

Ein für die Entwicklung der neuronalen Netze wichtige Anwendung war NETTALK, das unmittelbar nach der Erfindung des backpropagation Algorithmus von Sejnowski und Rosenberg (1987) entwickelt wurde. Hier handelt es sich um "das sprechende Netz", also um ein neuronales Netz, das lernt, geschriebenen Text in Lautumschrift zu übersetzen, die einem Sprachsynthesizer zugeführt werden kann. Der Zeichensatz bestand aus den 26 Buchstaben des englischen Alphabets und 3 Satzzeichen. Über den Text wurde ein Fenster geschoben, das sieben Zeichen umfaßt. Aufgabe des Netzes war es, unter Ausnutzung des Kontextes der Umgebungsbuchstaben im Fenster, dem mittleren Zeichen im Fenster das richtige Phonem und weitere Aussprachespezifikationen zuzuordnen. Die Zeichen wurden in einer 1 aus n Kodierung (unäre Kodierung) dargestellt, das Netz "sah" folglich $7 \times 29 = 203$ Inputunits. Für die Darstellung der Outputinformation dienten im gleichen Stil 26 Neuronen. Die Zwischenschicht bestand

aus 120 Neuronen.



Es müssen in dieser Architektur (zusätzlich zu den Schwellwerten der Neuronen) also 27 480 Gewichte adaptiert werden.

Nachdem das Netz mit zufälligen Wortkombinationen aus einer Bibliothek von 2000 Wörtern hinreichend lange trainiert wurde konnte es mit einer Trefferquote von 91% die richtige Aussprache der vorgelegten Sätze produzieren. Diese Leistung vergleicht sich mit dem kommerziellen System DECTALK, für dessen Entwicklung immerhin ca. 20 Mannjahre aufgewendet wurden und das eine Trefferrate von 95% vorweisen kann. Zur richtigen Einschätzung des neuronalen Ansatzes ist aber zu sagen, daß eben die fehlenden 4% sich gerade aus der Beachtung im allgemeinen komplizierter Ausnahmeregeln ergeben, die dem neuronalen Netz einzulernen sich als außerordentlich schwierig erweist. Das Netz hat ja keinerlei Verständnis der Regeln der englischen Sprache, es muß diese allein aus den vorgelegten Beispielen inferieren. Das ist aber für die im allgemeinen seltenen Ausnahmen besonders schwierig. Eventuell ist dann auch die durch die umgebenden 6 Buchstaben gelieferte Kontextinformation nicht ausreichend, das Netz müßte eigentlich zusätzlich ein Wörterbuch erlernen, in dem die Ausnahmen abgelegt sind. Das ist aber in der gewählten Architektur nicht möglich. Es haben sich deshalb in den letzten Jahren vor allem **hybride Architekturen** durchgesetzt, die eine Kombination von Regelwissen mit neuronalen Netzen realisieren.

Hybride Architekturen

3.3.2 Zeichenerkennung

In Vorbereitung.

. . .

3.3.3 Bottleneck–Netze zur Dimensionsreduktion und adaptiven Datenkompression

Viele Problemstellungen in der Informatik laufen auf das Auffinden geeigneter dimensionsreduzierter Darstellungen gegebener Datenverteilungen hinaus. Häufigstes Beispiel ist sicher die Datenkompression aber auch das Auffinden effektiverKodierungen gehört zu diesem Problemkreis. Oft geht die Dimensionsreduktion zusätzlich mit einer Informationsreduktion einher, wie z. B. bei der Hauptkomponentenanalyse zum Auffinden der relevanten Information in einer Datenmannigfaltigkeit. In der Signalverarbeitung werden diese Verfahren auch zur Datenentrauschung verwendet.

Der Zugang mit neuronalen Netzen hat den Vorteil, daß (i) diese Eigenschaften von den Netzen erlernt werden und daß (ii) die neuronlaen Netze oft recht zwanglos nichtlineare Verallgemeinerungen der bekannten klassischen Verfahren realisieren. Ein recht eleganter Zugang ergibt sich mit den sog. "bottleneck" Netzen. Dabei wird das Netz gezwungen, die Information möglichst unverfälscht durchzulassen, d. h. der Solloutput ist gleich dem Input

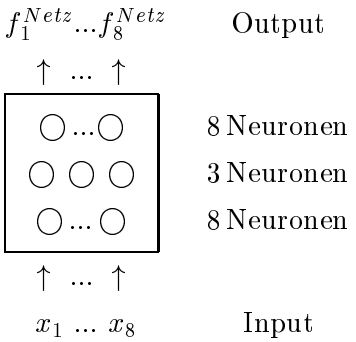
Bottle–neck Architektur

$$\vec{f}^{Netz}(\vec{x}) = \vec{y}^{soll}(\vec{x}) = \vec{x} \tag{3.16}$$

wobei aber das i. a. mehrschichtige Netz mindestens eine Zwischenschicht hat, die aus einer im Vergleich zur Zahl n der Inputunits kleinen Anzahl von Neuronen besteht. Durch diesen "Flaschenhals" muß dann die gesamte Information hindurch, was nur gut geht, wenn diese Schicht eine möglichst effektive Darstellungen für die Inputs $\vec{x} \in \Re^n$ findet.

Auffinden optimaler Kodierungen der Inputinformation

Beispiel ist die "Entdeckung" des Binärkodes durch ein Netz



(3.17)

das als Inputs und Solloutputs gemäß 3.16 die Zahlen eins bis acht in einer 1 aus n Kodierung, also z. B. $(0,0,1,0,0,0,0,0)$ für die Zahl drei, bekommt. Als Ergebnis eines erfolgreichen Trainings entsprechen die Outputs der Neuronen in der Zwischenschicht der Darstellung der Zahlen im Binärcode, also z. B. $(1,1,0)$ für den o. g. Input (bis auf Permutationen zwischen den Neuronen).

Umkodierung der Inputinformation

Dieses Prinzip kann vielfältig angewendet werden. Bei verrauschten Datenmannigfaltigkeiten repräsentieren die Zwischenneuronen die Hauptkomponenten der Daten, genauer gesagt kann man die Outputs der Neuronen in der Zwischenschicht als Darstellung der Inputs in einem i. a. nichtlinearen, krummlinigen Koordinatensystem betrachten, das aber so gewählt ist, daß es die für die gewünschte originalgetreue Rekonstruktion der Inputs in den Folgeschichten erforderliche Information möglichst vollständig enthält. Greift man also diese Zwischenoutputs ab, so hat man einen (i. a. sogar nichtlinearen) Hauptkomponentenanalysator "erlernt".

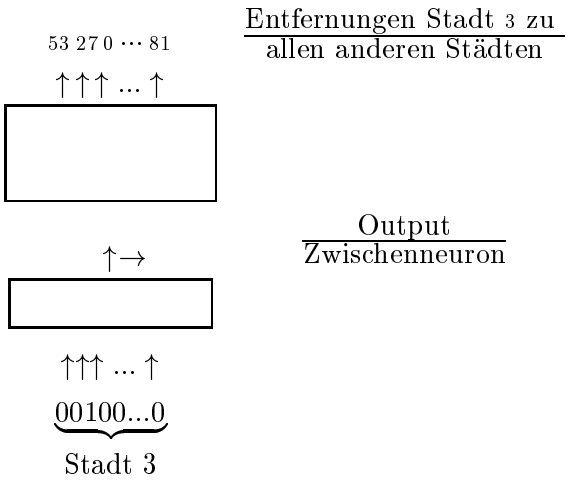
Nichtlineare Hauptkomponentenanalyse

Lösung von Optimierungsproblemen

Eine interessante Anwendung zur Lösung von Optimierungsproblemen wurde kürzlich von Wiles et. al. vorgeschlagen[44]. Die Lösung des NP–vollständigen Problems des reisenden Handelsmannes,der N Städte nacheinander in einer wegoptimierten Reihenfolge aufsuchen soll, wird von den Autoren auf die Abbildung der Landkarte mit den N Städten auf einen eindimensionalen Weg zurückgeführt. Die Autoren verwenden ein $N-1-k-N$ Bottleneck–Netzwerk. Die Inputs sind die Städteindizes, wobei Stadt i durch ein N–Bit Wort repräsentiert ist bei dem nur das i–te Bit gesetzt ist (1 aus n Kodierung), d. h. Stadt $i \quad \Leftrightarrow (0,0,...,1,...,0)$. Diese Kodierung garantiert die informationelle Gleichstellung aller Inputs. Informationen über die relative Lage der Städte ist nur in den Outputs enthalten. Dessen Komponenten sind gerade die Entfernungen

Travelling Salesman Problem

der betrachteten Stadt i zu allen anderen Städten.



Da der Flaschenhals aus einem einzigen Neuron besteht, kann die Output–Information in den nachfolgenden zwei Schichten erfolgreich nur dann aus der Aktivität $y^{(1)}$ des Zwischenneurons konstruiert werden, wenn in der internen, durch den Output des Zwischenneurons gegebenen eindimensionalen Darstellung sich die Ähnlichkeitsrelationen zwischen den Outputvektoren widerspiegeln. Das ist genau dann der Fall, wenn benachbarte Städte auch in der eindimensionalen Darstellung benachbart sind, denn die Soll—Outputs (Entfernungen zu allen anderen Städten) unterscheiden sich gerade für benachbarte Städte am wenigsten voneinander. Greift man den Output des Zwischenneurons ab, so sollte dieser der Lage der Stadt i auf einem Pfad minimaler Entfernung zwischen aufeinanderfolgenden Städte entsprechen.

Das Verfahren wurde mit $N = 10$ und $N = 30$ Städten in zufällig gewählten Lagen getestet. Dabei wurde $k = 4$ bzw. $k = 10$ als günstig gefunden. Die Ergebnisse vergleichen sich mit den Resultaten bekannter Optimierungsverfahren, die speziell für die Lösung des "travelling salesman" Problemes entwickelt wurden.

Interessant ist, daß hier das allgemeine Stetigkeitsprinzip "ähnliche Inputs produzieren ähnliche Outputs" zum Verständnis neuronaler Netze in gewisser Weise auf den Kopf gestellt wurde. Hier führen ähnliche Outputs zu ähnlichen internen Repräsentation.

3.3.4 Emergenz getrennter Verarbeitungskanäle für Objekt– und Positionsinformation im visuellen System

Eine der grundlegenden Leistungen des visulellen System besteht in seiner Fähigkeit, Objekte unabhängig von Position und Größe ihres Abbildes auf der Retina sicher identifizieren zu können. Eine genauere Untersuchung dieser durch seine Alltäglichkeit fast als selbstverständlich angenommenen Fähigkeit offenbart die hohe Komplexität der unterliegenden Verarbeitungsleistungen. Eine vom neuronalen Substrat abstrahierende, rein komputationale Theorie betrachtet Position und Identität des Objektes als formale Attribute, die in getrennten Informationskanälen oder Funktionseinheiten verarbeitet werden können[1]. Diese Vermutung scheint auch durch neuroanatomische Erkenntnisse gestützt zu sein, die allerdings nicht unumstritten sind.

Dem kann eine rein repräsentationale Theorie gegenübergestellt werden, die das gleiche Objekt in allen möglichen Positionen abgespeichert sieht. Das spart Komplexität in der Verarbeitung, allerdings, im Vergleich zu der o. g. funktionellen Segregation, um den Preis eines viel höheren Speicherbedarfs.

Um zwischen beiden Varianten besser abwägen zu können, entwarfen Kosslyn et al. in [35], [**?**] ein neuronales Netzwerkmodell für diesenAspekt der visuellen Informationsverarbeitung. Der Input in das als Feed–Forward–Netz ausgelegte Modell bestand aus einer "Retina" mit 5×5 Pixeln, auf der jeweils eines von 9 möglichen Objekten präsentiert wurde. Die Schicht der Output–Neuronen unterteilte sich in eine Gruppe von 9 Neuronen zur Repräsention der 9 verschiedenen Objekte in einer 1 aus n Kodierung plus eine Gruppe von weiteren 9 Neuronen zur Repräsentation der $3 \times 3 = 9$ möglichen Positionen des Objektes (Die Ausdehnung der Objekte schloß die Randpixel der Retina als Objektpositionen aus). Außerdem verfügte das Netz über eine verborgene Schicht, die mit 18 Neuronen so ausgestattet wurde, daß sie potentiell die Information über 'what' and 'where' im robusten 1 aus n Code repräsentieren konnte.

Das Netz wurde mit dem Backpropagation Algorithmus anhand der Muster und den Sollvorgaben für die 'what' und 'where' Outputneuronen trainiert. Im Ergebnis zeigte es sich, daß das Netz sehr wohl in der Lage war, sowohl (i)

[1] Andererseits sollten beide Verarbeitungskanäle auch kooperieren können. In der Tat mag unter einem semantischen Aspekt die Position eines Objektes in einer komplexen Szene Kontextinformation für die Identifikation des Objektes liefern. Umgekehrt kann die Position eines Objektes nur dann zweifelsfrei und eindeutig bestimmt werden, wenn das Objekt als Ganzes identifiziert worden ist.

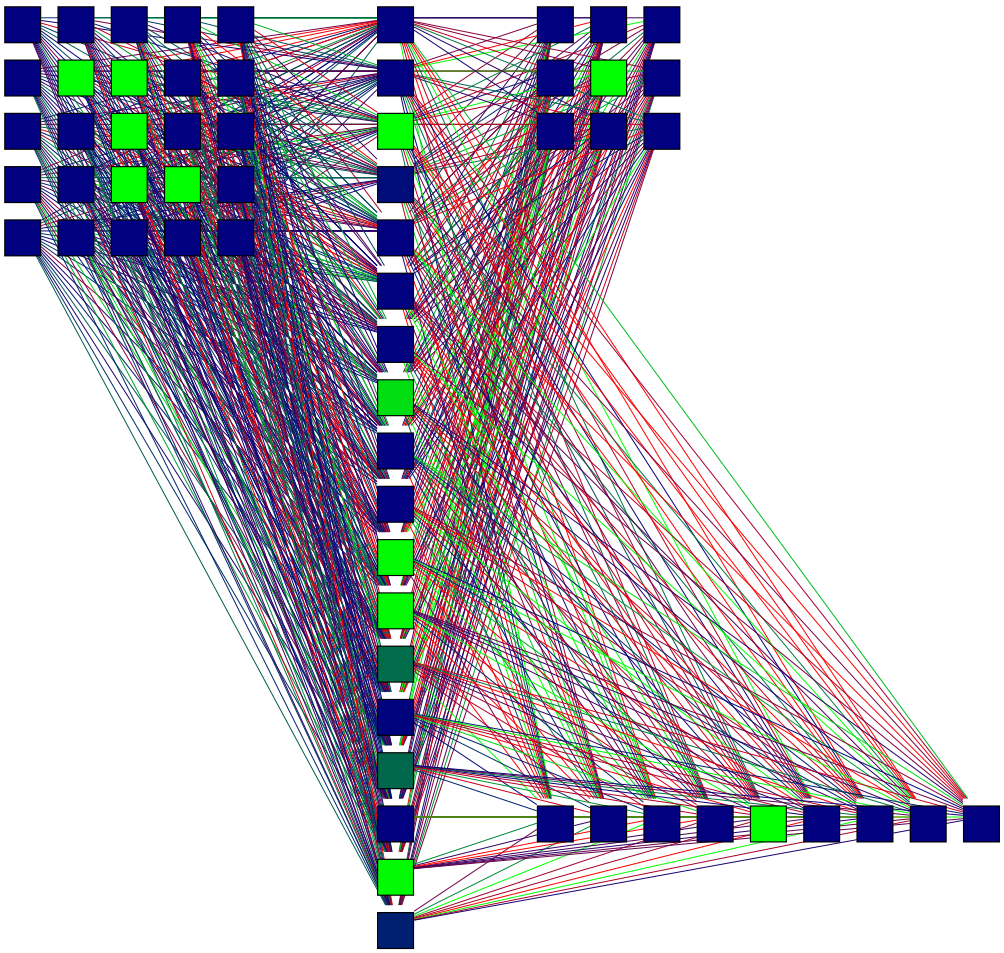


Abbildung 3.1: Das ursprünglich von Rueckl, Cave und Kosslyn verwendete Netz. Wir sehen im linken Teil die Retinaäus 5×5 Pixeln mit einem der Objekte in der Bildmitte. Die Retina speist eine Lage (hidden layer) von 18 Zwischenneuronen und diese wiederum zwei Gruppen von Output–Neuronen, oben die ẅhereünd darunter in horizontaler Anordnung die ẅhatN̈euronen. Die Farbe (Helligkeit in der s-w Darstellung) der Neuronen kodiert ihre Erregung.

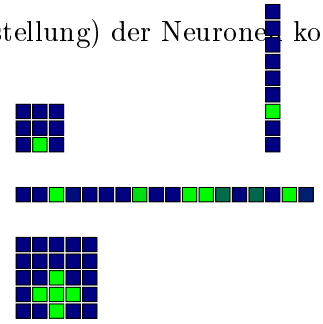


Abbildung 3.2: Das Netz von Abb. 3.1 mit einem weiteren Objekt auf der Retina. Die Darstellung wurde um 90^o gedreht.

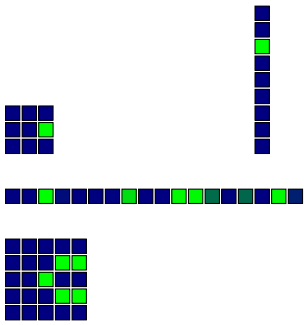


Abbildung 3.3: Wie Abbildung 3.2

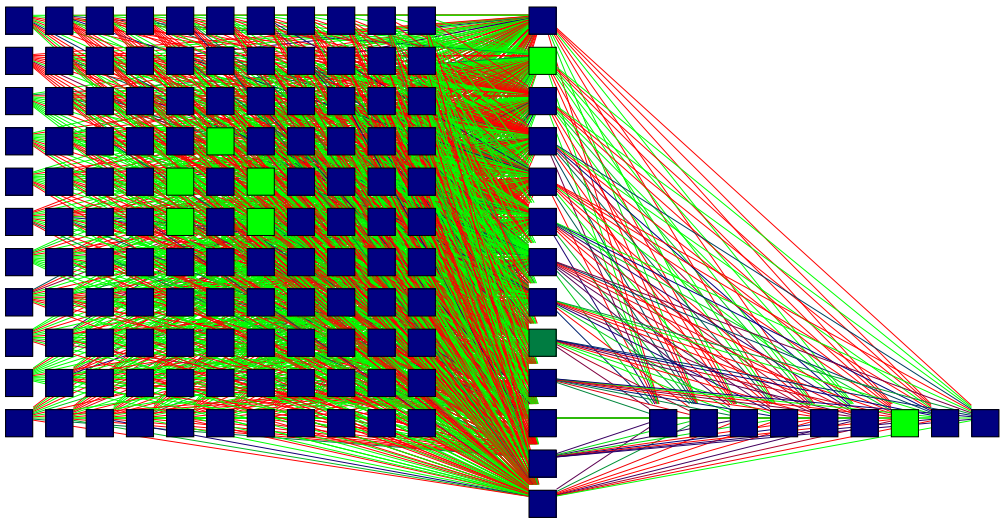


Abbildung 3.4: Wie Bild 3.1 für ein Netz mit einer Retina aus $11x11$ Pixeln. Im Gegensatz zu Bild 3.1 sind die 'where' Outputneuronen aus Platzgründen nicht mit dargestellt.

die Position als auch (ii) die Identität eines Objektes unabhängig von dessen Position zu erkennen.

Die an dem sehr einfachen 5×5 Beispiel gewonnen Aussagen mögen natürlich wenig verallgemeinerungsfähig erscheinen. Wir untersuchten deshalb ein analoges Modell [15] mit einer deutlich größeren Retina aus 11×11 Pixeln. Trotz der um einen Faktor 5 größeren Anzahl von Pixeln beobachteten wir in etwa das gleiche Verhalten. Das betraf insbesondere die Herausbildung der strukturierten rezeptiven Felder (s. u.) bei den für die Verarbeitung der 'where' Information zuständigen Zwischenneuronen, vgl. Abbildungen 3.5 und 3.6.

Interessant erscheint zunächst die Art wie das Netz das Problem gelöst hat.

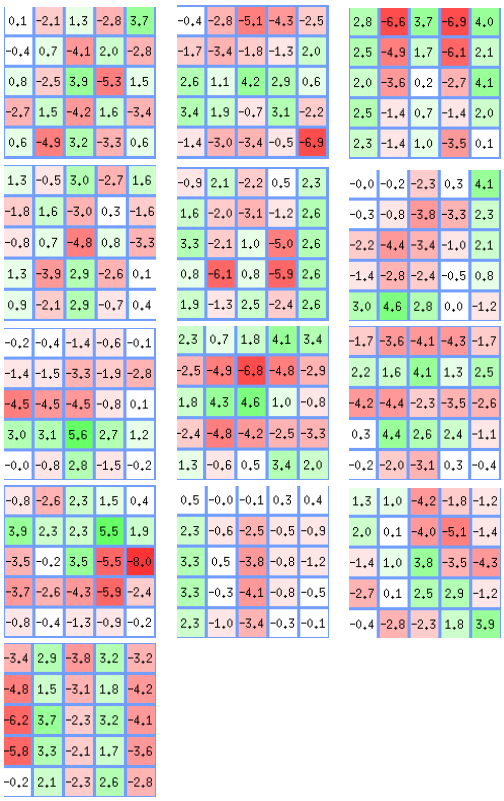


Abbildung 3.5: Die rezeptiven Felder der Zwischeneuronen. Die Abbildung zeigt die Retina, so wie sie von jedem der Zwischenneuronen "gesehen" wird, d. h. wie stark ein Pixel der Retina zur Erregung des Zwischenneurons beiträgt: Jedes Quadrat zeigt die synaptischen Kopplungsstärken von allen Pixeln der Retina zu jeweils einem der 10 Zwischenneuronen, die sich für die Verarbeitung der 'what' Information herausgebildet haben.

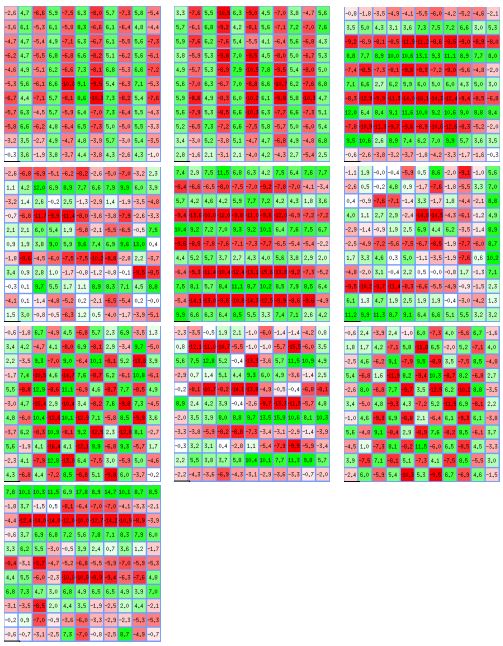


Abbildung 3.6: Wie Abbildung 3.5 aber für den Fall der 11×11 Retina. Die strukturierten rezeptiven Felder für Kanten und Schachbrettmuster sind deutlich zu erkennen.

Wie aus den Abbildungen 3.5 und 3.6 ersichtlich, entwickelten sich die Zwischenneuronen zu Feature–Detektoren, die jeweils auf ganz bestimmte Teilfeatures der Objekte sensitiv reagieren. So bildeten sich insbesondere globale (d. h. positionsinvariante) Detektoren für horizontale und vertikale Kanten heraus und sehr deutlich auch zwei Detektoren für das Vorhandensein schachbrettartiger Erregungsmuster auf der Retina. Diese Features werden durch die Zwischenneuronen einzeln detektiert, die Aufgabe der Outputneuronen ist dann nur noch, auf die zu einem bestimmten Objekt gehörige Kombination dieser Features selektiv anzusprechen. Diese Funktionalität entsprach speziell der Detektion der 'what' Information. Für das 'where' bildeten andere Zwischenneuronen rezeptive Felder aus, die sich einfach auf den Schwerpunkt der retinalen Erregung konzentrierten. Diese rezeptiven Felder können als simple, die der 'where' Zwischenneuronen als komplexe oder strukturierte charakterisiert werden.

Das Modell ist natürlich recht biologiefern. Insbesondere hat der Backpropagation–Algorithmus keine biologische Entsprechung. Die Ergebnisse können dennoch für die Entscheidung zwischen verschiedenen Paradigmen der visuellen Informationsvearbeitung herangezogen werden. Die Untersuchungen scheinen nämlich auf eine allgemeine Tendenz im Verhalten komplexer adaptiver Systeme, wie das betrachtete neuronale Netz, hinzudeuten. Der äußere Zwang, die retinalen Inputs auf getrennte Outputmodalitäten (what und where) abzubilden, manifestiert sich auch intern – in Form der ausgebildeten Feature–Detektoren – in einer Aufteilung der Verarbeitungsressourcen auf die verschiedenen Kanäle. Spekulativ könnte man weiter schlußfolgern, daß sich in der Evolution diese Trennung der Verarbeitung dann auch so herausgebildet haben sollte, wenn die getrennte Verarbeitung für die **nachgeschalteten** Verarbeitungseinheiten einen selektiven Vorteil darstellt. Das ist für Einheiten, die auf einer höheren abstrakten Ebenen arbeiten, aber durchaus anzunehmen.

.

.

3.4 Architekturen neuronaler Netze

Die Universalarchitektur $n-k-m$ eines neuronalen Netzes mit einer Zwischenschicht aus k Neuronen ist zwar bei hinreichend großem k zur Approximation einer jeden Funktion geeignet, in der Praxis ist es aber zur Beschleunigung des Lernens und zur Verbesserung der Generalsierungsleistungen des Netzes bes-

ser, auch andere Architekturen auszuprobieren. Für die praktische Arbeit hat es sich als erfoglreich erwiesen, immer mehrere Netze verschiedener Architektur gleichzeitig zu trainieren, bis sich eine Architektur als überlegen erweist.

3.4.1 XOR Netzwerke

Wie oben dargestellt, ist die Boolsche Funktion XOR nicht durch ein einzelnes Neuron realisierbar. Es gibt nach [32] insgesamt 16 Netze minimaler Komplexität, die eine Lösung des Problems darstellen. Das einfachste ist ein $2-1-2$ Netzwerk, wobei aber im Unterschied zu den bisher behandelten BP–Netzwerken direkte Verbindungen von den Inputs zum Outputneuron bestehen. Mit dem Output des einzelnen Neurons in der Zwischenschicht sieht das Outputneuron folglich drei Inputs, wir haben somit intern eine dreidimensionale Darstellung der Inputsignale. Diese kann geeignet so gewählt (gelernt) werden, daß die Klassentrennung sich auf ein lineares Problem reduziert. Das ist ein Beispiel für die "Strategie Dimensionserweiterung" zur Überführung des Problems in ein einfacher lösbares.

Strategie Dimensionserweiterung

Kürzlich konnte gezeigt werden, daß die Energielandschaft des XOR Poblems nur globale Minima hat, so daß das Gradientenabstiegsverfahren mit Sicherheit gegen die richtige Lösung konvergiert[38].

3.4.2 Erlernen Boolscher Funktionen

Bei verrauschten Daten oder zu kleinen Datensätzen droht immer das Problem der Überanpassung, das weiter unten behandelt wird. Hier soll zunächst der Fall "präziser" Daten untersucht werden, wie er insbesondere beim Erlernen einer Boolschen Funktion auftritt. Hier besteht das Problem des Overfitting nicht. Hat man ein Netz mit zu hoher Komplexität gewählt, so kann man damit rechnen, daß redundante Neuronen sich selbst "abschalten", d. h. ihre Verbindungen mit anderen Neuronen gehen im Laufe des Lernens gegen Null. Beispiel: Wird ein $2-3-1-2-...-1$ Netzwerk auf die XOR Funktion trainiert, dann wird i. a. diese schon in den ersten zwei Schichten des Netzes realisiert, die folgenden Neuronen leiten den Output des einzelnen Zwischenneurons in der zweiten verborgenen Schicht nur noch weiter, dort können also Neuronen freigesetzt werden, ohne die Funktion des Netzes zu beeinträchtigen.

Dieses Schema scheint ein wichtiges architektonisches Entwicklungsprinzip in der Frühphase der Hirnentwicklung zu sein. Die für eine bestimmte Funktion

sich im Verlaufe des Lernens herausbildenden neuronalen Netze sind in gewisser Weise minimal, d. h. sie beziehen nur die unbedingt erforderliche Anzahl von Neuronen ein. Voraussetzung ist aber die "Reinheit" des Datensatzes, wie eben beim Erlernen einer Boolschen Funktion. Die freiwerdenden Neuronen stehen für andere Aufgaben zur Verfügung und werden für diese aufgrund der allgemein vorherrschenden Wettbewerbssituation auch rekrutiert.

Ökonomie der Ressourcen

3.4.3 Selbstorganisierende Architekturen

Der Idealfall ist natürlich das Netz, das sich die optimale Architektur – ausgehend von einer minimalistischen Startarchitektur – selbst sucht. Es existieren eine Reihe von Verfahren zur Lösung dieses Problems, wir wollen hier vor allem das Kaskadenkorrelationsnetz besprechen.

Kaskadenkorrelation

Die Methode der Kaskadenkorrelation dient der automatisierten Konstruktion neuronaler Netze mit minimaler Komplexität zur Lösung einer gegebenen Aufgabenstellung. Ausgehend von einer Startarchitektur (nur Input- und Outputschicht) besteht das Grundprinzip im Hinzufügen immer weiterer Neuronen (hidden units). Diese fungieren als Repräsentanten für in der aktuellen Architektur noch nicht (oder nicht ausreichend) repräsentierte Merkmale der Inputs. Im Gegensatz zu den "klassischen" Feed-Forward-Architekturen verbinden die Synapsen auch nichtaufeinanderfolgende Schichten miteinander.

Der Algorithmus zur Konstruktion des Netzes und zum Anlernen der jeweils neu hinzukommenden hidden Units besteht aus folgenden Schritten:

1. Initialisierung als Minimalarchitektur bestehend aus der Input- und Outputschicht.
2. Training aller in die Outputunits einlaufenden Verbindungen.
3. Generierung der sog. Kandidatenunits. Das sind hidden Units, die von allen Inputunits und von allen schon existierenden hidden Units gespeist werden.
4. Training aller in die Kandidatenunits **einlaufenden** Verbindungen mit dem Ziel der Maximierung der Korrelationen zwischen dem Output der Kandidatenunit und dem **Fehler** des Netzwerkes.
5. Die beste Kandidatenunit (maximale Korrelation) wird als neue hidden Unit dem Netz hinzugefügt, indem sie als neue Inputeinheit für die Outputschicht interpretiert wird. Ihre Synapsen werden eingefroren.

6. Abbruch? Gehe zurück zu 2.

Eine ausführlichere Beschreibung findet sich in [45].

Interessant ist, daß sich XOR **nicht** mit diesem Verfahren realisieren läßt. Im Gegenteil, der Algorithmus konvergiert nicht, sondern generiert Netze mit bieliebig vielen Neuronen, ohne daß eine Lösung entsteht. Grund ist, daß schon im Schritt 2 die Synapsen des Outputneurons auf Null gelernt werden (synaptischer Kollpas), da die XOR–Funktion für das Einzelneuron nicht lernbar ist. Im Schritt 4 wiederholt sich dann das Dilemma, da das Outputneuron wegen seiner praktisch auf Null gelernten Synapsen keine sicheren Informationen über den Input zur Verfügung stellen kann und demzufolge auch die Kandidatenunits nicht wissen können, auf welche Features der Inputs sie sich konzentrieren sollen. So werden deren Synapsen ebenfalls jeweils auf Null gelernt, da sie nur einander widersprechende Fehlerinformationen erhalten.

Andererseits wurden mit den Kaskadenkorrelationsnetzen auch eine Reihe beeindruckender Ergebnisse erzielt.

3.5 Nichtlineare Regression

Wie im folgenden Abschnitt 3.5.1 dargestellt, können Feed–Forward–Netze Funktionen beliebig genau approximieren. Diese Netze stellen folglich allgemeine nichtlineare Funktionsmodelle dar. Das besondere an neuronalen Netzen ist, daß sie funktionelle Zusammenhänge aus Beipielen (Instanzen des Funktionszusammenhanges) erlernen können. Das gilt insbesondere auch, wenn diese Daten z. B. durch Meßfehler verrauscht sind. Die Inferenz funktioneller Zusammenhänge aus verrauschten Daten wird allgemein als Regression bezeichnet. Für den Fall linearer Funktionen stehen dafür eine Reihe von Standardverfahren zur Verfügung, die die Parameter der Regressionsgeraden bestimmen. Der backprop Algorithmus zur Bestimmung der Parameter des Funktionsmodells Feed–Forward–Netz, kann als nichtlineare Verallgemeinerungen der linearen Regressionsverfahren betrachtet werden.

Regression

3.5.1 Feed–Forward Netze als universelle Funktionsapproximatoren

Die FFN können als parametrisierte Funktionsmodelle angesehen werden. Für jede Wahl einer bestimmten Architektur und des Parametersatzes (gegeben

durch die Menge aller synaptischen Verbindungen, der Schwellen und des Steilheitsparameters β der Ausgabefunktion der Neuronen) ergibt sich eine bestimmte Funktion. Zu fragen ist nun nach der Mächtigkeit dieser Darstellung, d.h. ob damit der gesamte Funktionsraum überdeckt oder anders gefragt, ob auf diese Weise jede beliebige Funktion dargestellt werden kann. Die Antwort findet sich in dem Satz von Hornik, Stinchcombe und White, [24]):

Satz über Funktionsapproximation

- Schon ein Netz, gegeben durch seine Transferfunktion

$$\vec{y} = F\left(\vec{x}, \{\vec{w}\}\right) \tag{3.18}$$

mit einer einzigen verborgenen Schicht kann eine beliebige (hinreichend glatte) Funktion $\vec{y} = f\left(\vec{x}\right)$ beliebig genau approximieren, d. h.

$$\exists\, \vec{w}^{opt} : \; F\left(\vec{x}, \left\{\vec{w}^{opt}\right\}\right) \simeq f\left(\vec{x}\right) \quad \forall \vec{x}$$

bzw.

$$\text{Fehler} \quad E\left(\left\{\vec{w}^{opt}\right\}, \vec{x}\right) = \left(f\left(\vec{x}\right) - F\left(\vec{x}, \left\{\vec{w}^{opt}\right\}\right)\right)^2 \simeq 0, \quad \forall \vec{x} \tag{3.19}$$

Eine anschauliche Begründung dieser strengen Aussage findet sich in den Abbildungen **??**. Diese allgemeine Fähigkeit aber eher von rein akademischem Interesse. Typischer Fall in der Praxis: Die Menge Ω der Trainingsdaten $\vec{\phi} \in \Omega = (\Omega_{in}, \Omega_{out})$

$$\vec{\phi} = \left(\vec{x}, \vec{y}^{soll}\right), \quad \vec{y}^{soll} = f\left(\vec{x}\right), \quad \vec{x} \in \Omega_{in} \tag{3.20}$$

ist nur klein, die Daten sind zudem "verrauscht". Die Bestimmung

$$\vec{w} : \quad E\left(\{\vec{w}\}, \vec{x}\right) = \left(\vec{y}^{soll} - F\left(\vec{x}, \{\vec{w}\}\right)\right)^2 \simeq 0, \quad \forall \vec{x} \in \Omega_{in}$$

entspricht einer Überanpassung (overfitting) des Netzes an die Daten ("das Rauschen wird mit angefittet"). Ein solches Netz hat es damit nicht oder nur unvollständig geschafft, die unter dem Rauschen liegende Gesetzmäßigkeit richtig zu erfassen. Das äußert sich z. B. darin, daß ein solches Netz neu hinzukommende Daten in der Regel nur sehr schlecht repräsentieren kann.

3.5.2 Prognosegenauigkeit, Generalisierungsfähigkeit, Overfitting

Im folgenden sollen einige Verfahren zum Umgang mit "verrauschten" Daten vorgestellt werden.

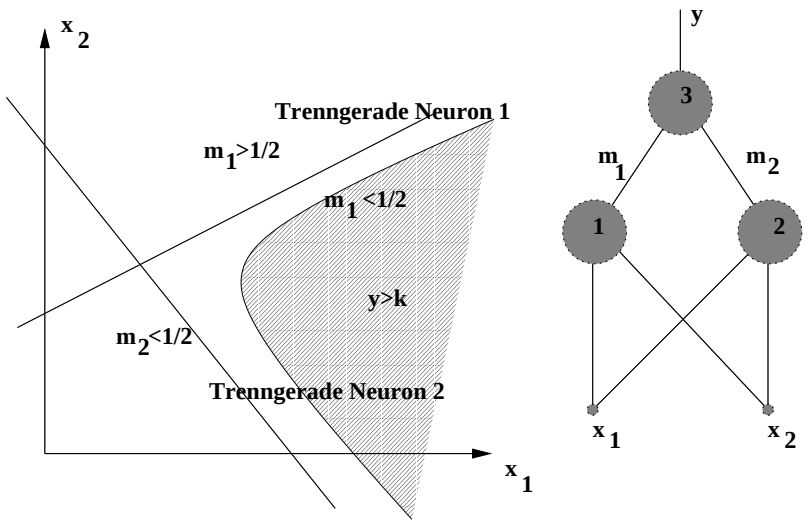


Abbildung 3.7: Illustration des Funktionsprinzips mehrschichtiger Netze. Die beiden Neuronen $(1, 2)$ der verborgenen Schicht definieren je eine Trenngerade im Raum der Inputvektoren $\vec{x} = (x_1, x_2)$. Diese sind durch die Schwelle und die synaptischen Gewichte des jeweiligen Neurons festgelegt. Im Bild sollen diese gerade so sein, daß für die Outputs der Zwischeneuronen oberhalb der jeweiligen Geraden $m_1 > 1/2, m_2 > 1/2$ gilt. Das Outputneuron (3) kann dann allen $\vec{x}$ in dem schraffiert gezeichneten Gebiet einen Wert z. B. oberhalb einer bestimmten Schwelle $k > 1/2$ zuweisen. Damit kann das Netz eine **nichtlineare** Separation der Inputs vornehmen.

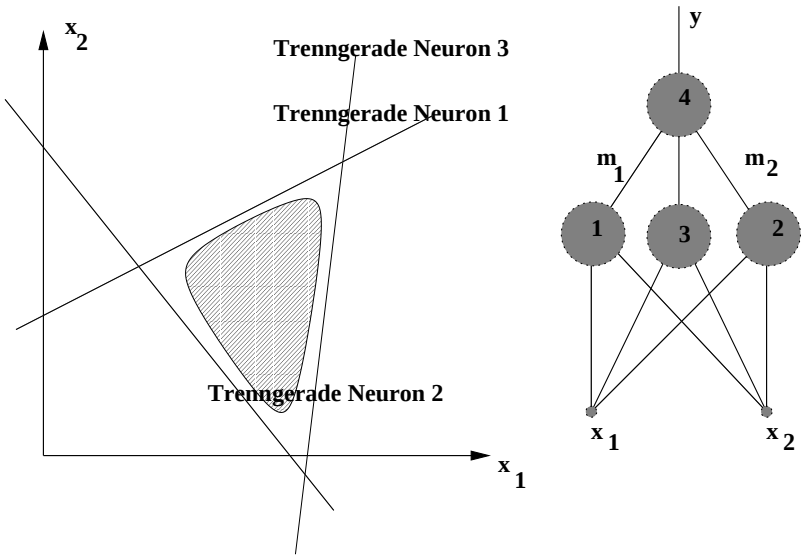


Abbildung 3.8: Gegenüber der Abb. 3.7 hat die Zwischenschicht jetzt ein Neuron mehr. Das Outputneuron kann seine Gewichte so einstellen, daß nur innerhalb des schraffierten Bereiches $y > k$ und weiter außerhalb praktisch $y = 0$ gilt. Damit wirkt dieses Netz wie ein Filter, für diese Bereiche des Inputraumes. Weitere Neuronen in der Zwischenschicht können entsprechende Filter für andere Bereiche des Inputraumes ausbilden. Das Outputneuron braucht dann nur die Filter in geeigneter Weise zu kombinieren und zu wichten, um die vorgegebenen Funktionswerte über dem gesamten Inputraum hinreichend genau zu approximieren.

Aufteilung in Trainings– und Testmenge

Trainings– und Testdaten

Liegen ausreichend viele Daten vor, so kann Ω aufgeteilt werden in einen (eigentlichen) Trainigs- und einen Testsatz, $\Omega = \left\{\Omega^{train}, \Omega^{test}\right\}$. Training mehrerer Netze $n = 1, 2, ...$ auf Ω^{train} liefert $\left\{\vec{w}^{(n)}\right\}$ durch Minimierung des Trainingsfehlers

$$\left\{\vec{w}^{(n)}\right\} \quad : \qquad E\left(\left\{\vec{w}^{(n)}\right\}\right) = \sum_{\phi \in \Omega^{train}} E\left(\left\{\vec{w}^{(n)}\right\}, \vec{x}\right) \tag{3.21}$$

$$\tag{3.22}$$

$$= \sum_{\phi \in \Omega^{train}} \left(\vec{y}^{soll} - F\left(\vec{x}, \left\{\vec{w}^{(n)}\right\}\right)\right)^2 \Rightarrow \text{Min} \tag{3.23}$$

Dann Bestimmung des mittleren quadratischen Prognosefehlers auf Ω^{test}

$$V\left(\left\{\vec{w}^{(n)}\right\}\right) = \sum_{\phi \in \Omega^{test}} E\left(\left\{\vec{w}^{(n)}\right\}, \vec{x}\right) = \sum_{\phi \in \Omega^{test}} \left(\vec{y}^{soll} - F\left(\vec{x}, \left\{\vec{w}^{(n)}\right\}\right)\right)^2 \tag{3.24}$$

Auswahl des besten Netzes das V (oder eine geeignete Kombination von E und V) minimiert. Der mittlere quadratische Prognosefehler ist gleichzeitig ein Maß für die **Verallgemeinerungsfähigkeit des Netzes**.

Sind nur wenige Daten vorhanden, so kommen folgende Verfahren zur Anwendung:

Kreuzvalidierung (cross validation)

Wähle $\forall \phi_i, \quad i = 1, 2, ..., N^\Omega, \quad \phi_i \in \Omega$ den reduzierten Datensatz $\Omega^{(i)} = \Omega \backslash \phi_i$ (d. h. $\Omega^{(i)}$ ist die Menge aller Trainingsbeispiele ohne ϕ_i), bestimme $\vec{w}^{(i)}$

$$\vec{w}^{(i)} \quad : \quad E\left(\left\{\vec{w}^{(i)}\right\}\right) = \sum_{\phi \in \Omega^{(i)}} E\left(\left\{\vec{w}^{(i)}\right\}, \vec{x}\right) \Rightarrow \text{Min}$$

Kreuzvalidierung womit der mittlere quadratische Prognosefehler zu

$$V\left(\{\vec{w}\}\right) = \frac{1}{N^\Omega} \sum_{i=1}^{N^\Omega} \sum_{\phi} E\left(\left\{\vec{w}^{(i)}\right\}, \vec{x}\right)$$

abgeschätzt werden kann. Dieser dient dann der Auswahl des besten Netzes wie oben. Standardverfahren zur Auswahl des besten Netzes.

Der Bootstrap Algorithmus

Relativ neues Verfahren in der Neuroinformatik, s. [17]Statt den Datensatz um jeweils ein Trainigsbeipiel zu reduzieren werden hier aus dem Trainigssatz

Ω neue Datensätze $\Omega^{(i)}$ durch Stichprobenziehung (mit Zurücklegen) gebildet. Der Algorithmus besteht aus den Schritten:

1. Aus Ω werden N^{Ω} Datenvektoren ϕ mit Zurücklegen gezogen. Die resultierende Stichprobe $\Omega^{(b)}$ enthält einige Paare mehrfach und andere (im Mittel 36%) überhaupt nicht.

2. Das Netzwerk wird $\forall b = 1, 2, ..., N^{(b)}$ mit dem Trainingssatz $\Omega^{(b)}$ trainiert. Die so erhaltenen $\left\{\vec{w}^{(b)}\right\}$ bilden die Bootstrap-Verteilung, die als eine Menge unterschiedlicher Schätzwerte für die Parameter $\vec{w}$ betrachtet werden kann und die die Variationsbreite der möglichen Parameter überstreicht.

3. Die Güte der Approximation durch das Netz n kann dann durch die Varianz v der Ausgaben $y = F\left(\vec{x}, \left\{\vec{w}^{(b)}\right\}\right)$ (genauer der entsprechenden Prognoseverteilungen) bezüglich der $\vec{w}^{(b)}$ bestimmt werden, d. h. berechne

$$v = \left\langle \left\langle \left(y - \langle y \rangle^{(b)} \right)^2 \right\rangle^{(b)} \right\rangle^{(\Omega)}$$

wobei $\langle ... \rangle^{(b)}$ die Mittelung über die Ausgaben bezüglich der verschiedenen Gewichtsvektoren und $\langle ... \rangle^{(\Omega)}$ die über den Datensatz bedeutet. Die Varianz bestimmt dann wieder die Auswahl des besten Netzes. (Die Idee dahinter ist natürlich, daß für das ideale Netz und nichtverrauschte Daten alle $\vec{w}^{(b)}$ übereinstimmen und damit die Varianz der Ausgabewerte gleich Null wird).

3.5.3 Verfahren zur Verbesserung der Generalisierungsfähigkeit

Obige Verfahren erlauben unter zwei oder mehreren Netzen die geeignetsten auszuwählen. Wegen des Aufwandes beim Training kann man aber nicht beliebig viele und beliebig komplexe Netze betrachten. Für Konstruktion und Training möglichst effektiver Netze stehen eine Vielzahl von Verfahren zur Verfügung, z. B.

Weight decay

Überanpassung ist oft mit dem "Weglaufen" der Parameter (synaptische Verbindungsstärken w_{ij}) nach sehr hohen Werten verbunden. Vermeidung durch Hinzufügen eines Straftermes zur Kostenfunktion, der hohe w Werte bestraft: Weight decay

$$E\left(\{\vec{w}\}, \vec{x}\right) \Rightarrow E\left(\{\vec{w}\}, \vec{x}\right) + \lambda \sum_{i,j} w_{i,j}^2 \tag{3.25}$$

Pruning

Pruning zur Verbesserung der Generalisierungsfähigkeit

Beim Prunig werden nach dem Training synaptische Verbindungen zufällig mit einer bestimmten Wahrscheinlichkeitsverteilung gekappt. Vornehmlich solche mit niedrigen Werten für die Verbindungsstärken $w_{i,j}$ bzw. mit starken Fluktuationen beim Lernen.

3.6 Eigenschaften und Varianten des Backprop-Verfahrens

Die Fehlerlandschaft ist im allgemeinen sehr komplex und durch eine Vielzahl lokaler Minima sowie durch stark unterschiedliche Anstiege der Täler gekennzeichnet. Das beeinflußt die Geschwindigkeit und Sicherheit der Konvergenz in erheblichem Maße. Es wurden deshalb eine Vielzahl von Verfahren zur Überwindung dieser Probleme entwickelt, von denen wir im folgenden die wichtigsten besprechen wollen. Zunächst soll aber eine interessante Eigenschaft des Algorithmus dargestellt werden.

3.6.1 Die Austauschbarkeit von Steilheit und Lernrate

In [39] wurde gezeigt, daß die Veränderung des Steilheitsparameters β der Ausgabefunktion durch eine Änderung der Lernrate und der synaptischen Gewichte kompensiert werden kann. Da die Gewichte gelernt werden und die Lernrate ohnehin extern kontrolliert (z. B. nach einer vorgegebenen Strategie eingefroren) wird, kann β auf seinem einmal eingestellten Wert festgehalten werden. Der BP Algorithmus hat damit **einen einzustellenden Parameter weniger**.

Das ist keine Überraschung. Der Steilheitsparameters β multipiziert in der Transferfunktion

$$f(\vec{x}) = \frac{1}{1 + e^{-\beta \vec{w} \vec{x}}}$$

direkt den Vektor der synaptischen Verbindungsstärken, eine Reskalierung $\beta\vec{w} \rightarrow \vec{w}$ eliminiert β aus der Transferfunktion. Allerdings ist die Lerngeschwindigkeit für die reskalierten w_{ij} anders: In der Lernregel 3.7 tritt β (wegen der für die sigmoide Ausgabefunktion $S = 1/\left(1 + e^{-\beta z}\right)$ gültigen Regel $S'(z) = \beta S(1 - S)$) als multiplizierender Faktor auf. Durch die Reskalierung $\epsilon\beta \rightarrow \epsilon$ kann aber der Einfluß von β vollständig aus der Lernregel eliminiert werden, vgl. 3.8, 3.9.

Steilheit und Lernrate

3.6.2 Variable Schrittlänge

Die Lerngeschwindigkeit wird entscheidend von der Lernrate ϵ bestimmt, die die Schrittlänge im Gradientenverfahren

$$\Delta\vec{w} = -\epsilon \frac{\partial}{\partial\vec{w}} E\left(\{\vec{w}\}, \vec{x}\right) \tag{3.26}$$

festlegt. Wegen des Faktors $S'(z)$ in der backprop-Lernregel kann bei gegebenem ϵ die Schrittweite sehr klein ausfallen. Idee: ϵ in flachen Gebieten der Kostenlandschaft gößer wählen. Ausführung: Bestimme Gradient, damit gemäß (3.26) $\Delta\vec{w}$ und $\Delta E = E\left(\{\vec{w}'\}, \vec{x}\right) - E\left(\{\vec{w}\}, \vec{x}\right)$, wobei $\vec{w}' = \vec{w} + \Delta\vec{w}$. Falls $\Delta E < 0$: Bestimme **mit dem gleichen** $\Delta\vec{w}$ ein $\vec{w}'' = \vec{w}' + \Delta\vec{w}$ und damit $\Delta E = E\left(\{\vec{w}''\}, \vec{x}\right) - E\left(\{\vec{w}'\}, \vec{x}\right)$. (*gradient-reuse* Verfahren), usw. solange $\Delta E < 0$. Vergrößerung von ϵ falls das mehrfach erfolgreich ist, Verkleinerung bei Vorzeichenumkehr. Gute Ergebnisse bei kleinen Lernproblemen (Faktor 5). *gradient-reuse*

3.6.3 Backpropagation mit Massenterm.

Die Gradientenregel kann für kleine ϵ in eine Differentialgleichung umgeschrieben werden

$$\frac{d}{dt}\vec{w} = -\epsilon \frac{\partial}{\partial\vec{w}} E\left(\{\vec{w}\}, \vec{x}\right) = -\epsilon\Phi\left(\{\vec{w}\}, \vec{x}\right)$$

die die mechanische Bewegung eines masselosen Teilchens mit Ortsvektor $\vec{w}$ in einem viskosen Medium unter Einfluß der Kraft $\Phi\left(\{\vec{w}\}, \vec{x}\right) = \frac{\partial}{\partial\vec{w}} E\left(\{\vec{w}\}, \vec{x}\right)$ beschreibt. Falls das Teilchen eine Masse haben soll ist ein Term in $\frac{d^2}{dt^2}\vec{w}$ hinzuzufügen, das entspricht der Ersetzung von (3.26) durch

$$\Delta\vec{w}(t) = -\epsilon \frac{\partial}{\partial\vec{w}} E\left(\{\vec{w}(t)\}, \vec{x}\right) + \gamma\Delta\vec{w}(t-1) \tag{3.27}$$

Durch den zugefügten Massenträgheits– oder Schwungterm zeigt diese Regel ein besseres Verhalten beim Überwinden lokaler Minima und auch beim Überwinden von Plateaus. Wenn der Gradient nämlich sehr klein ist, dann ist im wesentlichen $\Delta\vec{w}(t)$ nur durch seinen Vorgängerwert $\Delta\vec{w}(t-1)$ bestimmt. Ist γ nahe 1, so wird $\Delta\vec{w}$ auf dem Plateau im wesentlichen durch seinen Eingangswert in das Plateau bestimmt. Massenträgheit

3.6.4 Quickprop

Ist ein Verfahren zweiter Ordnung. Entspricht der Newton Methode zur Nullstellensuche. Idee: Bestimme eine quadratische Approximation von E um den

aktuellen Gewichtsvektor $\vec{w}_t$ und Wahl des neuen Gewichtsvektors $\vec{w}_{t+1}$ als Minimum $\nabla E = 0$ dieser quadratischen Fläche, d. h. Taylorentwicklung

$$E(\{w\}) \approx E(\{w_t\}) + (\vec{w} - \vec{w}_t)\,\nabla E(\{w_t\}) + \frac{1}{2}(\vec{w} - \vec{w}_t)\,H(\vec{w}_t)\,(\vec{w} - \vec{w}_t)^T$$

mit H Hesse-Matrix

$$H_{i,j}(\vec{w}) = \frac{\partial^2}{\partial w_i \partial w_j} E(\{\vec{w}\})$$

Damit

$$\nabla E(\{\vec{w}\}) = \nabla E(\{\vec{w}_t\}) + (\vec{w} - \vec{w}_t)\,H$$

und Wahl des neuen Gewichtsvektor (vermutetes Minimum) als

$$\vec{w}_{t+1} = \vec{w}_t - H^{-1} \nabla E(\{\vec{w}_t\}) \tag{3.28}$$

(Das Verfahren wird meist in batch-Mode (Mittelung von $E(\{\vec{w}(t)\}, \vec{x})$ über mehrere $\vec{x}$ vor Bildung des Gradienten) ausgeführt, deshalb $\vec{x}$ in E unterdrückt). Schwierig: Bestimmung von H^{-1} und Speicherung nichtlokaler Information im Netz.

Quickprop

Eigentliches Quickprop: Ebenfalls quadratische Approximation jedoch nur in Richtung des Gradienten und damit eindimensional! Für ein $w_{i,j}$ gilt

$$\Delta w_{i,j}(t) = \frac{G(t)}{G(t-1) - G(t)} \Delta w_{i,j}(t-1) \tag{3.29}$$

wobei $G(t)$ partielle Ableitung der Fehlerfunktion nach $w_{i,j}$.

Weitere Angaben in z. B. [32].

Kapitel 4

Neuronale Netze als Modelle von Zeitreihen

4.1 Allgemeines

Zeitreihen spielen in Natur und Technik eine außerordentlich große Rolle. Jede Meßreihe eines zeitlichen Vorganges ist eine Zeitreihe. Formal nehmen wir an, daß an den diskreten Zeitpunkten $t = \mu h, \quad \mu = 0, 1, 2, 3, \ldots$ ein Datenvektor $\vec{x}(t)$ (z.B. als Ergebnis einer Messung) gegeben ist. Wenn die Zeitreihe von einem natürlichen oder technischen Prozeß erzeugt wird, dann gibt es im allgemeinen einen gesetzmäßigen Zusammenhang zwischen den Daten zu verschiedenen Zeitpunkten und damit prinzipiell die Möglichkeit der Vorhersage zukünftiger Werte der Zeitreihe aus den bereits bekannten Werten (gilt leider nicht immer, s. Lottozahlen).

Konkret heißt das für die **Einschrittprognose,** einen Zusammenhang zwischen dem Vorhersagewert zur Zeit $t + h$ und einer bestimmten Menge seiner Vorgängerwerte herzustellen, d. h. einen funktionellen Zusammenhang der Form [1]

$$\vec{x}_{\mu+1} = \vec{\Phi}\left(\vec{x}_\mu, \vec{x}_{\mu-1}, \ldots, \vec{x}_{\mu-k}\right) \tag{4.1}$$

zu finden, wobei hier und im folgenden

$$\vec{x}_\mu = \vec{x}\left(\mu h\right)$$

gesetzt ist.

[1] Wir betrachten der Einfachheit halber einen stationären Zusammenhang, so daß die rechte Seite explizit nicht von der Zeit abhängt.

Die unterliegenden Gesetzmäßigkeiten sind aber im allgemeinen nicht oder nur unvollständig bekannt. Gleichzeitig sind die Messungen oder Beobachtungen selten exakt, sondern im allgemeinen durch verschiedene zufällige Störgrößen verfälscht (verrauscht).

Das Zusammenspiel von (unbekanntem) Gesetz und Zufall bildet die Herausforderung an die Vorhersage einer Zeitreihe. Neuronale Netze mit ihrer Fähigkeit, aus Beispielen zu lernen und trotz Störeinflüssen die den Daten zugrundeliegende Gesetzmäßigkeit "erahnen" zu können, haben sich deshalb hier zu ernsthaften Konkurrenten der oft aufwendigen klassischen Verfahren entwickelt. Insbesondere liefern die neuronalen Netze oft nichtlineare Verallgemeinerungen bekannter klassischer Verfahren.

4.1.1 Beispiele für Zeitreihen:

Deterministische Zeitreihen aus mathematischen Gesetzmäßigkeiten:

Jede beliebige Funktion der Zeit $\vec{x}(t)$, genommen an den diskreten Zeitpunkten $t = \mu h, \quad \mu = 0, 1, 2, \ldots$,kann als eine Zeitreihe betrachtet werden. Zum Beispiel bilden die $x_\mu, \;\; x_\mu \in \Re^1$ aus

$$x_\mu = \alpha \sin(\omega \mu h - \beta) \tag{4.2}$$

eine Zeitreihe.

Verrauschung: Um den Einfluß von zufälligen Störungen bei der Messung zu studieren, führen wir eine Verrauschung der deterministischen Zeitreihe durch, d. h. wir ersetzen

$$x_\mu \to x_\mu + \gamma z$$

wobei z eine standardverteilte Zufallszahl $z \in [-1, 1]$ ist.

Chaotische Zeitreihen:

Viele Prozesse in der Natur sind durch das Phänomen des deterministischen Chaos geprägt. Diese Prozesse produzieren sog. chaotische Zeitreihen.Mathematische Modelle chaotischer Zeitreihen können aus gewissen iterierten Abbildungen, wie z. B. der *logistischen Abbildung,* gewonnen werden. Die Zeitreihe ergibt sich dabei aus der wiederholten Iteration der Abbildung vergangener Werte des Datenvektors $\vec{x}$ auf den aktuellen Zeitpunkt. Verschiedene Zeitreihen ergeben sich aus der Iteration verschiedener Abbildungen. Beispiele:

Logistische Abbildung: Ist die Abbildung die logistische, dann lautet die Bildungsvorschrift für die Zeitreihe

$$x_{\mu+1} = a\, x_\mu \,(1 - x_\mu)$$

mit dem Kontrollparameter a, $0 \leq a \leq 4$ und $x_0 \in [-1, 1]$ womit auch $x_\mu \in [-1, 1] \quad \forall \mu$. Chaos tritt auf, sobald a seinen kritischen Wert $a_c \approx 3.8$ überschreitet.

Mackey-Glass Abbildung: Eine weitere chaotische Zeitreihe mit einer Verzögerungszeit $\tau > 1$ ist die sog. Mackey-Glass Reihe, die sich aus der folgenden, die Leukozytenbildung modellierenden Delay–Differentialgleichung ergibt,

$$\dot{x}(t) = \frac{ax\,(t-\tau)}{1 - x\,(t-\tau)^c} - bx\,(t) \tag{4.3}$$

mit der Verzögerungszeit τ und den frei wählbaren Parametern a, b, c. Ein "gängiger" Parametersatz ist $a = 0,2$; $b = 0,1$; $c = 10$ und $\tau = 17$ oder $\tau = 30$.

Henon-Abbildung: Die Henon-Abbildung ist ein Spezialfall einer allgemeineren Abbildung, die zur Beschreibung eines periodisch angestoßenen Rotators dient. Geschrieben in zwei Variablen lautet die Abbildung

$$x_{\mu+1} = 1 - ax_\mu^2 + y_\mu$$

$$y_{\mu+1} = bx_\mu,$$

was offensichtlich äquivalent ist zu

$$x_{\mu+1} = 1 - ax_\mu^2 + b\, x_{\mu-1}.$$

"Gängige" Parameter: $a = 1,4$ und $b = 0,3$.

Natürliche Zeitreihen:

Im Rahmen des Praktikums "Angewandte Neuroinformatik" kann auch auf eine Reihe von "Real-World" Daten zurückgegriffen werde, die von Prozessen in Natur, der Ökonomie oder der Technik generiert wurden. Im Unterschied zu obigen sind nun die Bildungsgesetze der Zeitreihen weitgehend unbekannt. Hier können die spezifischen Eigenschaften neuronaler Netze zur Erfassung und Nachbildung verborgener Gesetzmäßigkeiten besonders zur Wirkung gelangen. Es stehen folgende Zeitreihen zur Verfügung:

Sonnenfleckentätigkeit: Zahl und Größe der Sonnenflecken (Gebiete auf der Sonnenoberfläche, in denen die Temperatur um ca. 1000 Grad reduziert ist) sind starken Schwankungen unterworfen, wobei wir eine 11 bzw. 12 jährige und eine ca. 100-jährige Periodizität beobachten. Die für die Bildung der Sonnenflecken verantwortlichen Prozesse sind nur unvollkommen verstanden, so daß eine sichere Prognose nicht möglich ist.

Die mittlere jährliche Sonnenfleckentätigkeit wird seit 1700 quantitativ registriert. Diese Daten liegen in einem File vor.

Volkswirtschaftliche Daten: Besonders wichtig und interessant sind die monatlich ermittelten Daten für eine Reihe ökonomischer Kenngrößen, die die wirtschaftliche Entwicklung eines Landes widerspiegeln. Für das Praktikum stehen uns hier, beginnend mit Januar 1953, die Werte für den Diskont- und Lombardsatz und ab Januar 1980 für eine Reihe weiterer Größen zur Verfügung. Diese beinhalten die Preisentwicklung (Erzeuger-, Verbraucher- und Einfuhrpreise), die Auftragslage der Wirtschaft, Kreditvergabe, Staatsverschuldung, Kapitalbildung u. a., die ein ideales Testfeld für die prognostische Leistungsfähigkeit neuronaler Netze darstellen. Die Daten liegen im EXCEL-Format vor (Beschreibung des Dateiformats vorhanden) und sind für die Anwendung geeignet zu konvertieren.

Medizinische Daten Umfangreiche EEG–Daten liegen aus Untersuchungen vor, die am Max–Planck–Institut für Neuropsychologie Leipzig gewonnen wurden. Diese Daten wurden schon einer Waveletanalyse unterworfen, s. [16].

4.2 Prognose von Zeitreihen mit neuronalen Netzen

4.2.1 Ein *ad hoc* Ansatz

Ist das Bildungsgesetz der Zeitreihe bekannt, so ist eine Prognose der zukünftigen Werte von $\vec{x}\,(t)$ aus einer bestimmten Mindestzahl k von Vorgängerwerten zum aktuellen t exakt möglich. Zum Beispiel gilt $k = 2$ für die o. g. Zeitreihe aus der sin-Funktion (Beweis?) bzw. der Henon-Abbildung und $k = 1$ für die logistische Abbildung.

Hier erscheint es ziemlich trivial, ein neuronales Netz zur Prognose dieser Zeitreihen zu konstruieren. Bei geeigneter Konstruktion ist hier schon ein einzelnes (Ausgabe-) Neuron in der Lage, die Zeitreihe exakt vorauszusagen.

Betrachten wir als Beispiel eine harmonische Schwingung der Frequenz ω an den Zeitpunkten $t = 0, h, 2h, 3h, \ldots$, mit h beliebig. Mit $t = \mu h, \quad x_\mu = x(t)$ gelte

$$x_\mu = \alpha \sin(\omega \mu h + \beta) \tag{4.4}$$

Aus den Additionstheoremen folgt

$$x_{\mu+1} = c\, x_\mu + d\, x_{\mu-1}, \tag{4.5}$$

wobei

$$c = 2\, \cos(\omega h), \quad d = -1.$$

Aufgrund von 4.5 kann ein einzelnes **lineares** Neuron (d. h. die Ausgabefunktion $S(z) = z$) mit zwei Inputeinheiten, d. h. mit der Transferfunktion

$$y = F(\vec{x}, \{\vec{w}\}) = \sum_{i=1,2} w_i\, X_i \tag{4.6}$$

schon die Zeitreihe der x_μ exakt voraussagen. In der Tat, mit der Wahl $w_1 = c$ und $w_2 = d$ für die synaptischen Gewichte gibt das Neuron gerade $x_{\mu+1}$ aus, wenn inputseitig die Vorgängerwerte anliegen, d. h. $X_1 = x_\mu$ und $X_2 = x_{\mu-1}$. Das Neuron führt damit eine Einschrittprognose aus. Durch Iteration kann das Neuron dann auch die gegebene Zeitreihe 4.4 für alle Zeiten exakt voraussagen.

Die Einstellung der synaptischen Gewichte w_i kann durch einen Lernalgorithmus anhand von Trainingsbeispielen aus der Zeitreihe erfolgen. Diese bestehen aus Input-Output Paaren der Form

$$y_{\mu\ soll} = x_{\mu+1}, \quad X_{\mu\,1} = x_\mu, \;\; X_{\mu\,2} = x_{\mu-1}$$

mit zufällig gewähltem μ. Als Lernalgorithmus eignet sich der Backpropagation–Algorithmus (s. Kap. 3).

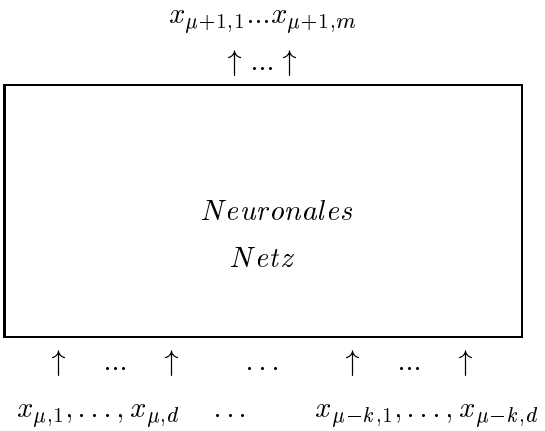
Das gewählte Beispiel zeichnet sich durch den linearen Zusammenhang zwischen den Werten der Zeitreihe zu verschiedenen Zeitpunkten aus. Eine Übertragung des Zuganges etwa auf die Henon-Abbildung erfordert wegen des quadratischen Termes die Einbeziehung von Synapsen zweiter Ordnung, d. h. einen Ansatz für die Transferfunktion der Art

$$f(X) = \sum w_i X_i + \sum_{i,j} w^{(2)}_{i,j} X_i X_j. \tag{4.7}$$

4.2.2 Feed-Forward-Netze als universelle Modelle von Zeitreihen

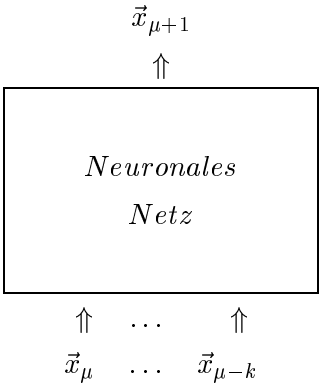
Bei unbekannten oder komplexeren Bildungsgesetzen ist dieser direkte Zugang weniger geeignet (Mehrdeutigkeiten, Instabilitäten, lange Trainingszeiten, keine Standardhardware). Universell einsetzbare Lösungsansätze nutzen aus, daß neuronale Netze als universelle Funktionsapproximatoren dienen können, d. h. daß ausreichend komplex strukturierte neuronale Netze in der Lage sind, aus einer hinreichenden Zahl von Beispielen jede beliebige Abbildung der Form 4.1 zu inferieren, s. Kap. 3.

Dieser gedankliche Ansatz kann in folgender Weise formalisiert werden: Für die Zeitreihe $\vec{x}_\mu = (x_{\mu,1}, \ldots, x_{\mu,d})\,,\ \vec{x} \in \Re^d,\ \ \mu = 0, 1, 2, \ldots$ konstruieren wir ein Feed-Forward-Netz mit $D = k \cdot d$ Input- und d Output-Einheiten.



(4.8)

oder kompakter



(4.9)

Das neuronale Netz ist durch seine Transferfunktion

$$\vec{y} = F\left(\vec{X}, \{w\}\right) \tag{4.10}$$

gegeben, wobei $\vec{X}$ =Inputvektor und $\{w\}$ = Menge der synaptischen Gewichte.Als Inputvektor $\vec{X}$ dient der $D = kd$-dimensionale Vektor, der aus k Vorgängerwerten des vorherzusagenden $\vec{x}_{\mu+1}$ gebildet wird (*time-delay* Koordinaten)[2]:

$$\vec{X}_\mu = (\vec{x}_\mu, \vec{x}_{\mu-1}, \ldots, \vec{x}_{\mu-k}) = (x_{\mu,1}, \ldots, x_{\mu-k,d}) \tag{4.11}$$

Als Trainingsbeispiele wählen wir für hinreichend viele μ Werte die Input-Output-Paare $\mathcal{Z}_\mu$

$$\mathcal{Z}_\mu = \left[\vec{y}_{\mu\, soll}, \vec{X}_\mu\right] = \left[\vec{x}_{\mu+1}, \vec{X}_\mu\right] \tag{4.12}$$

Der Output y des neuronalen Netzes 4.10 liefert den Schätzwert

$$\boxed{\tilde{x}_{\mu+1} = F\left(\vec{X}_\mu, \{w\}\right)} \tag{4.13}$$

Aufgabe des Backprop-Lernalgorithmus ist es dann, die Gewichte so einzustellen, daß

$$\vec{y}_\mu \Rightarrow \vec{y}_{\mu\ soll} \ \text{, d. h. } \tilde{x}_{\mu+1} \Rightarrow x_{\mu+1} \quad \forall\mu$$

so daß im ideal trainierten Netz gilt

$$\vec{\Phi}\left(\vec{X}_\mu\right) = F\left(\vec{X}_\mu, \{w\}\right) \quad \forall\mu$$

Zu beachten ist dabei, daß in den Fällen mit bekanntem Bildungsgesetz der Zeitreihe die Zahl k der benötigten Vorgängerwerte für eine sichere Vorhersage der Zeitreihe bekannt ist. Die Bestimmung dieser Zahl bildet jedoch bei den natürlichen Zeitreihen eine oft außerordentlich schwierige Aufgabe, deren effektive Lösung die Leistungen des Netzes in entscheidender Weise beeinflußt. Relevante Punkte sind dabei:

1. Der Einfluß des Rauschens läßt sich durch Mitnahme einer, im Vergleich zum deterministischen Bildungsgesetz, größeren Zahl von Vorgängerzeitpunkten im Inputvektor eliminieren.

2. Einfluß auf die Prognoseleistungen des Netzwerkes hat nicht nur die Zahl k der Vorgängerzeitpunkte, sondern auch die gewählte Schrittweite und damit die Länge des Intervalls in dem diese liegen (Zeithorizont). Eine elegante Lösung zur Rauschreduktion und flexiblen Schrittweitenwahl besteht in der Glättung der Inputs vor der Einspeisung in das neuronale

[2]Im allgemeinen können die Vorgängerwerte auch zu beliebigen Zeitpunkten in der Vergangenheit gewählt werden.

Netz. Dazu sind gleitende Mittelungen besonders geeignet, d. h. wir bestimmen in jedem Zeitpunkt μ einen Mittelwert $\bar{x}_\mu$ über ein Zeitintervall in der unmittelbaren Vergangenheit, wobei $\bar{x}_\mu$ gemäß

$$\bar{x}_\mu = \bar{x}_{\mu-1} - \epsilon\,(x_{\mu-1} - \bar{x}_{\mu-1})$$

in jedem Schritt aktualisiert wird und ϵ^{-1} den gewählten Zeithorizont in der Vergangenheit angibt (Länge des Mittelungsintervalls in Einheiten von h). Werden mehrere Vorgänger benötigt, so sind diese durch Mittelwerte $\bar{x}^\epsilon_\mu, \bar{x}^\eta_\mu, ...$ zu verschiedenen Zeithorizonten $\epsilon, \eta, ...$ gegeben, wobei $\vec{X}_\mu = \left(\bar{x}^\epsilon_\mu, \bar{x}^\eta_\mu, ...\right)$.

Bei der freien Iteration des Netzwerkes wird ein Startvektor $\vec{X}_\mu$ aus bekannten Werten der Zeitreihe als Input in das Netzwerk eingefüttert und der entsprechende Output $\vec{y}_\mu$ als neuer Wert $\vec{x}_{\mu+1}$ des Datenvektors interpretiert. Damit wird ein neuer Inputvektor generiert, indem durch eine Zeitverschiebung um eine Einheit alle Datenvektoren im Inputvektor $\vec{X}_{\mu+1}$ um einen Platz nach rechts rücken (Schieberegister). Dabei fällt der älteste Wert heraus und der jüngste Wert im Inputvektor ist jetzt $\vec{x}_{\mu+1}$, d. h.

$$\vec{X}_\mu = (\vec{x}_\mu, \vec{x}_{\mu-1}, \ldots, \vec{x}_{\mu-k}) \Rightarrow \vec{X}_{\mu+1} = (\tilde{x}_{\mu+1}, \vec{x}_\mu, \vec{x}_{\mu-1}, \ldots, \vec{x}_{\mu-k+1})$$

wobei $\vec{y}_\mu = F\left(\vec{X}_\mu, \{w\}\right)$. Diese Prozedur wird iteriert, so daß nach $k+1$ Schritten alle Komponenten des Inputvektors nur noch aus den vom Netzwerk selbst erzeugten Daten bestehen und damit kein Rückgriff auf die beobachteten Daten der Zeitreihe mehr erfolgt (*freie Iteration)*. Wenn das neuronale Netz die Zeitreihe exakt repräsentiert, kann damit die Zeitreihe für alle Zeiten aus den k Startwerten vorausgesagt werden.

Interessante Anwendungen zur Systemidentifikation und Zeitreihenvorhersage finden sich in [45, 8, 23] u. a.

4.2.3 Qualitätskriterien

Im allgemeinen wird schon die Einschrittprognose mit Fehlern behaftet sein. Als quantitatives Maß dieses Fehlers kann die mittlere quadratische Abweichung

$$E = \frac{1}{P} \sum_{\mu \in \mathcal{M}} \|\vec{y}_{\mu\,\mathrm{soll}} - \vec{y}_\mu\|^2 = \frac{1}{P} \sum_{\mu \in \mathcal{M}} \|\vec{x}_{\mu+1} - \vec{y}_\mu\|^2 \tag{4.14}$$

für P Prognoseschritte dienen, wobei $\mathcal{M}$ die Menge der μ-Werte ist, für die das Netz angelernt wurde. Allerdings tritt auch hier wieder die Gefahr des Overfitting auf, so daß weitere Maße einbezogen werden müssen, die insbesondere

die Generalisierungsfähigkeit des Netzwerkes beurteilen, s. dazu das Kap. 3 zu Feed-Forward-Netzwerken (Kreuzvalidierung, Bootstrap-Verfahren).

Für die Mehrschrittprognose sind, insbesondere bei chaotischen Zeitreihen, diese Maße nicht besonders aussagekräftig. Geeigneter ist hier eine Darstellung der Trajektorien im *Phasenraum* der Zeitreihe im Vergleich mit ihrer aus der freien Iteration gewonnenen Approximation.

4.3 Rekurrente Netze

Ein wesentlicher Nachteil der Modellierung von Zeitreihen mit feed–forward Netzen ist die Tatsache, daß die Zahl der Vorgängerzeitpunkte die Architektur (Anzahl der Inputunits) beeinflußt. Das macht die probeweise Veränderung des Zeithorizontes schwierig. Eine sehr interessante und effektive Alternative bieten hierzu die rekurrenten Netze, die ja über innere Zustände verfügen und damit in der Lage sind, intern eine Erinnerung über die vergangenen Werte der Zeitreihe aufzubauen. Diese Netze brauchen folglich keine time–delay Koordinaten als Inputs und sind deshalb für Zeitreihen mit unbekanntem Bildungsgesetz besondes gut geeignet.

Bekannte Varianten sind die Netze von Elman und Jordan, vgl. [18, 19, 45], die noch mit dem Backpropagation–Algorithmus trainierbar sind. Für das Training allgemeinerer rekurrenter Netze stehen verschiedene Verallgemeinerungen des Backpropagation zur Verfügung, wie z. B. rekurrentes Backpropagation oder *backpropagation through time*, s. [23].

4.4 Zeitreihenprognose mit selbstorganisierenden Merkmalskarten (Kohonen-Netze)

Ein alternatives Verfahren beruht auf der Verwendung der auf Kohonen zurückgehenden selbstorganisierenden Merkmalskarten [12].

4.4.1 Das Funktionsprinzip

Wie im Abschnitt "Zeitreihen" beschrieben, führen wir die sog. *time-delay* Koordinaten ein, d. h. wir definieren einen $D-$dimensionalen Vektor $\vec{X}_\mu$

$$\vec{X}_\mu = (\vec{x}_\mu, \vec{x}_{\mu-1}, \ldots, \vec{x}_{\mu-k})$$

der mit dem Vorhersagewert $\vec{x}_{\mu+1}$ zu einem $D+d-$dimensionalen Vektor

$$\vec{\mathcal{Z}}_\mu = \left(\vec{x}_{\mu+1}, \vec{X}_\mu\right) \tag{4.15}$$

zusammengeschlossen wird. Jeder Inputvektor $\vec{\mathcal{Z}}_\mu$ repräsentiert jeweils ein Trajektorienstück über $k+2$ Zeitschritte, so daß jedem Trajektorienstück damit ein Punkt im $D+d$ -dimensionalen Phasenraum entspricht. Im Unterschied zu den Prognoseverfahren mit anderen neuronalen Netzen (z. B. backpropagation) wird beim Modellieren der Zeitreihe mittels Kohonennenetzen der Vektor $\vec{\mathcal{Z}}_\mu$ nicht als Input-Output-Paar interpretiert sondern in der Lernphase insgesamt als **Input** in das Kohonennetz eingegeben.

4.4.2 Der Kohonenalgorithmus zur Zeitreihenvorhersage

Wir betrachten ein neuronales Netz, bei dem die N Neuronen im sog. Neuronenraum an Orten $\vec{r}_i,\ i = 1, \ldots, N$ sitzen. Üblich ist die Anordnung der Neuronen in einem Gitter der Dimension d_K. Jedes Neuron ist über seine synaptischen Gewichte mit den Inputunits verbunden. Wir schreiben den Vektor der synaptischen Gewichte des Neurons i hier als

$$\vec{\mathcal{W}}_i = \left(\vec{w}_i^{(1)}, \vec{w}_i^{(0)}, \ldots, \vec{w}_i^{(-k)}\right) = \left(w_{i,1}^{(1)}, \ldots, w_{i,d}^{(-k)}\right)$$

wobei $\vec{w} \in \Re^d$. Der Kohonen-Algorithmus besteht aus den Schritten

- **Bestimmung des Siegerneurons**: Bei Präsentation des Inputs $\vec{\mathcal{Z}}_\mu$ wird das Neuron Sieger, dessen synaptischer Verktor dem Neuron am nächsten liegt = best match neuron (im Sinne des Euklidischen Abstandes), d. h. Neuron s ist Sieger, falls gilt

$$\left(\vec{\mathcal{Z}}_\mu - \vec{\mathcal{W}}_s\right)^2 \leq \left(\vec{\mathcal{Z}}_\mu - \vec{\mathcal{W}}_i\right)^2 \quad \forall i \neq s \tag{4.16}$$

- **Lernschritt:** Im anschließenden Lernschritt werden die synaptischen Gewichte der Neuronen geändert,

$$\Delta\vec{\mathcal{W}}_i = \epsilon h_{i,s} \cdot \left(\vec{\mathcal{Z}}_\mu - \vec{\mathcal{W}}_i\right) \tag{4.17}$$

  mit der Nachbarschaftsfunktion

$$h_{i,s} = e^{-\frac{(\vec{r}_s - \vec{r}_i)^2}{2\sigma^2}} \tag{4.18}$$

$h_{i,s}$ mißt die Stärke mit der das Neuron i mitlernen darf, wenn Neuron s Sieger geworden ist. Diese Stärke klingt also mit dem Abstand zwischen den Neuronen mit der Halbwertsbreite σ ab.

Das Netz in der Lernphase:

Das Netz wird mit einer hinreichend großen Anzahl von Inputvektoren $\vec{\mathcal{Z}}_\mu$ (für verschiedene μ) trainiert. Hierbei wird also kein Unterschied zwischen bekannten Daten ($\vec{X}_\mu$) und den Vorhersagewerten ($\vec{x}_{\mu+1}$) gemacht. In der Trainingsphase lernen die Neuronen des Netzes ganz bestimmte Trajektorienstücke zu repräsentieren, da ja genau das Neuron s Sieger wird, dessen synaptischer Vektor $\vec{\mathcal{W}}_s$ am besten mit dem Trajektorienstück $\vec{\mathcal{Z}}_\mu$ übereinstimmt, s. Gleichung 4.16.

Das Netz in der Kannphase:

Nachdem das Netz angelernt ist, kann es zur Vorhersage der Zeitreihe verwendet werden. Das Prinzip zur Reproduktion der Zeitreihe besteht darin, passende Trajektorienstücke aneinanderzufügen. Bei der Einschrittprognose - Vorhersage von $\vec{x}_{\mu+1}$ aus $\vec{X}_\mu$ - wird dabei folgendermaßen vorgegangen:

- **Partielles Matching zur Siegerbestimmung:** Sei $\vec{X}_\mu$ zur Zeit t_μ bekannt. Suche das Neuron, dessen synaptischer Vektor

$$\vec{\mathcal{W}}_i = \left(\vec{w}_i^{(1)}, \vec{w}_i^{(0)}, \ldots, \vec{w}_i^{(-k)}\right) = \left(w_{i,1}^{(1)}, \ldots, w_{i,d}^{(-k)}\right)$$

  bezüglich der Komponenten $\vec{w}_i^{(0)}, \ldots, \vec{w}_i^{(-k)}$ am besten mit den entsprechenden Komponenten des Inputvektors, d. h. mit den Komponenten $\vec{x}_\mu, \vec{x}_{\mu-1}, \ldots, \vec{x}_{\mu-k}$ von $\vec{X}_\mu$ übereinstimmt, d. h. Neuron s ist Sieger, wenn

$$\sum_{j=0}^{k} \left(\vec{w}_s^{(-j)} - \vec{x}_{\mu+j}\right)^2 \leq \sum_{j=0}^{k} \left(\vec{w}_i^{(-j)} - \vec{x}_{\mu+j}\right)^2 \quad \forall i \neq s$$

- **Der Vektor $\vec{w}_s^{(1)}$ ist der Vorhersagewert für $\vec{x}_{\mu+1}$.**

Bemerkung: In den Kohonenalgorithmus geht in Form der Nachbarschaftsfunktion wesentlich die Topologie des Neuronenraumes ein, i. a. werden die Neuronen auf ein Gitter der Dimension d_K gesetzt. Die Topologie des Neuronenraumes muß dem Problem in gewisser Weise angepaßt werden. Dazu überlege man sich, daß die Leistung des Kohonenalgorithmus in der Herstellung einer **nachbarschaftstreuen** Abbildung des Inputraumes auf den Neuronenraum besteht. Dabei wird nicht der gesamte Inputraum, sondern im wesentlichen nur das Gebiet, in dem die Wolke der Datenpunkte liegt, abgebildet.

Die Inputvektoren $\vec{\mathcal{Z}}_\mu$ repräsentieren jeweils ein Trajektorienstück über $k+2$ Zeitschritte. Jedem solchen $\vec{\mathcal{Z}}_\mu$, d. h. jedem Trajektorienstück, entspricht damit ein Punkt im $D+d$ dimensionalen Phasenraum. Wenn die Zeitreihe etwa durch eine harmonische Funktion erzeugt wird,

$$x_\mu = \alpha \sin(\omega \mu h - \beta)$$

dann liegen diese Punkte - unabhängig vom Wert von k und damit von der Dimension des Inputraumes - auf einer **eindimensionalen** geschlossenen Mannigfaltigkeit. Dementsprechend ist es hier sinnvoll, die Neuronen im Neuronenraum auf einem Kreis anzuordnen.

Analoge Überlegungen sind für alle betrachteten Zeitreihen anzustellen.

4.5 Übungen

Aufgabe 1: Zeitreihenvorhersage mit Feed-Forward-Netzwerken

Untersuchen Sie mit dem SNNS (Stuttgarter Neuronaler Netzwerksimulator) [46] die Leistungsfähigkeit von Feed-Forward-Netzwerken für die Zeitreihenvorhersage sowohl in der Einschritt- als auch in der Mehrschrittprognose. Aufgabenstellungen sind:

- Realisieren Sie mit geeigneten Feed-Forward-Netzen ein Progosesystem sowohl für Zeitreihen mit bekannten Bildungsgesetzen (sin-Funktion, chaotische Zeitreihen) als auch für Beobachtungsdaten (Sonnenflecken, ökonomische Zeitreihen).

- Untersuchen Sie den Einfluß der richtigen Wahl der Vorgängerzeitpunkte (s. o.), insbesondere zur Rauschreduktion und den

- Einfluß der Architektur des Netzes auf Prognosegenauigkeit und Verallgemeinerungsfähigkeit des Netzwerkes.

- Varianten des Backpropagation-Algorithmus und Lerngeschwindigkeit.

- Stellen Sie die Ergebnisse geeignet dar, z. B. durch eine Lernkurve und/oder den direkten Vergleich der Zeitreihen bzw. der entsprechenden Phasenraumdarstellungen.

Aufgabe 2: Vorhersage von Zeitreihen mit Kohonennetzwerken

Alternativ (oder zusätzlich) zu **Aufgabe 1** (Vorhersage mit Feed-Forward-Netzwerken) sollen sowohl für eine künstliche (sin-Funktion) und eine natürliche Zeitreihe (Sonnenfleckentätigkeit) mit der dargestellten Methode die Zeitreihenvorhersage durchgeführt werden. Dabei sind folgende Aufgabenstellungen zu beachten:

- Untersuchung der Prognosegenauigkeit und Generalisierungsfähigkeit als Funktion der Zahl der Neuronen und der Reichweite der Nachbarschaftsfunktion.
- Stellen Sie die Ergebnisse geeignet dar (Lernkurve, direkter Vergleich der vorhergesagten mit der vorgegebenen Zeitreihe). Diskutieren Sie die Ergebnisse, speziell im Vergleich mit Feed-Forward-Netzwerken und zum unter 4.2.1 beschriebenen *ad hoc* Ansatz (z. B. für die sin-Funktion)

Kapitel 5

Partiell rekurrene Netze

In Vorbereitung.

Kapitel 6

Attraktornetzwerke

6.1 Architektur und Dynamik des Netzwerkes

6.1.1 Definition des Netzwerkes

Wir betrachten zunächst Netzwerke mit einer deterministischen Update-Regel (s.u.).

Betr. vollständig verkoppeltes Netze aus N bipolaren Modellneuronen $i = 1, \ldots, N$, d.h. Aktivitätszustand des Neurons i mit $\xi_i \in \{-1, +1\}$ codiert. Aktivitätszustand des Netzwerkes zur Zeit t durch den

$$\text{Zustandsvektor} \quad \vec{\xi}(t) = (\xi_1(t), ..., \xi_N(t)) \tag{6.1}$$

gegeben, Beispiel:

$$\text{Zustandsvektor des Netzwerkes:} \quad \vec{\xi}(t) = (-1, 1, 1, 1, \ldots, -1, 1) \tag{6.2}$$

Die Aktivität (Output) des Neurons i ist durch

$$\xi_i = sign(z_i) \tag{6.3}$$

als Funktion des postsynaptischen Potentials (PSP) z_i und damit der Inputs von allen anderen Neuronen bestimmt, wobei das PSP sich aus der gewichteten Summe der Outputs aller anderen Neuronen ergibt:

$$\text{Postsynaptisches Potential (PSP) des Neurons } i: \tag{6.4}$$

$$z_i = \sum_{j=1}^{N} w_{i,j}\xi_j - h_i = \vec{w}_i \cdot \vec{\xi} - h_i$$

mit

w_{ij} = Synaptische Kopplungsstärke zwischen Neuron i und j

wobei i. a. $w_{i,i} = 0$ (keine Selbstkoppelung). Die Schwelle h_i kann wie gehabt durch Einführung einer Zusatzkomponente (fiktives Neuron) $\xi_0 = 1$ und $w_{i,0} = -h_i$ automatisch mitgeführt werden.

Bei den betrachteten statischen Neuronen ist aber die Regel 6.3 wegen der Rückkopplungsschleifen nicht eindeutig, die outputs können sich erst nach einer minimalen Verzögerungsszeit aus den Inputs bilden. Eine eindeutige Festlegung ergibt sich durch Einführung eines

$$\text{Zeittaktes } t = 0, 1, 2, \ldots .$$

Aktualisierung (Updaten) der neuronalen Aktivitäten entweder für alle Neuronen gleichzeitig (parallel, synchron) oder in systematischer oder stochastischer Abfolge (sequentiell, asynchron). Durch das Aktualisieren kann sich der Netzwerkzustand verändern, d. h. wir erhalten einen zeitlich veränderlichen Netzwerkzustand $\vec{\xi}(t)$, der die Dynamik des Netzwerkes repräsentiert.

Algorithmus für asynchrones "Updaten":

1.	$t = 0$:	Initialisierung: Lege Ausgangszustand $\vec{\xi}(0)$ des Netzwerkes fest.
2.	$t = t + 1$:	Wähle ein $i \in \{1, 2, \ldots, N\}$ aus. Bestimme das aktuelle (effektive) PSP $z_i = \vec{w}_i \cdot \vec{\xi} - h_i$ Aktualisiere den Zustand ξ_i dieses Neurons gemäß $\xi_i^{neu} = sign\,(z_i)$
3.		Abbruch? Sonst GOTO 2

(6.5)

Bemerkung: ξ_i^{neu} ist der Zustand des Neurons zur neuen Zeit.

Die Dynamik kann auch im Sinne einer iterierten Abbildung geschrieben werden

$$\xi_i := sign\left(\vec{w}_i \cdot \vec{\xi} - h_i\right)$$

oder

$$\xi_i := sign\left(\sum_{j \neq i} w_{ij} \cdot \xi_j - h_i\right) \tag{6.6}$$

wobei beim asynchronen Updaten das Neuron i etwa zufällig ausgewählt wird. Offensichtlich bildet diese iterierte Abbildung den Zustandsraum, d. h. die Menge $\{-1, +1\}^N$ auf sich selbst ab.

6.1.2 Formale Eigenschaften

Darstellung der Dynamik im Konfigurationsraum

Die Dynamik 6.5 definiert ein zeitdiskretes dynamisches System. Darstellung des Zustandes $\vec{\xi}(t)$ im **Konfigurations- oder Zustandsraum**, d. h. nach (6.2) als Eckpunkt des N-dimensionalen Hyperwürfels. Eine (echte, d. h. $\xi_i^{neu} \neq \xi_i^{alt}$) Aktualisierung gemäß (**??**) bedeutet Übergang zu einem benachbarten Eckpunkt (ein Bit geflippt), Zeitverlauf $\vec{\xi}(t)$ des Zustandes durch Trajektorie, d. h. Abfolge von solchen Sprüngen gegeben.

Die Energiefunktion. Symmetrische Kopplungen.

Die Definition der Dynamik (6.5) gilt ganz allgemein für beliebige Kopplungen. Besonderheit bei symmetrischen Kopplungsstärken

$$w_{i,j} = w_{j.i}$$

ist die Exisenz einer Energiefunktion $H(t)$, die durch die Dynamik (6.5) minimiert wird. Betr.

$$H(t) = H\left[\vec{\xi}\right] = -\frac{1}{2}\sum_{i\neq j}\xi_i w_{i,j}\xi_j - \sum_{i=1}^{N}\xi_i h_i, \quad \text{wobei} \quad \xi_i = \xi_i(t) \tag{6.7}$$

wobei $\sum_{i\neq j}$ auf den Ausschluß der Selbstkopplungen hinweist. Unter der Dynamik (6.5) gilt

$$\frac{d}{dt}H(t) \leq 0 \tag{6.8}$$

so daß $H(t)$ monoton fallend (oder konstant) ist.

Beweis: Nehme an, Neuron k werde gerade aktualisiert, die Änderung des Netzwerkzustandes infolge der Aktualisierung ist

$$\delta\xi_k = \xi_k^{neu} - \xi_k^{alt} = sign(z_k) - \xi_k^{alt}, \quad \text{und} \quad \delta\xi_i = 0 \quad \forall i \neq k$$

Wegen $\xi_k^{neu} = sign(z_k)$ ist $\delta\xi_k$ entweder 0 oder $2\,sign(z_k)$. Für die Änderung $\Delta H = H\left[\vec{\xi}^{neu}\right] - H\left[\vec{\xi}^{alt}\right]$ von H infolge der Aktualisierung gilt folglich (verwende $w_{i,j} = w_{j,i}$)

$$\Delta H = -\sum_j \delta\xi_k w_{k,j}\xi_j - \delta\xi_k h_k = -\delta\xi_k\, z_k \leq 0 \tag{6.9}$$

da entweder $\delta\xi_k = 0$ oder $\delta\xi_k = 2\,sign(z_k)$.

Fixpunkte der Dynamik

Damit nähert sich der Zustand des Netzwerkes zumindest asymptotisch einem festen Wert an, da ja die Energie nicht beliebig klein werden kann. Der angestrebte Zustand $\vec{\xi}^{Fix}$ ist Fixpunkt der Dynamik (Endpunkt der Trajektorie), denn für ihn gilt

$$\xi_i^{Fix} = sign\left(\vec{w}_i \cdot \vec{\xi}^{Fix} - h_i\right)$$

bzw.

$$\xi_i^{Fix} = sign\left(\sum_j w_{ij} \cdot \xi_j^{Fix} - h_i\right)$$

Lage und Zahl der Fixpunkte hängen bei gegebenener Dynamik (6.5) entscheidend von den Kopplungen $\{\vec{w}_i\}$ ab (s. u.). Welcher Fixpunkt angelaufen wird bestimmt der Startpunkt $\vec{\xi}(0)$. Das macht diese Systeme zu Kandidaten für Assoziativspeicher, falls es gelingt, die Fixpunkte, die dann die Speicherzustände sein sollten, durch geeignete Wahl der Kopplungsstärken $\{w_{ij}\}$ vorzugeben (s. u.).

6.1.3 Optimierung frustrierter Systeme als Netzwerkdynamik

Das Auffinden der Fixpunkte der Dynamik entspricht der Minimierung der Energiefunktion. In diesem Sinne löst die Dynamik ein Optimierungsproblem. Bei Abbildung eines Optimierungsproblems auf eine Energiefunktion mit symmetrischen Kopplungen und mit einem binären Lösungsraum kann die obige Dynamik die Lösung liefern. Bei zufällig verteilten $w_{i,j}$ mit unterschiedlichen Vorzeichen ist das Problem des Auffindens eines optimalen $\vec{\xi}^{opt}$, das also die Energiefunktion minimiert, ein NP-vollständiges Problem (nicht in polynomialer Zeit lösbar). Grund: das System ist frustriert. Zur Diskussion betr. wir den Fall gleichstarker Koplungen $w_{i,j} = w$ mit unterschiedlichen Vorzeichen. Betr.zunächst Kopplung zwischen zwei Neuronen. Für (verwende $\Leftrightarrow$ für positive und $\rightarrow$ für negative Kopplungen, Pfeilrichtung spielt keine Rolle)

$$\oplus \Leftrightarrow \ominus \quad , \quad \text{oder} \qquad \ominus \Leftrightarrow \oplus \tag{6.10}$$

ergibt sich ein positiver Beitrag $|w|$, für

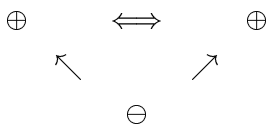
$$\oplus \Leftrightarrow \oplus \quad , \quad \text{oder} \qquad \ominus \Leftrightarrow \ominus \tag{6.11}$$

ein negativer Beitrag $-|w|$ zur Energiefunktion (6.7), d. h. bei (6.11), ist die Verbindung durch die gefundenen Konfiguration "befriedigt", weil sie zu einer

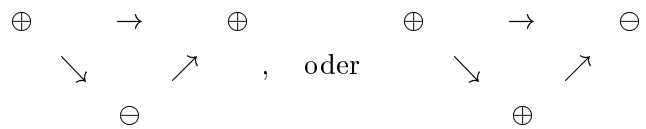
Energieverringerung führt. Bei negativer Kopplung ist die befriedigende Konfiguration entsprechend

$$\oplus \to \ominus \quad , \quad \text{oder} \qquad \ominus \to \oplus$$

Bei drei Neuronen kann auch im Fall rein positiver Kopplungen eine befriedigende Lösung gefunden werden, bei gleichsinniger Aktivität aller Neuronen ergibt sich die maximale Energieminimierung $-3\,|w|$, ebenso wie bei



Sonst ist, z. B. bei rein negativen Kopplungen, nur eine von mehreren gleichberechtigten Kompromißlösungen als suboptimale Variante realisierbar, z. B.



mit Beitrag $-\,|w|$ zur Energiefunktion. Hier bleibt immer eine Verbindung ($\oplus \;\to\; \oplus$) unbefriedigt oder "frustriert".

Bei größerer Neuronenzahl und zufällig verteilten Vorzeichen der Kopplungen steigt der Frustrationsgrad (Zahl der frustrierten Kopplungen) schnell an, so daß die optimale Konfiguration immer nur eine von vielen etwa gleichberechtigten Kompromißlösungen ist. Die Zahl dieser Kompromißlösungen steigt mit N wie $\exp(cN)$, wobei c eine problemspezifische Konstante ist, an[1] und damit ist die systematische Suche nach dem optimalen $\vec{\xi}$ nicht in polynomialer Zeit durchführbar$\Rightarrow$ NP-vollständiges Problem.

Abbildung von Optimierungsproblemen auf symmetrische Attraktornetzwerke

Ein Zweiklassenproblem Betr. eine Menge von Personen $i = 1, 2, \ldots, N$, die in zwei Klassen eingeteilt werden sollen. Beziehungen zwischen den Personen (Sympathie, Antipathie) durch Wechselwirkungsterme $w_{i,j}$ bewertet. Ziel: Einteilung der Personen in zwei Klassen mit möglichst geringer Frustration, d. h. Personen mit positiver (negativer) Wechselwirkung sind möglichst in die gleiche (unterschiedliche) Klasse(n) einzuordnen. Klassenlabel ξ_i: $\xi_i = 1\,(-1)$

[1] Es gilt $c \prec \log 2$, da die Gesamtzahl der Zustände gleich $2^N = \exp(N \log 2)$ ist.

heißt Person $i \in$Klasse 1 (2). Wenn $-\xi_i w_{i,j} \xi_j$ den Beitrag der Personen i, j zur Gesamtspannung in dem System mißt, so ist die zu minimierende objektive Funktion des Optimierungsproblems

$$H\left[\vec{\xi}\right] = -\frac{1}{2}\sum_{i \neq j} \xi_i w_{i,j} \xi_j$$

in Übereinstimmung mit (6.7). Die Dynamik (6.5) minimiert damit diese Funktion, zumindest wird ein lokales Minimum erreicht, s.u.

Das Graphenpartitionierungsproblem

Der reisende Handelsmann (travelling salesman)

6.2 Das Hopfieldnetzwerk

Wegen der Existenz von Fixpunkten mit endlichen Einzugsgebieten können Attraktornetzwerke allgemein als Assoziativspeicher verwendet werden, allerdings müssen dazu die Kopplungen entsprechend eingestellt werden, so daß die Fixpunkte mit den zu speichernden Mustern übereinstimmen. Das gelingt mit folgendem Verfahren

6.2.1 Die Hebbsche Lernregel

Eine Einstellung der Kopplungen, so daß die Fixpunkte vorgegebenen Mustern

$$\vec{\xi}^{\mu}, \quad \mu = 1, 2, \ldots, P \tag{6.12}$$

entsprechen, gelingt zumindest näherungsweise mit dem Ansatz

$$w_{i,j} = \frac{1}{N}\sum_{\mu=1}^{P} \xi_i^{\mu} \xi_j^{\mu} \tag{6.13}$$

(Hebb- bzw. Hopfield-Matrix).Die Vorschrift kann als Resultat der Anwendung folgender Lernregel betrachtet werden:

1. Initialisierung $w_{i,j} = 0, \quad \forall i, j$ (Tabula rasa)

2. For $\mu = 1$ to P do

For $i = 1$ to N do

For $j = 1$ to N do

$w_{i,j} = w_{i,j} + \frac{1}{N}\xi_i^\mu \xi_j^\mu$

Diese entspricht dem Hebbschen Paradigma des Lernens, wonach synaptische Verbindungen zwischen Neuronen im gleichen (ungleichen) Aktivitätszustand verstärkt (geschwächt) werden.

Ein vollständig verkoppeltes neuronales Netz mit der Dynamik (6.5) und den synaptischen Kopplungen (6.13) bildet das eigentliche Hopfieldmodell.

6.2.2 Eigenschaften des Hopfieldmodells

Vorbemerkung – Eigenschaften hochdimensionaler binärer Zufallsvektoren

Seien $\vec{\xi}, \vec{\eta}, \quad \xi_i, \eta_i \in \{-1, +1\}$ zwei Zufallsvektoren, d. h. ξ_i, η_i seien unabhängige, gleichverteilte Zufallszahlen mit Mittelwert Null,

$$\langle \xi_i \rangle = 0, \quad \langle \eta_i \rangle = 0, \quad \langle \xi_i \xi_j \rangle = \delta_{i,j}, \quad \langle \xi_i \eta_j \rangle = 0 \quad \forall i, j$$

und Streuung

$$\sigma = \sqrt{\langle \xi_i^2 \rangle} = 1$$

Das Skalarprodukt

$$\left(\vec{\xi}, \vec{\eta}\right) = \frac{1}{N} \sum_{i=1}^{N} \xi_i \eta_i = \frac{1}{N} \vec{\xi} \cdot \vec{\eta}$$

ist gleich 1 für $\vec{\xi} = \vec{\eta}$ (da dann $\xi_i \eta_i = 1$) bzw. für $\vec{\xi} \neq \vec{\eta}$ gleich einer Zufallszahl ζ mit $\langle \zeta \rangle = \left\langle \left(\vec{\xi}, \vec{\eta}\right) \right\rangle = 0$ und

$$\sqrt{\langle \zeta^2 \rangle} = \frac{1}{\sqrt{N}} \tag{6.14}$$

da $\zeta = \left(\vec{\xi}, \vec{\eta}\right) = \frac{1}{N} \sum_{i=1}^{N} \omega_i, \quad$ mit $\omega_i =$random$\in \{-1, +1\}$. Für $N \to \infty$ genügt $\bar{\zeta} = \sqrt{N}\zeta$ einer Gaußverteilung (zentraler Grenzwertsatz) mit Streuung 1, vgl. 6.14

$$p\left(\bar{\zeta}\right) = \frac{1}{\sqrt{2\pi}} e^{-\frac{\bar{\zeta}^2}{2}} \tag{6.15}$$

Mit (6.14) folgt, daß zwei im obigen Sinne unabhängige binäre Zufallsvektoren für $N \to \infty$ immer orthogonal zueinander sind. Für endliche N gilt der Satz bis auf Abweichungen der Ordnung $\frac{1}{\sqrt{N}}$

$$\left(\vec{\xi}, \vec{\eta}\right) = O\left(1/\sqrt{N}\right) \tag{6.16}$$

Wichtige Kenngrößen des Netzwerkes

- Die **Überlappung** (Skalarprodukt) eines gegebenen Netzwerkzustandes $\vec{\xi}$ mit einem Muster $\vec{\xi}^{\mu}$

$$m^{(\mu)}(t) = \left(\vec{\xi}(t), \vec{\xi}^{\mu}\right) \tag{6.17}$$

  mißt die Übereinstimmung des Netzwerkzustandes mit dem Muster. Zusammenhang mit Hammingdistanz:

$$D\left[\vec{\xi}(t), \vec{\xi}^{\mu}\right] = \frac{N}{2}\left(1 + m^{(\mu)}(t)\right) \tag{6.18}$$

  mißt die Zahl der Bits in denen $\vec{\xi}(t)$ und $\vec{\xi}^{(\mu)}$ übereinstimmen. Damit ist $\left(1 + m^{(\mu)}(t)\right)/2$ die relative Zahl der übereinstimmenden Bits. Die gespeicherten Zustände (Speicherinhalte) entsprechen Fixpunkten $\vec{\xi}^{Fix} = \lim_{t\to\infty} \xi(t)$, ihre Überlappung mit den Mustern sei

$$\left(\vec{\xi}^{Fix}, \vec{\xi}^{\mu}\right) = m^{(\mu)} = \lim_{t\to\infty} m^{(\mu)}(t)$$

  (Achtung: Nicht alle Fixpunkte sind Speicherinhalte, s. u.).

- Das System befindet sich in einem **idealen Speicherzustand** (retrieval state) zum Muster $\vec{\xi}^{\mu}$ falls

$$\vec{\xi}^{Fix} = \vec{\xi}^{\mu} \tag{6.19}$$

  und damit

$$m^{(\mu)} = \delta_{\mu,\mu^F}$$

  Ideale Speicherung heißt also daß die Überlappung eines Speicherinhaltes mit genau einem der Muster gleich eins und mit allen anderen Mustern gleich Null ist. Dies wird bei hoher Beladung des Netzes nicht erreicht, $m^{(\mu)}$ definiert deshalb die sog.

- **Wiedergabequalität** (retrieval quality): Ist das System in einem retrievfal statezum Muster μ, so gilt im allgemeinen, d. h. für $m^{\mu} < 1$

$$\vec{\xi}^{Retr.} = \vec{\xi}^{\mu} + \text{"cross talk"} \quad = \quad \text{Muster} + \text{"cross talk"} \tag{6.20}$$

  m^{μ} definiert dann die **Wiedergabequalität** und $D^{\mu} = (1 + m^{\mu})/2$ ist die relative Zahl der Bits, in denen die Speicherinhalte und die zu speichernden Muster übereinstimmen. (s. o.) Wie wir sehen werden ist die Zahl der speicherbaren Muster proportional zur Neuronenzahl N wir definieren deshalb den

- **Load parameter** α

$$\alpha = \frac{P}{N} \tag{6.21}$$

der die Zahl der im Netzwerk gespeicherten Muster angibt.

Speicherung eines einzigen Musters (Mattismodell)

Vorbemerkung: Rechnen mit der *sign*–Funktion

Es gilt (mit $|\xi_i| = 1$)

$$\begin{aligned} sign\,(ax) &= sign\,(a)\,sign\,(x)\,, \quad sign\,(a\xi_i) = \xi_i sign\,(a) \\ sign\,(x+a) &= sign\,(x) \quad \text{falls} \quad |x| > |a| \end{aligned} \tag{6.22}$$

Definition des Mattismodells: Das Mattismodell entspricht der Speicherung nur eines einzelnen Musters $\vec{\xi}^1$, d.h.

$$w_{i,j} = \frac{1}{N}\xi_i^1\xi_j^1, \qquad w_{ii} = 0$$

Eigenschaften: Das Muster $\vec{\xi}^1$ ist ein Fixpunkt der Dynamik, da

$$z_i = \sum_{j\neq i} w_{ij}\xi_j^1 = \xi_i^1\left(\vec{\xi}^1, \vec{\xi}^1\right) - \frac{1}{N}\xi_i^1 = \left(1 - \frac{1}{N}\right)\xi_i^1 \tag{6.23}$$

(der $\frac{1}{N}$ Term folgt aus der wegen $i \neq j$ beschränkten Summation). Sei hier und im folgenden immer

$$\left|m^1\right| \geq |h_i| + \frac{1}{N}$$

dann folgt mit 6.22

$$\xi_i^1 = sign\left(\xi_i^1\left(\vec{\xi}^1, \vec{\xi}^1\right) - \frac{1}{N}\xi_i^1 + h_i\right) = sign\left(\xi_i^1\right) \tag{6.24}$$

Eine analoge Überlegung gilt für das Inverse $-\vec{\xi}^1$ des Musters. Am Fixpunkt ist folglich

$$\vec{\xi}^{Fix} = m^{(1)}\vec{\xi}^1, \qquad m^{(1)} = \pm 1$$

wobei $m^{(1)} = 1$ bei Speicherung des Musters und $m^{(1)} = -1$ für sein Inverses gilt.

Dynamik: Bei Start von einem beliebigen Anfangszustand $\vec{\xi}\,(0)$ ist man bei parallelem Updaten in einem Schritt am Fixpunkt (Muster oder sein Inverses), denn

$$\xi_i\,(1) = sign\left(\xi_i^1\left(\vec{\xi}^1, \vec{\xi}\,(0)\right) + h_i\right)$$

$$= sign\left(m^{(1)}\,(0)\;\xi_i^1\right) = \xi_i^1 sign\left(m^{(1)}\,(0)\right)$$

(falls nur $\left|m^1(0)\right| \geq |h_i| + \frac{1}{N}$). Für den Fall des asynchronen Updatens beachten wir, daß für das PSP

$$z_i(t) = \xi_i^1 m^{(1)}(t) - \frac{1}{N}\xi_i(t) = \xi_i^1 m^{(1)}(t) - \frac{1}{N}\xi_i(t)$$

gilt und damit

$$\xi_i(t+1) = \xi_i^1 sign\left(m^{(1)}(t)\right) \tag{6.25}$$

Eine Änderung von ξ_i und damit von $m^{(1)}$ findet nur statt falls (vergl. Kap. 6.1.2) $0 \neq \delta\xi_i = \xi_i^{neu} - \xi_i^{alt} = 2\xi_i^{neu} = 2sign\left(m^{(1)}\right)$. Die **dabei** stattfindende Änderung $\Delta m^{(1)} = m^{(1)}(t+1) - m^{(1)}(t)$ ist (mit $m^{(1)}(t) \geq \frac{1}{N}$)

$$\Delta m^{(1)} = \frac{1}{N}\left\{\xi_i(t+1) - \xi_i(t)\right\}\xi_i^1 = \frac{2}{N} sign\left(m^{(1)}(t)\right) \tag{6.26}$$

so daß also bei jeder Zustandsänderung eines Neurons die Überlappung mit dem Muster (oder seinem Inversen) immer weiter vergrößert wird oder konstant bleibt – jedes fehlerhafte Bit wird durchh einmaliges Updaten korrigiert. Für $\left|m^{(1)}\right| = 1$ kommt der Vorgang zum Stehen, da dann keine Zustandsänderung mehr erfolgt (Fixpunkt erreicht). Das ist genau dann der Fall, wenn jedes Neuron mindestens einmal aktualisiert wurde.

Damit ist dieses Netzwerk ein Assoziativspeicher, der bei Anlegen eines beliebigen Anfangszustand $\vec{\xi}(0)$ das Muster $\vec{\xi}^1$ (oder $-\vec{\xi}^1$) als Speicherinhalt ausgibt. Der Konfigurationsraum zerfällt in zwei Teilräume, die die Einzugsgebiete der beiden Speicherzustände sind.

Ein analoges Ergebnis folgt für mehrere Muster, falls diese ein **Orthogonalsystem** bilden. Das ist im allgemeinen nicht der Fall und die Speicherungseigenschaften hängen im allgemeinen stark von der Struktur des Mustersatzes ab. Analytische Ergebnisse allgemeinerer Natur können für den **Fall zufällig gewählter Muster** gewonnen werden.

Signal-Rauschanalyse für den Fall vieler Muster

Für Zufallsmuster können die Ergebnisse des Mattismodelles verwendet werden, indem das gesamte PSP in einen Signal- und einen Rauschterm zerlegt wird, der Signalterm entspricht dem Beitrag des einzelnen Musters im Mattismodell. Betr. Netzwerkzustand $\vec{\xi}$, dieser sei hinreichend nahe an einem der Muster, o.B. d. A. Muster 1

$$\left(\vec{\xi}, \vec{\xi}^1\right) = m^{(1)} \approx 1, \quad \left(\vec{\xi}, \vec{\xi}^\nu\right) = m^{(\nu)} = O\left(1/\sqrt{N}\right), \quad \text{für } \nu \neq 1$$

entsprechend Zerlegung des PSP

$$z_i = \vec{w}_i \cdot \vec{\xi} = \sum_{j,(j\neq i)} w_{i,j}\xi_j = \frac{1}{N}\sum_{j,(j\neq i)}\sum_{\mu=1}^{P} \xi_i^\mu \xi_j^\mu \xi_j = \xi_i^1 m^{(1)} + \sum_{\nu=2}^{P} \xi_i^\nu m^{(v)} = \text{Signal+"cross talk"} \tag{6.27}$$

Die $m^{(\nu)}$ und damit die $\xi_i^\nu m^{(v)}$ sind (Gaußverteilte) Zufallszahlen mit Streuung $1/\sqrt{N}$. Um einen Eindruck von der Stärke des cross–talk Termes zu bekommen, setzen wir $\vec{\xi} = \vec{\xi}^1$ und schreiben

$$z_i = \xi_i^1 + \zeta$$

Für den cross–talk Term ζ findet man elementar

$$\langle \zeta^2 \rangle = \frac{P}{N} = \alpha$$

falls alle Muster unkorreliert sind (wie bisher immer angenommen). Damit muß die Speicherung zusammenbrechen, wenn α in die Größenordnung von 1 kommt. Für $N \to \infty$ nähert sich ζ einer Gaußverteilung mit Breite α

$$p(\zeta) = \sqrt{\frac{N}{2\pi P}} e^{-\frac{N}{2P}\zeta^2}$$

Damit ist die Wahrscheinlichkeit P_{Fehler}, daß für ein Neuron das Rauschen größer als das Signal ist (was zu einer fehlerhaften Speicherung dieses Bits führt)

$$P_{Fehler} = \sqrt{\frac{N}{2\pi P}} \int_1^\infty d\zeta\, e^{-\frac{N}{2P}\zeta^2} \simeq \sqrt{\frac{P}{2\pi N}} e^{-\frac{N}{2P}}$$

Die mittlere Anzahl der fehlerhaft geseicherten Bits ist $P_{Fehler}N$. Damit ist eine korrekte Speicherung des Musters mit einer Wahrscheinlichkeit nahe 1 nur dann möglich, wenn (in der Grenze großer N)

$$P \leq N/\ln N$$

so daß für große N der kritische Wert von α mit N wie

$$\alpha_c \sim \frac{1}{\ln N} \tag{6.28}$$

geht.

Der Rauschterm entspricht dem Einfluss der restlichen Muster $\nu \neq \mu$ auf die Speicherung des betrachteten Musters μ, reflektiert also eine mangelnde Trennschärfe vergleichbar dem Effekt des Übersprechens zwischen verschiedenen Nachrichtenkanälen. Wenn der Rauschterm in die gleiche Größenordnung kommt wie der Signalterm, d. h.

$$P \sim N, \quad \text{d.h. } \alpha > 0 \quad \text{für } N \to \infty$$

(so daß α =endlich für $N \to \infty$) werden die Muster nicht mehr exakt gespeichert, d. h. eine bestimmte Zahl von Bits in dem gespeicherten Muster (retrieval state) ist gegenüber dem Muster "geflippt", falls überhaupt eine Speicherung möglich ist.

Für P endlich und $N \to \infty$, d.h-

$$\alpha = 0 \quad \text{für } N \to \infty$$

folgt allerdings, daß für $0 << \left|m^1\right| \leq 1$ der Signalterm beliebig stark überwiegt, so daß die gleichen Verhältnisse wie bei nur einem gespeicherten Muster vorliegen, d. h. **jedes Muster entspricht exakt einem Fixpunkt der Dynamik**, und jeder dieser Fixpunkte hat ein endliches Einzugsgebiet. Beachte, daß daneben noch weitere Zustände, z. B. **Mischzustände** (Überlagerung der Muster) als Fixpunkte auftreten können, z. B.

$$\xi_i^{mix} = sign\left(a_1\xi_i^1 + a_2\xi_i^2 + a_3\xi_i^3\right)$$

mit geeignet gewählten Koeffizienten a_i.

Für eine endliche Speicherkapazität α

$$\frac{P}{N} = \alpha$$

d. h. $P \sim N$, so daß $\alpha > 0$ für $N \to \infty$ sind die "konspirativen" Korrelationen im Rauschterm nicht mehr vernachlässigbar, ihr Verhalten verleiht dem Hopfieldnetzwerk folgende Eigenschaften:

- Eine genauere Analyse zeigt, daß mit steigendem α der Rauschterm von der gleichen Ordnung wie der Signalterm wird, für

  $$\alpha > \alpha_c \approx 0.138 \tag{6.29}$$

  erfolgt ein plötzliches Zusammenbrechen der Speicherung, d.h. wenn für unterkritisches α noch eine befriedigende Speicherleistung erreicht wird ($|m^\mu| \approx 1$ für das gespeicherte Muster μ), ist die Zahl der speicherbaren Muster unmittelbar oberhalb von α_c gleich Null, $(m^\mu \approx 0 \quad \forall\mu)$. Das ist ein Phasenübergang erster Ordnung ("Konfusionsübergang"). Die genauen Verhältnisse ergeben sich aus der Diskussion der Überlappung als Funktion der Kapazität α, s. Abbildung. (Der Wert für das kritische α gilt für $N \to \infty$ und ist das Ergebnis einer Replica-Rechnung ohne Brechung der Replica-Symmetrie. Mit Symmetriebrechung, also als realistischerer Wert folgt $\alpha_c = 0,142$).

- Diese zeigt auch, daß die Fixpunkte nur näherungsweise mit den Mustern übereinstimmen, d. h. die Speicherung ist nur näherungsweise korrekt, für $\alpha \nearrow \alpha_c$ werden nur noch ca. 98% der Bits richtig gespeichert (ist für die meisten Zwecke, z. B. Bildspeicherung, ohne Bedeutung).

- Neben den Mustern (bzw. ihren Inversen) und ihren Mischungen treten noch sog. Geisterzustände (*spurious states*) als Fixpunkte auf. Insgesamt ist die Zahl der Fixpunkte (Minima der Energiefunktion) von der Ordnung $\exp(cN)$ mit $c \approx 0,2$.

6.2.3 Stochastische Netzwerke – das Hopfield Modell bei endlichen Temperaturen

Wegen der vielen Nebenminima ist es zum Auffinden der Speicherinhalte günstig, eine "simulated annealing" Strategie zur Verfügung zu haben. Das gelingt über die Einführung einer stochastischen Update–Regel, d. h. die Vorschrift bei der Aktualisierung des Neurons i, den neuen Zustand des Neurons nach dem Vorzeichen des PSP auszurichten, $\xi_i(t+1) = sign(z_i(t))$, (Anpassungsregel), wird nur mit einer gewissen Wahrscheinlichkeit befolgt, mit der komplementären Wahrscheinlichkeit wird das Gegenteil dieser Vorschrift ausgeführt.

Einen geeigneten Ansatz für diese Wahrscheinlichkeit findet man auf folgende Weise. Die deterministische Update–Regel kann man auch so interpretieren, daß die Wahrscheinlichkeit $P(\xi_i = +1) = \theta(z_i)$ ist, d. h. die Wahrscheinlichkeit, nach dem Updaten das Neuron im Zustand $\xi_i = 1$ zu finden, ist gleich 1 falls $z_i \geq 0$ und gleich Null sonst. Analog ist $P(\xi_i = -1) = \theta(-z_i)$. Das stochastische Neuron ist durch eine aufgeweichte Sprungfunktion charakerisiert, allgemein üblich ist, die Sprungfunktion $\theta(z)$ durch die Fermifunktion $f_\beta(z)$ mit

$$f_\beta(z) = \frac{1}{1+e^{-2\beta z}} \tag{6.30}$$

zu ersetzen, d.h. die stochastische Update–Regel lautet

$$\begin{aligned} P(\xi_i^{neu} = 1) &= \tfrac{1}{1+e^{-2\beta z_i}}, \\ P(\xi_i^{neu} = -1) &= \tfrac{1}{1+e^{2\beta z_i}} \end{aligned} \tag{6.31}$$

mit $\beta = 1/T$ und T einer fiktiven Temperatur (die natürlich nichts mit der biologischen Temperatur des Nervengewebes zu tun hat). Dies entspricht der Glauberdynamik eines Spinsystems. Mit

$$f_\beta(-z) = 1 - f_\beta(z)$$

gilt auch

$$P(\xi_i^{neu} = \pm 1) = \frac{1}{1+e^{-(\pm)2\beta z_i}} \tag{6.32}$$

Die Wahrscheinlichkeit für die Übereinstimmung des neuen Zustandes mit dem Vorzeichen des PSP (Anpassung) kann andererseits in Termen der Ener-

gieänderung ΔH, vgl. 6.9 ausgedrückt werden (beachte $\Delta H \leq 0$):

$$P\left(\text{Anpassg}\right) = \frac{1}{1+e^{\beta \Delta H}} \tag{6.33}$$

und

$$P\left(\text{Anti-Anpssg.}\right) = 1 - P\left(\text{Anpassg}\right) = \frac{1}{1+e^{-\beta \Delta H}} \tag{6.34}$$

Das Verhältnis der Wahrscheinlichkeiten ist dementsprechend durch einen Boltzmannfaktor gegeben

$$\frac{P\left(\text{Anpassg}\right)}{P\left(\text{Anti-Anpssg.}\right)} = e^{-\beta \Delta H} \tag{6.35}$$

Wichtigste Folge dieser Regel: Die Dynamik konvergiert nicht, da bei $T \neq 0$ die Wahrscheinlichkeit für Zustandsänderungen nie Null ist. Allerdings werden bei nicht zu hoher Temperatur die Trajektorien vornehmlich in der Nähe der Fixpunkte der $T = 0$ Dynamik bleiben, wenn Sie diese einmal angelaufen haben.

Die Speicherinhalte sind auf diese Weise bei endlicher Temperatur nicht mehr eindeutig indentifizierbar, Aussagen etwa über die Wiedergabegenauigkeit (retrieval quality) nur im statistischen Mittel sinnvoll: Definiere Mittelwerte über den Netzwerkzustand als zeitlichen Mittelwert:

$$\left\langle \vec{\xi}\left(t\right) \right\rangle^{th} = \frac{1}{t_0} \sum_{t'=0}^{t_0} \vec{\xi}\left(t-t'\right), \quad t >> t_0, \qquad t_0 >> 1 \tag{6.36}$$

Alle obigen Formeln bleiben erhalten, wenn Zustand $\vec{\xi}(t)$ generell durch $\left\langle \vec{\xi}\left(t\right) \right\rangle^{th}$ ersetzt wird, speziell ist die Wiedergabegenauigkeit (retrieval quality) durch die Überlappung mit dem gemittelten Zustandsvektor des Netzwerkes gegeben

$$m^{(\mu)}\left(t\right) = \frac{1}{N} \sum_{i=1}^{N} \xi_i^{\mu} \left\langle \xi_i\left(t\right) \right\rangle^{th} \tag{6.37}$$

wobei die Überlappung auch bei ansonsten idealer Speicherung mit steigender Temperatur betragsmäßig abnehmen wird. Ebenso

$$\begin{aligned} \left\langle z_i \right\rangle^{th} &= \vec{w}_i \cdot \left\langle \vec{\xi} \right\rangle^{th} = \sum_{j,(j\neq i)} w_{i,j} \left\langle \xi_j \right\rangle^{th} = \frac{1}{N} \sum_{j,(j\neq i)} \sum_{\mu=1}^{P} \xi_i^{\mu} \xi_j^{\mu} \left\langle \xi_j \right\rangle^{th} \\ &= \xi_i^1 m^{(1)} + \sum_{\nu=2}^{P} \xi_i^{\nu} m^{(v)} = \text{Signal+"cross talk"} \end{aligned} \tag{6.38}$$

mit

$$m^{(\mu)} = \frac{1}{N} \vec{\xi}^{\mu} \cdot \left\langle \vec{\xi}\left(t\right) \right\rangle^{th}$$

Für die Existenz eines retrieval states (RS) muß schon bei $T = 0$ immer Signal ¿¿ "cross talk" gelten, gleichzeitig aber auch $0 << m^{(1)} < 1$. Für den retrieval state zum Muster $\vec{\xi}_\mu$ macht es folglich Sinn zu schreiben

$$\left\langle \xi_i^{retr} \right\rangle^{th} = m^\mu \xi_i^\mu + \text{"cross talk"} \tag{6.39}$$

Im folgenden sollen nun Formeln für die Existenz des RS abgeleitet werden.

Die Methode des mittleren Feldes

Speicherzustand für den Fall $P << N$:

Wir betrachten zunächst den Fall $\alpha \to 0$ für $N \to \infty$, d.h. eine endliche Anzahl P von Mustern, so daß der "cross talk" vernachlässigt werden kann. Aus 6.38 folgt, daß – bis auf das Vorzeichen – das PSP für alle Neuronen im thermischen Mittel gleich groß ist. Wir machen deshalb den Ansatz

$$\langle z_i \rangle = m \xi_i^\mu \tag{6.40}$$

Mit 6.38 liefert die Kombination beider Gleichungen m, wenn $\left\langle \xi_i^{retr} \right\rangle$ als Funktion von $\langle z_i \rangle$ bekannt. Ansatz des mittleren Feldes:

$$\langle z_i \rangle = z_i \tag{6.41}$$

(das entspricht der Annahme des Verschwindens der Fluktationen für $N \to \infty$). Bestimmung von $\langle \xi_i \rangle$ mittels **??**:

$$\langle \xi_i \rangle = \textstyle\sum_{\xi_i = \pm 1} \xi_i P(\xi_i) = P(\xi_i = +1) - P(\xi_i = -1) =$$

$$\frac{1}{1+e^{-2\beta z_i}} - \frac{1}{1+e^{+2\beta z_i}} = \frac{e^{\beta z_i} - e^{-\beta z_i}}{e^{\beta z_i} + e^{-\beta z_i}} = \tanh(\beta z_i)$$

Mit $\tanh(x) = -\tanh(-x)$ und $m = \langle \xi_i^\mu \xi_i \rangle$ folgt

$$m = \tanh(\beta m) \tag{6.42}$$

Die (grafische) Lösung dieser transzendenten Gleichung für m zeigt die Existenz einer kritischen Temperatur T_c oberhalb derer die Überlappung $m = 0$ (keine Speicherung) und unterhalb derer $m \neq 0$ (Speicherung). Bei Erhöhung der Temperatur gibt es also einen Phasenübergang an T_c (Zusammenbruch des Speicherverhaltens). T_c entspricht der Curietemperatur des Magnetismus.

Der Speicherzustand für den Fall endlicher Beladung ($\alpha > 0$ für $N \to \infty$):

Im Ansatz 6.39

$$\left\langle \vec{\xi}\,(t) \right\rangle = m^{(\mu)}\,\vec{\xi}^{\mu} + \text{"cross talk"} \tag{6.43}$$

weist der "cross talk"–Term auf die auch bei $T = 0$ noch fehlerhaft gespeicherten Bits hin. Die Stabilität der Speicherung eines jeden Bits hängt nun sowohl von der Stärke des cross talk ab als auch von der Stärke der Überlappung $m^{(\mu)}$, die ihrerseits durch die thermischen Fluktuationen reduziert wird. Beide Effekte bedingen im Wechselspiel die Speicherleistung des Netzwerkes.

Die genaue Bestimmung des Verhaltens bei endlicher Temperatur ist sehr schwierig und Gegenstand zahlreicher Untersuchungen. Verhalten inzwischen, zumindest in der replicasymmetrischen Näherung (s.u.), fast restlos aufgeklärt, insbesondere ex. ein detailliertes Phasendiagramm, d. h. Darstellung der Eigenschaften des Netzes in der α, T- Ebene, in der scharfe Grenzen die Existenzgebiete der einzelnen Phasen, die das System annehmen kann, markieren. Unterschieden wird speziell die Spinglasphase, in der eine exponentiell grosse Zahl von stationären Zuständen existieren, die aber keine Überlappung mit den Mustern mehr haben, von der retrieval Phase, bei der die Speicherinhalte stationären Zuständen entsprechen. Hier funktioniert das Netz als Assoziativspeicher auch bei höheren Temperaturen.

Zusammenhang mit Isingmodell des Ferromagnetismus:

Wir führen eine Umeichung des Netzwerkzustandes $\vec{\xi}\,(t)$ bezüglich eines der Muster, o. B. d. A. Muster 1 durch, d. h. definieren neue Zustandsvariable

$$\tilde{\xi}_i\,(t) = \xi_i^1 \xi_i\,(t) \quad \forall i = 1, 2, \ldots, N$$

analog $\tilde{z}_i = z_i\,\xi_i^1$. Interpretation: $\tilde{\xi}_i$ ist gleich $+1$ wenn $\xi_i = \xi_i^1$ d. h. gleiche Ausrichtung. Unter Vernachlässigung der Übersprechterme wird die Energiefunktion

$$H\left[\vec{\xi}\right] = H\left(\tilde{\xi}\right) = -\frac{1}{N}\sum_{i<j}\tilde{\xi}_i\tilde{\xi}_j \tag{6.44}$$

Interpretieren wir $\tilde{\xi}_i$ als Spins des Teilchens i, so ist 6.44 das Modell eines Ferromagneten mit konstanter Kopplung zwischen den Spins.

Mit 6.38 gilt

$$\langle \tilde{z}_i \rangle = m^{(1)} + \text{Rauschen} \tag{6.45}$$

In der neuen Eichung ist also bis auf das Rauschen das mittlere PSP für alle Neuronen gleich und entspricht dem mittleren Feld im Ferromagneten! Unter

Vernachlässigung des Rauschens gilt also auch

$$\langle \tilde{z}_i \rangle = \frac{1}{N} \sum_{i'=1}^{N} \langle \tilde{z}_{i'} \rangle = \frac{1}{N} \left\langle \sum_{i'=1}^{N} \tilde{z}_{i'} \right\rangle = \frac{1}{N} \sum_{i'=1}^{N} \tilde{z}_{i'} + O\left(\frac{1}{\sqrt{N}}\right)$$

oder

$$\langle \tilde{z}_i \rangle = \tilde{z}_i$$

in Übereinstimmung mit 6.41. Im thermodynamischen Limes ($N \to \infty$) kann also das thermische Mittel durch ein Mittel über die Plätze ersetzt werden (Voraussetzung: Die thermischen Fluktuationen des Platzmittels sind vernachlässigbar - keine "konspirativen" Korrelationen).

6.3 Informationstheorie und Statistische Mechanik

Eine detaillierte Analyse der Eigenschaften von Attraktornetzwerken mit symmetrischen Kopplungen (wie das Hopfieldmodell) gelingt mit Methoden der statistischen Mechanik, die im folgenden kurz dargestellt werden sollen. Wir setzen dabei keine Kenntnisse der Statistischen Mechanik voraus sondern leiten die gewünschten Ausdrücke aus der Informationstheorie ab.

6.3.1 Informationstheorie

Betrachte ein Alphabet $\mathcal{A}$ aus N_A Zeichen α.

Beispiel: Jeder Zustandsvektor $\vec{\xi} = (1, 1, -1, ..., 1, ..., -1, ...)$ eines NNs aus N Neuronen kann als ein Zeichen $\alpha \in \mathcal{A}$ interpretiert werden, $\alpha \in \{-1, 1\}^N$. Das Alphabet besteht dann aus 2^N Zeichen, d.h. $N_A = 2^N$.

Betrachte Nachrichtenquelle, die Zeichenketten $\alpha_1, \alpha_2, ...$ aus Buchstaben des Alphabets ($\alpha_i \in \mathcal{A}$) generiert.

Beispiel: Neuronales Netz mit der Update–Regel 6.5 generiert die Zeichenkette $\vec{\xi}(0), \vec{\xi}(1), \vec{\xi}(2), ...$

Über die Zeichenfolge seien nur die Wahrscheinlichkeiten (relative Häufigkeit) P_α für das Auftreten jedes Zeichens $\alpha \in \mathcal{A}$ bekannt. **Frage**: Wie viele Bits sind im Mittel **mindestens** zur Übertragung dieser Zeichenkette erforderlich. Um die Mindestzahl von Bits zu bestimmen überlege man sich eine geeignete Verschlüsselung der Kette (Datenkompression), z. B.

Übertrage nur den Abstand bis zum Wiederauftreten des Zeichens. Beispiel: sei $\mathcal{A} = \{a, b, c\}$. Die Zeichenkette $(a, b, b, b, a, c, a, a, a, c, ...)$ wird verschlüsselt als Tabelle

a	0	3	1	0	...
b	1	0	0	...	
c	5	3	...		

übertragen (der Abstand ist die Zahl der dazwischenliegenden Zeichen). Mit dieser Information kann die Zeichenkette eindeutig rekonstruiert werden.

Der mittlere Abstand zwischen zwei gleichen Zeichen α ist

$$d_\alpha = \frac{1}{P_\alpha} \tag{6.46}$$

da P_α die relative Häufigkeit (N_α/N_{Kette}) des Zeichens α ist. Der Abstand in Bits ist

$$d_\alpha^{[Bits]} = \log_2 \frac{1}{P_\alpha} = -\log_2 P_\alpha \tag{6.47}$$

Die im Mittel nötige Anzahl von Bits pro Zeichen ist demzufolge[2]

$$I = -\sum_\alpha P_\alpha \log_2 P_\alpha \tag{6.48}$$

Das ist die Shannonsche Formel für die Information einer Nachrichtenquelle.

Beispiele: Treten alle Zeichen gleich häufig auf, d. h. $P_\alpha = 1/N_A$ so ist

$$I = \log_2 N_A$$

d. h. die Information ist gleich der Anzahl der Zeichen des Alphabetes gemessen in Bits. Beispiele:

- Ist speziell $\mathcal{A} = \{-1, 1\}^N$, so ist in diesem Fall gleichwahrscheinlicher Zeichen $I = N$. Das ist klar, denn für zufällig gewählte Bitketten der Länge N gibt es keine Datenkompression die die Übertragung mit weniger denn N Bits ermöglichen würde.

- Treten nur K von den N_A Zeichen des Alphabetes überhaupt auf, so gilt bei Gleichwahrscheinlichkeit dieser Zeichen

$$I = \log_2 K \tag{6.49}$$

6.3.2 Statistische Mechanik

Die Boltzmann-Gibbssche Verteilung

Wir deuten die Mikrozustände eines physikalischen Systems als die Zeichen des Alphabets $\mathcal{A}$. Für unser NN sind das die Zustandsvektoren $\vec{\xi} \in \{-1, 1\}^N$. Angenommen wir hätten praktisch keine Information über die realisierten Mikrozustände im System uaßer der Tatsache, daß etwa gewisse makroskopische Größen einen vorgegebenen Wert annehmen sollen. Im Zustand des thermischen Gleichgewichtes ist das speziell die mittlere Energie des Systems

$$\langle H \rangle^{th} = \sum_\alpha H_\alpha P_\alpha \tag{6.50}$$

[2]Wir betrachten sehr lange Zeichenketten, so daß die nur einmal zu übermittelnde Information über die Zeilennummer der Tabelle nicht ins Gewicht fällt.

Für das NN ist $\alpha = \vec{\xi}$, so daß $H_\alpha = H\left[\vec{\xi}\right]$ und $\sum_\alpha$ der Summation über alle $\vec{\xi}$ entspricht, d. h.

$$\sum_\alpha g_\alpha = \sum_{\xi_1 \in \{-1,1\}} \ldots \sum_{\xi_N \in \{-1,1\}} g\left[\xi_1, \ldots \xi_N\right]$$

Gesucht ist die willkürfreie (d.h. ohne willkürliche Information einzuführen) Verteilung P_α der Mikrozustände des Systems, die diesen Mittelwert via **??** liefert. Lösung: Variation der Information 6.48 unter Berücksichtigung der Nebenbedingung **??**. Die Methode der Lagrangeschen Parameter liefert (wie in der SM üblich messen wir die Information nicht in Bits, d. h. $\log_2 \Rightarrow \log$).

$$0 = \frac{\partial}{\partial P_\alpha}\left(-\sum_\gamma P_\gamma \log P_\gamma + \beta \sum_\gamma H_\gamma P_\gamma\right) \qquad \forall \alpha \tag{6.51}$$

so daß

$$0 = -\log P_\alpha - 1 + \beta H_\alpha \tag{6.52}$$

mit Lösung

$$P_\alpha \sim e^{-\beta H_\alpha}$$

Unter Berücksichtigung der Normierung der Wahrscheinlichkeitsverteilung folgt

$$P_\alpha = \frac{1}{Z} e^{-\beta H_\alpha} \tag{6.53}$$

mit der sog. Zustandssumme Z als Normierungsfaktor

$$Z = \sum_\alpha P_\alpha \tag{6.54}$$

Die Verteilung 6.53 ist die Boltzmann-Gibbssche Verteilung der klassischen Statistischen Mechanik. Wir wählen die Einheiten so, daß die Boltzmannkonstante gleich 1 ist, dann ist

$$\beta = \frac{1}{T} \tag{6.55}$$

mit T der Temperatur des Systems (im Wärmebad).

Freie Energie und Entropie

Die Information 6.48 in "natürlichen" Einheiten gemessen heißt in der SM Entropie S

$$S = -\sum_\alpha P_\alpha \log P_\alpha \tag{6.56}$$

Analog zu 6.49 kann man folgern, daß

$$M = e^S \tag{6.57}$$

die mittlere Zahl von Zuständen ist, die im thermischen Gleichgewicht mit etwa gleicher Wahrscheinlichkeit realisiert werden. Entsprechend ist e^S bzw. S selbst ein Maß für die Information die man braucht, um einen speziellen der unter dem makroskopischen *constraint* $\langle H \rangle^{th} = E$ möglichen Mikrozustände festzulegen.

Die freie Energie F ist definiert als

$$F = \langle H \rangle^{th} - TS \tag{6.58}$$

so daß

$$F = -T \log Z \tag{6.59}$$

Damit kann die freie Energie als das erzeugende Funktional für die Bestimmung fast aller physikalischen Größen des Systems verwendet werden. So folgt z. B. für die Korrelationsfunktion

$$\langle \xi_i \xi_j \rangle^{th} = -\frac{\partial F}{\partial w_{ij}} \tag{6.60}$$

Führt man zum Hamiltonian H des Systems geeignete Hilfsfelder hinzu und bestimmt die F, so ergeben sich die gesuchten Größen durch Ableitung nach den Feldern. Beispiel:

$$H = -\frac{1}{2} \sum_{ij} \xi_i w_{ij} \xi_j \Rightarrow -\frac{1}{2} \sum_{ij} \xi_i w_{ij} \xi_j - \sum_i b_i \xi_i \tag{6.61}$$

dann ist

$$Z_b = \sum_{\xi_1 \in \{-1,1\}} \ldots \sum_{\xi_N \in \{-1,1\}} \exp\left(\frac{\beta}{2} \sum_{ij} \xi_i w_{ij} \xi_j + \beta \sum_i b_i \xi_i \right) \tag{6.62}$$

$$\tag{6.63}$$

$$= Tr_\xi \exp\left(\frac{\beta}{2} \sum_{ij} \xi_i w_{ij} \xi_j + \beta \sum_i b_i \xi_i \right) \tag{6.64}$$

(Tr_ξ bedeutet die Summation über alle Aktivitätsvariablen ξ_i). Mit 6.59 folgt

$$\langle \xi_i \rangle^{th} = \frac{1}{Z_b} \sum_{\xi_1 \in \{-1,1\}} \ldots \sum_{\xi_N \in \{-1,1\}} \xi_i \exp\left(\frac{\beta}{2} \sum_{ij} \xi_i w_{ij} \xi_j \right) \tag{6.65}$$

$$\tag{6.66}$$

$$= -\left. \frac{\partial F_b}{\partial b_i} \right|_{h=0} \tag{6.67}$$

wobei $F_b = -T \log Z_b$

Phasenübergänge und Brechung der Ergodizität

Wir haben oben die zeitlichen Mittel $\langle ... \rangle^{th}$ gleich dem Mittel über die Gesamtheit gesetzt. Das ist dann richtig, wenn für alle Zustände $\vec{\xi} \in \{-1, +1\}^N$ die Wahrscheinlichkeiten der Zustände gemäß der BG-Verteilung mit der relativen Häfigkeit des Auftretens dieses Zustandes im Laufe der Zeit übereinstimmt[3]. Da $P_\alpha = P\left[\vec{\xi}\right] > 0$ heißt das, daß das System jeden dieser Zustände mit der relativen Häufigkeit $P\left[\vec{\xi}\right]$ auch wirklich anläuft. Damit wandert der Systemzustand $\vec{\xi}(t)$ im Laufe der Zeit durch den gesamten Zustandsraum. Ein solches System heißt ergodisch.

Das kann aber eigentlich nicht richtig sein. Wegen $H\left[\vec{\xi}\right] = H\left[-\vec{\xi}\right]$ gilt nämlich ganz allgemein für die Boltzmann–Gibbs–Verteilung des Netzwerkes

$$P\left[\vec{\xi}\right] = P\left[-\vec{\xi}\right] \tag{6.68}$$

Für das Mattismodell ist z. B.

$$H = -\frac{1}{2}\sum_{i,j} \xi_i w_{ij} \xi_j = -\beta \frac{N}{2} \left(m^1\right)^2 \tag{6.69}$$

und $m^1 \Rightarrow -m^1$ falls $\vec{\xi} \Rightarrow -\vec{\xi}$. Damit folgt für den Mittelwert

$$\langle \xi_i \rangle^{th} = \sum_{\xi_1 \in \{-1,1\}} ... \sum_{\xi_N \in \{-1,1\}} \xi_i \exp\left(\frac{\beta}{2}\sum_{ij} \xi_i w_{ij} \xi_j\right) / Z \tag{6.70}$$

daß

$$\langle \xi_i \rangle^{th} = 0 \tag{6.71}$$

für alle Zustände $\vec{\xi}$. Das steht in eklatantem Widerspruch zu 6.43, d. h. (bei kleinem cross talk=geringe Zahl von Mustern)

$$\langle \xi_i \rangle^{th} = \xi_i^1 m^1 \tag{6.72}$$

mit $\left|m^1\right| > 0$ falls sich das Netzwerk in einem retrieval state befindet.

Der Widerspruch löst sich auf, wenn wir berücksichtigen, daß 6.72 unter der Bedingung gewonnen wurde, daß einerseits die Zeitmittelung mit Zeitfester t_0

[3]Wir nehmen an, daß das System sich in einem stationären Zustand befindet, d. h. daß hinreichend viel Zeit seit dem Initialisieren des Systems vergangen ist, so daß das zeitliche Mittel selbst nicht mehr zeitabhängig ist. Falls die Temperatur nicht zu groß ist fluktuiert der Systemzustand $\vec{\xi}(t)$ dann um einen Fixpunkt der $T = 0$ Dynamik.

zwar lang genug zu wählen ist, um die thermischen Fluktuationen sicher auszumitteln[4], $t_0 >> t_F$, daß aber andererseits $t_0 << t_E$ sein soll, wobei t_E die mittlere Zeit des Überganges (unter dem Einfluß der "thermischen" Störungen) von einem Einzugsgebiet in ein anderes ist. Die Wahrscheinlichkeit $P\left[\vec{\xi}\right]$ für die Realisierung eines bestimmten Zustandes $\vec{\xi}$ ist nur dann durch die Boltzmann–Gibbs–Verteilung gegeben, wenn wir das System (bei $N = endlich$) im Prinzip unendlich lange beobachten, so daß es tatsächlich **alle überhaupt erreichbaren** Zustände auch anläuft. Real sollte die Beobachtungszeit t_B aber zumindest $t_B >> t_E$ sein. t_B ist von der Zahl der Zustände abhängig und wächst mit diesen an. Man kann grob

$$t_E \sim e^{\gamma N} \tag{6.73}$$

schätzen. Jedenfalls geht diese Zeit mit $N \to \infty$ so daß es dann keine endliche Beobachtungszeit mehr gibt, unter der das System aus einem Einzugsgebiet eines Attraktors in ein anderes fliehen kann. Dann ist die Boltzmann–Gibbs–Verteilung nicht mehr anwendbar, bzw. die Boltzmann–Gibbs–Verteilung ist nur in dem Sinne anwendbar, daß man eine auf die jeweilige Phase beschränkte Summation für die Mittelwertbildung ausführt, d. h. anstelle von 6.71 gilt für den Mittelwert einer beliebigen Zustandsgröße $f\left[\vec{\xi}\right]$

$$\left\langle f\left[\vec{\xi}\right]\right\rangle^{th} = \sum_{\vec{\xi}\in\text{Phase}} f\left[\vec{\xi}\right] \exp\left(\frac{\beta}{2}\sum_{ij}\xi_i w_{ij}\xi_j\right)/Z_b \tag{6.74}$$

Für den Fall des einfachen Mattismodells finden wir z. B.

$$\left\langle f\left[\vec{\xi}\right]\right\rangle^{th} = \sum_{\vec{\xi}\,\text{mit}\,m^1>0} f\left[\vec{\xi}\right] e^{-\frac{N}{2}m^1} \tag{6.75}$$

was für $f = \xi_i$ wieder auf 6.43

$$\langle\xi_i\rangle^{th} = \xi_i^1 m^1 + \text{"cross talk"} \tag{6.76}$$

führt.

Dieses Scenario zeigt, daß bei der Existenz von Phasenübergängen der Zustandsraum in verschiedene, zu den einzelnen Phasen assoziierte Gebiete unterteilt zu denken ist, die das System nicht verlassen kann. Energetisch bedeutet

[4]Die Mittelungszeit muß also so lang sein, daß alle Zustände, die bei der Fluktuation um einen Attraktorzustand des Netzes bei $T = 0$ möglich sind auch wirklich oft genug aufgetreten sind.

das, daß zwischen diesen Gebieten für $N \to \infty$ unendlich hohe Energiebarrieren bestehen. Das entspricht einer **Brechung der Ergodizität** des Systems. Im Hopfieldmodell bzw. im Spinglas ist das Besondere, daß wir es mit einer makroskopisch großen Vielzahl von nichtergodischen Phasen zu tun haben.

Kapitel 7

Neuronale Netze und abstrakte Automaten

Sowohl die bisher besprochenen Feed–Forward–Netze als auch die rekurrenten Netze können als spezielle abstrakte Automaten aufgefaßt werden. Auch die Attraktornetzwerke stehen in enger Verwandschaft zu endlichen Automaten und sind insbesondere Verallgemeinerungen von zellulären Automaten.

7.1 Abstrakte Automaten

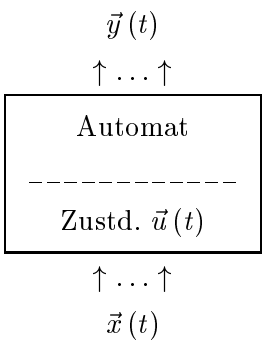
Ein abstrakter Automat ist ein Tupel

$$\mathbf{A} = (\mathbf{X}, \mathbf{Y}, \mathbf{U}, f, g) \tag{7.1}$$

mit

- $\mathbf{X}$ = Eingabealphabet, mit den Elementen $x_i \in \mathbf{X}$, die Eingabebuchstaben genannt werden. Analog ist $\mathbf{Y}$ das Ausgabealphabet.
- $\mathbf{U}$ ist die Menge der inneren Zustände.
- Die Funktionen f bzw. g vermitteln eindeutige Abbildungen der Produktmengen $\mathbf{X}\times\mathbf{U}$ in die Menge der Zustände $\mathbf{U}$ bzw. in die Ausgabemenge $\mathbf{Y}$. Diese Abbildungen definieren die Übergänge zwischen den Zuständen bzw. die Ausgabe des Automaten als Funktion der Eingabe und des aktuellen inneren Zustandes von A, s.u. Folglich heißt f auch die Überführungs– und g die Ergebnisfunktion des Automaten.

Die Arbeit des Automaten



(7.2)

ist bezüglich der getakteten Zeit $t = 0, 1, ...$ durch folgenden Algorithmus bestimmt:

1. Initialisierung. Zum Zeitpunkt (o.B.d.A.) $t = 0$: Einlesen des inneren Zustandes (Startzustand) $\vec{u}$ des Automaten.

2. Einlesen eines Buchstabens $\vec{x}$ aus dem Eingabealphabet. Bestimmung der Ausgabe $\vec{y}\,(t)$ und des Folgezustandes $\vec{u}\,(t+1) = \vec{u}^{neu}$ seines aktuellen inneren Zustandes $\vec{u}$

$$\vec{y}\,(t) = g\,(\vec{x}, \vec{u}) \tag{7.3}$$

$$\vec{u}\,(t+1) = f\,(\vec{x}, \vec{u})$$

3. $\vec{u} = \vec{u}^{neu}$ ($\vec{u}^{neu}$ wird neuer aktueller innerer Zustand). $t = t + 1$

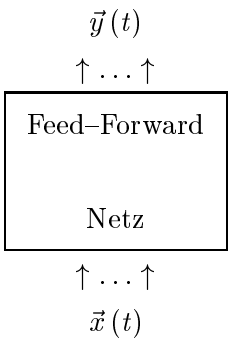
4. Abbruch? $GOTO$ 2

Als endliche Automaten werden speziell solche bezeichnet, für die die Mengen $\mathbf{X}, \mathbf{Y}, \mathbf{U}$ endlich sind.

7.2 Neuronale Netze als abstrakte Automaten

Im folgenden wollen wir den formalen Zusammenhang zwischen verschiedenen neuronalen Netzen und abstrakten Automaten darstellen.

- Die in Kap. 3 besprochenen Feed–Forward–Netze



(7.4)

  sind spezielle Automaten, die sich von 7.2 einzig dadurch unterscheiden, daß sie nicht über einen inneren Zustand verfügen, der die Ausgabe in Abhängigkeit von den Inputs $\vec{x}$ beeinflussen könnte. Die Buchstaben des Ein– bzw. Ausgabealphabetes des Automaten sind mit der Menge aller möglichen Input– bzw. Outputvektoren des Netzes zu identifizieren. Feed–Forward–Netze können wegen dieser Eigenschaft als ein im Sinne der Signaltheorie kombinatorische Systeme bezeichnet werden.

- **Rekurrente Netze:** Der fehlende innere Zustand der Feed–Forward–Netze macht den Output des Netzes zu einer direkten Funktion des aktuellen Inputs des Netzes. Auf diese Weise geht die Vorgeschichte der Inputs nicht in den Ouput ein. Das ist vor allem für die Verwendung von Netzen als Modelle von Zeitreihen oder allgemeiner von dynamischen Systemen oft ein entscheidender Nachteil.

  Dieses Problem kann durch die Verwendung rekurrenter Netze beseitigt werden. In rekurrenten Netzen koppeln ausgewählte Neuronen ihren Output auf sich selbst oder auf andere Neuronen zurück. Die Outputs dieser ausgewählten Neuronen können als innerer Zustand des Netzes $\vec{u}$ interpretiert werden, der nach der Regel 7.3 upgedatet wird. Damit sind rekurrente Netze Realisierungen "echter" abstrakter Automaten und können wie diese eine innere Repräsentation der Vorgeschichte des aktuellen Inputs aufbauen.

  Für die Beziehungen zwischen rekurrenten neuronalen Netzen und abstrakten Automaten existieren eine Reihe von strengen Aussagen. So wurde schon 1991 von Alon, Dewdney und Ott [3] gezeigt, daß rekurrente neuronale Netze jeden endlichen abstrakten Automaten in Echtzeit simulieren können. Siegelman und Sontag [36] bewiesen 1995 die gleiche Aussage für

die Simulation einer multi–stack Turing Maschine. Damit besitzen die rekurrenten Netze die gleiche Mächtigkeit wie diese Berechnungsmodelle.

Diese Aussagen sind zunächst nur recht prinzipieller Natur, es konnten aber auch Ergebnisse für Netze konkreter Architekturen gewonnen werden. So gilt z. B. der Satz, daß sich rekurrente Netze vom Typ der von Elman, vgl. [18, 19] zuerst eingeführten Architektur zur Simulation eines beliebigen endlichen Automaten in Echtzeit eignen, während rekurrente Kaskadenkorrelationsnetze diese Performanz nicht haben, vgl. [37].

- **Attraktornetzwerke (Hopfieldnetz):** Eine weitere Spielart der abstrakten Automaten sind die in **??** dargestellten Attraktornetzwerke (Hopfielnetzwerk). Diese dienen als Assoziativspeicher, die einen beliebigen Inputvektor $\vec{x}$ auf einen der gespeicherten Mustervektoren $\vec{x}^{\mu}$ abbilden. Diese Abbildung ist ein Ergebnis der Attraktordynamik. Das Netz wird mit $\vec{x}$ als Anfangszustand gestartet und erreicht nach endlicher Zeit einen Fixpunkt, der die Ausgabe des Speichers darstellt. Das entspricht einem Automaten mit Ergebnis– bzw. Überführungsfunktionen, die nicht von einem Eingabebuchstaben abhängen, d. h.

$$\vec{y}\,(t) = g\,(\vec{u}) \tag{7.5}$$

$$\vec{u}\,(t+1) = f\,(\vec{u})$$

anstelle von 7.3.

Beim Hopfieldnetz ist speziell $g\,(\vec{u}) = \vec{u}$. Die Eingabe erfolgt über die Initialisierung des inneren Zustandes, d. h. $\vec{u}_0 = \vec{x}$. Die Überführungsfunktion ist so zu wählen, daß der Automat nach einer endlichen Zahl von Schritten zum Halten kommt. Die Ausgabe des Assoziativspeichers ist mit $\vec{y} = g\left(\vec{u}^{halt}\right)$ durch den inneren Zustand $\vec{u}^{halt}$ des Automaten gegeben. Wegen 7.5 sind die Attraktornetze spezielle Realisierungen des Moore–Automaten.

7.3 Stochastische Automaten und Netze

In Vorbereitung

Kapitel 8

Selbstorganisierende Karten

Wir betrachten nun eine wichtige Klasse von neuronalen Netzen, die als Modelle neuronaler Karten dienen. Karten stellen ein Hauptelement der kortikalen Informationsverarbeitung dar. Von verschiedenen Autoren wird die gesamte Informationsverarbeitung im Kortex als eine Hierarchie von Karten in Stufen steigender Abstraktion der primären Sinneseindrücke (sensorische Inputs) betrachtet (van Essen). Auf jeden Fall spielen Karten eine wichtige Rolle beim Aufbau einer inneren Repräsentation der Außenwelt. Diese Repräsentation ist immer informationsreduziert, d. h. es findet immer eine Datenreduktion statt. Jedes Neuron, das an der Repräsentation eines primären Sinneseindruckes beteiligt ist, überdeckt mit seinem rezeptiven Feld einen Teil der sensorischen Informationsflächen und ist damit für eine bestimmte Auswahl der sensorischen Reize "zuständig".

Im folgenden werden wir uns zunächst mit der datenreduzierenden Projektion der sensorischen Inputs auf die zuständigen Neuronen beschäftigen. Technisch führt das auf den Begriff des Vektorquantisierers und auf die Frage, wie eine solche Datenreduktion möglichst effektiv, d. h. mit möglichst geringen Informationsverlusten, geleistet werden kann. Wie sich zeigen wird, kann das mit einem einfachen Wettbewerbslernen biologisch effektiv realisiert werden. Danach soll die Organisation der Einzelprojektionen auf die Neuronen in eine topografische Abbildung untersucht und insbesondere die Frage studiert werden, wie diese topografischen Abbildungen gelernt werden können.

8.1 Der Vektorquantisierer

8.1.1 Rezeptive Felder und Voronoiparkettierung des Inputraumes

Wir betrachten eine Menge von Neuronen $i = 1, ..., N$, die alle von den n Inputunits $l = 1, ..., n$ gespeist werden. Die Inputvektoren (Merkmalsvektoren) $\vec{x} = (x_1, ..., x_n) \in \mathcal{R}^n$ werden als Punkte im n - dimensionalen Raum gedacht. Jedes Neuron i führt den Vektor $\vec{w}_i = (w_{i1}, ..., w_{in})$ seiner synaptischen Gewichte, der ebenfalls als Punkt im $\mathcal{R}^n$ interpretiert werden kann. Die Menge der $\{\vec{w}_i\}$ definiert eine Voronoiparkettierung des $\mathcal{R}^n$ d. h. eine Zerlegung des Raumes in die Menge $\mathcal{V} = \{V_i \mid i = 1, ..., n\}$ der Voronoizellen der einzelnen Vektoren $\vec{w}_i$, so daß

$$\mathcal{R}^n = V_1 \cup V_2 \cup ... \cup V_N$$

Sei im $\mathcal{R}^n$ ein Abstand $D(\vec{x}, \vec{y})$ zwischen zwei beliebigen Vektoren $\vec{x}$, $\vec{y}$ definiert. Das kann z. B. der Euklidische Abstand

$$D(\vec{x}, \vec{y}) = \sqrt{\sum_i (x_i - y_i)^2}$$

sein. Die zum Vektor $\vec{w}_i$ gehörige Voronoizelle V_i ist definiert als das Gebiet aller $\vec{x}$, die zu $\vec{w}_i$ einen kleineren (oder höchstens den gleichen) Abstand als zu allen restlichen Repräsentanten haben

$$V_i = \{\vec{x} \mid D(\vec{x}, \vec{w}_i) \leq D(\vec{x}, \vec{w}_j) \quad \forall j \neq i\} \tag{8.1}$$

Die Grenzen der einzelnen Voronoizellen sind folglich Hyperebenen, die die Verbindungslinien zwischen benachbarten Repräsentanten auf halbem Wege senkrecht schneiden.

Betrachtet man die Übereinstimmung des Inputs mit dem synaptischen Vektor als Maß für die Erregung des Neurons, so ist die Voronoizelle das Gebiet der Inputs, die das betrachtete Neuron i am stärksten erregen. Dieses wird deshalb für gegebenes $\vec{x}$ auch als das *Siegerneuron* oder best match Neuron bezeichnet. Die Voronoizelle ist damit die Domäne des Neurons und ist gleichbedeutend mit seinem rezeptiven Feld.[1]

[1] In der neurobiologischen Formulierung können die rezeptiven Felder durchaus auch überlappend sein.

8.1.2 Der Vektorquanisierer (VQ)

VQ werden allgemein zur **Datenreduktion** eingesetzt, d. h. zur problemangepaßten Diskretisierung (=Quantisierung) oder Partitionierung von hochdimensionalen Datenräumen, für die bei der Kodierung ein gewisser Informationsverlust hingenommen werden kann. (Unterschied zur **Datenkompression** bei der eine vollständige Rekonstruktion der Ausgangsdaten möglich sein soll). Ziel ist, eine diskrete Menge von Datenvektoren $\vec{x}$ in Gruppen zusammenzufassen, die jeweils durch einen einzigen Vektor, ihren **Codebuchvektor** oder **Repräsentanten**, repräsentiert werden. Das entspricht der Aufteilung der Menge der Datenvektoren in Teilmengen entsprechend den Gruppen $i = 1, 2, \ldots, N$, wobei allen $\vec{x} \in$ Gruppe i der Repräsentant $\vec{w}_i$ zugeordnet wird (**Kodierungsschritt**). Umgekehrt ist die Gruppierung durch die Voronoiparkettierung des Raumes in eindeutiger Weise vorgegeben. Zur Informationsübertragung wird dann nur der Index i übertragen und im Decodierungsschritt dann $\vec{w}_i$ ausgegeben

$$\text{Input} \quad \vec{x} \quad \Rightarrow \quad \boxed{\begin{array}{c}\text{Kodierer} \\ \vec{x} \;\rightarrow\; i\end{array}} \quad \rightarrow \quad i \quad \rightarrow \quad \boxed{\begin{array}{c}\text{Dekodierer} \\ i \;\rightarrow\; \vec{w}_i\end{array}} \quad \Rightarrow \vec{w}_i \quad \text{Output} \tag{8.2}$$

Der Kodierer ist also durch die Funktion $i : \vec{x} \rightarrow i(\vec{x})$ gegeben, wobei (s. 8.1)

$$i = \arg\min_j D(\vec{x}, \vec{w}_j) \tag{8.3}$$

d. h. die Datenvektoren zum Repräsentaten $\vec{w}_i$ liegen alle in dessen Voronoizelle.

Frage ist nun, wie eine möglichst effektive Vekrorquantisierung aussieht. Zur Beurteilung der Güte einer Vektorquantisierung gibt es im wesentlichen zwei Kriterien – die übertragene Transinformation und den Rekonstruktionsfehler.

Informationstheoretische Betrachtungen

Der VQ kann als ein Kodierer betrachtet werden, der Inputs $\vec{x}$ Klassenlabel $l, \quad l = 1, \ldots, L$ zuordnet. Wir stellen nun zunächst die Frage nach der im Sinne der Informationstheorie optimalen Form der Kodierung. Diese Frage kann ganz allgemein ohne Bezug auf eine spezielle Realisierung eines VQ beantwortet werden. Die geeignete Größe ist die Transinformation, die die durch den Kodierer übertragene Information mißt. Wenn die Inputvektoren $\vec{x} \in \Re^n$ stochastische Vektoren mit einer Verteilung $p(\vec{x})$ sind, so ist die Inputinformation

$$I^{(in)} = -\sum_{\vec{x}} p(\vec{x}) \ln p(\vec{x})$$

Wegen der Zuordnung $\vec{x} \rightarrow \vec{w}_l$ bzw. $\vec{x} \rightarrow$ Klassenlabel l, $\quad l = 1, \ldots, L$ definiert die Verteilung $p(\vec{x})$ auch eine output Verteilung $P(l)$ mit Information

$$I^{(out)} = -\sum_{l=1}^{L} P(l) \ln P(l)$$

Die bedingte output Information $I^{(out|\vec{x})} = -\sum_{l=1}^{L} P(l \mid \vec{x}) \ln P(l \mid \vec{x})$ bzw. im Mittel über die Inputs

$$I^{(out|in)} = \left\langle I^{(out|\vec{x})} \right\rangle = \sum_{\vec{x}} p(\vec{x}) \sum_{l=1}^{L} P(l \mid \vec{x}) \ln P(l \mid \vec{x})$$

ist ein Maß für die output Information bei festgehaltenem Input $\vec{x}$. Bei einer (wie im betrachteten Fall) deterministischen Zuordnung ist $P(l \mid \vec{x})$ entweder Null oder Eins und damit ist die bedingte Information gleich Null, wie es sein muß, denn die bedingte Information mißt die durch das Rauschen des Kodierers erzeugte "falsche" Information.. Die am Ausgang anliegende nutzbare oder unverfälschte Information ist gerade die Transinformation, d. h. die durch den Kodierer übertragene Information

$$I^{(trans)} = I^{(out)} - I^{(out|in)}$$

Im betrachteten Fall gilt also $I^{(trans)} = I^{(out)}$ und der optimale Kodierer ist durch $I^{(out)} = \max$ gegeben, was durch

$$P(l) = \frac{1}{L}$$

erfüllt wird, d., h. **der Kodierer arbeitet optimal, wenn alle Repräsentanten gleich oft angesprochen werden**, wenn also alle Voronoizellen im Mittel gleich viele Datenpunkte enthalten. Für $p(\vec{x}) = const$ heißt das, daß alle Domänen gleich groß sind.

Der Rekonstruktionsfehler

Der Rekonstruktionsfehler drückt den durch die Ersetzung der $\vec{x}$ durch die $\vec{w}$ gemachten Fehler in Termen der mittleren quadratischen Abweichung aus

$$E(\mathbf{w}) = E(\{w_i \mid i = 1, \ldots, N\}) = \langle E(\mathbf{w}, \vec{x}) \rangle = \sum_{\vec{x}} p(\vec{x})\ E(\mathbf{w}, \vec{x})$$

mit

$$E(\mathbf{w}, \vec{x}) = D\left(\vec{w}_{i(\vec{x})}, \vec{x}\right)^2$$

wobei $i(\vec{x})$ der Index des Siegervektors zu $\vec{x}$ ist, d. h. $D\left(\vec{w}_{i(\vec{x})}, \vec{x}\right) = \min_l D(\vec{w}_l, \vec{x}) \quad \forall \vec{x}$.

Bei gleichverteilten Daten ist der Rekonstruktionsfehler am kleinsten, wenn alle Zellen gleich groß sind. Das enspricht also auch dem Maximum der Transinformation. Bei inhomogenen Datenverteilungen liefern beide Maße aber durchaus unterschiedliche optimale VQ.

8.1.3 Wettbewerbslernen

Die Bestimmung des besten Satzes von Prototypvektoren erfordert die Minimierung von E bzw. die Maximierung von I^{trans} als Funktion der $\vec{w}$. Das kann durch einen Gradientenfolgeverfahren erreicht werden. Für die Minimierung des Rekonstruktionsfehlers E wird dabei jeder Vektor $\vec{w}_i$ schrittweise gemäß

$$\Delta \vec{w}_{i\alpha} = -\epsilon \frac{\partial E}{\partial w_{i\alpha}} \tag{8.4}$$

verändert verändert.

Dazu muß aber der mittlere Fehler E bekannt sein, d. h. man muß schon alle Daten vorliegen haben. Ein inkrementelles Verfahren, bei dem mit jedem neu einkommenden Datenvektror $\vec{x}$ eine "infinitesimal" kleine Veränderung der vorgenommen wird, ergibt sich durch die stochastische Approximation des Gradientenabstieges 8.4. Das wird mit folgendem Algorithmus realisiert: Bei Eingabe von $\vec{x}$ bestimmen wir zunächst das "Siegerneuron" i, d. h. den *best match* Repräsentanten $\vec{w}_i$) und führen für dieses den Lernschritt gemäß

$$\Delta \vec{w}_i = -\epsilon \left(\vec{w}_i - \vec{x}\right) \qquad i = i(\vec{x}) \tag{8.5}$$

aus, mit geeigneter Abkühlungsregel für den Lernparameter ϵ.(Linde, Buzo, Gray 1980, s. [27]). Der Algorithmus erzeugt einen zeitdiskreten stochastischen Prozeß $\vec{w}(t)$. Im stationären Zustand gilt $\langle \Delta \vec{w} \rangle = 0$ und folglich

$$\langle \vec{w}_l \rangle^{(l)} = \langle \vec{x} \rangle^{(l)}$$

wobei die $\langle \vec{x} \rangle^{(l)}$ die Mittelung über die Domäne des Repräsentanten $\vec{w}_l$ ist. Der Repräsentant ist also bei sorgfältigem "Einfrieren" gleich dem Mittelwert der in seiner Domäne gelegenen Inputvektoren $\vec{x}$.

Das Verfahren heißt Wettbewerbslernen, weil die Neuronen im Wettbewerb um die beste Repräsentation der Inputs stehen.

8.1.4 Der überwacht lernende Vektorquantisierer (LVQ)

Das oben eingeführte Wettbewerbslernen als Verfahren zur Optimierung eines Vektorquantisierers ist ein nichtüberwachtes Lernverfahren. In der Tat wird kein Information durch einen Lehrer vorgegeben, die Positionierung der Repräsentanten und damit die Gruppierung der Inputvektoren erfolgt rein nach statistischen Gesichtspunkten.

Der LVQ als Klassifikator

Ein Klassifikator teilt die Inputvektoren in L **vorgegebene** Klassen ein. Der LVQ ist ein lernender Klassifikator, der anhand von Beispielen lernt, die Inputvektoren in genau diese Klassen einzuteilen. Die Klassen belegen bestimmte Gebiete im Raum der Inputvektoren. Diese Gebiete müssen mit Repräsentanten der jeweiligen Klasse belegt werden. Für eine möglichst variable Grenzziehung zwischen den Gebieten der verschiedenen Klassen müssen die Klassen jeweils durch hinreichend viele Prototypvektoren repräsentiert werden. Zu jeder Klasse Γ_k gehört folglich eine Menge von klassenspezifischen Repräsentanten $\vec{w}_{k_l}$, $l = 1, \ldots, n_k$. Zur Klassifikation wird $\vec{x}$ genau dann der Klasse zugeordnet, wenn es **einem** der Repräsentanten dieser Klasse den kleinsten Abstand hat.

Ziel des Lernverfahrens ist es, die Repräsentanten $\vec{w}_{k_l}$ zu lernen. Ein gut geeignetes Verfahren wurde von *Kohonen* [25] angegeben und *Lernender Vektorquantisierer* (LVQ) genannt. In der Grundversion lautet der Lernalgorithmus des LVQ für eine Euklidische Abstandsfunktion und für diskrete Zeiten $t = 0, 1, \ldots$

$$\Delta\vec{w}_{k_l}(t) = \epsilon_t \left[\vec{x} - \vec{w}_{k_l}(t)\right] \quad bei\ richtiger\ Klassifikation \tag{8.6}$$

und

$$\Delta\vec{w}_{k_l} = 0 \quad bei\ falscher\ Klassifikation$$

Es lernt also immer nur der Repräsentant mit dem geringsten Abstand zum Inputvektor $\vec{x} \in \Re^n$. Dieser Repräsentant kann als Sieger im Wettbewerb um die Repräsentation des aktuellen Inputs betrachtet werden. Zur "Belohnung" darf der Sieger lernen (*Wettbewerbslernen*). Ergebnis des Lernschrittes ist eine Annäherung des *best match* Repräsentanten an **einen** der zu seiner Klasse gehörigen Merkmalsvektoren. Über viele Lernschritte erfolgt somit eine langsame "Diffusion" der Repräsentanten einer bestimmten Klasse in das für diese Klasse relevante Gebiet des Datenraumes. Die Gebiete zwischen den einzelnen

Klassen werden durch einen sich aus der *Voronoi-Parkettierung* ergebenden Polygonzug begrenzt.

Probleme bei diesem Lernverfahren sind sog. *dead units*, d. h. Repräsentanten $\vec{w}_{k_l}$ die auf der Trainingsmenge niemals Sieger werden und demzufolg überflüssig sind (Speicherplatz!) und die Tatsache, daß man die Zahl der Repräsentanten pro Klasse (die die Struktur der Gebietsgrenze bestimmt) von außen vorgeben muß.

Varianten des LVQ

Eine Erhöhung der Lerngeschwindigkeit und teilweise Verringerung von Fehlklassifikationen ergibt sich bei folgendem Lernalgorithmus

$$\Delta\vec{w}_{k_l} = \epsilon_t \, [\vec{x} - \vec{w}_{k_l}] \quad \text{bei richtiger Klassifikation}$$

und

$$\Delta\vec{w}_{k_l} = -\epsilon_t \, [\vec{x} - \vec{w}_{k_l}] \quad \text{bei falscher Klassifikation}$$

Ein klassenfremder Prototypvektor $\vec{w}_{k_l}$ wird also vom Datenpunkt $\vec{x}$ wegbewegt. Eine weitere Verbesserung stellt der **LVQ2** dar, bei dem der Lernschritt für den Prototypvektor $\vec{w}_{k_l}$ nur ausgeführt wird, wenn

1. Der Vektor $\vec{x}$ fehlklassifiziert wurde, d. h. $\vec{x}$ ist seinem Repräsentanten klassenfremd.

2. Der zweitnächste Prototypvektor (zweiter Sieger) zur Klasse von $\vec{x}$ gehört,

3. der Merkmalsvektor $\vec{x}$ hinreichend dicht an der Gebietsgrenze zwischen beiden Prototypvektoren liegt.

Der Algorithmus steht mit einigen Anwendungsbeispielen als *public domain* Programm zur Verfügung.

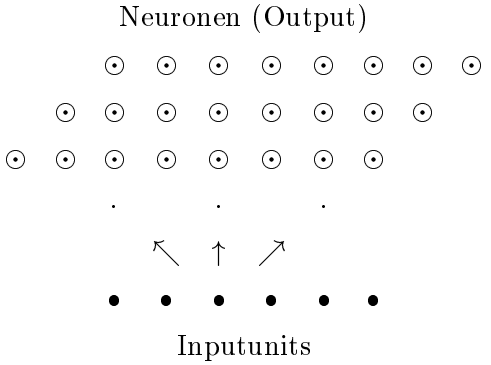
Eine Verallgemeinerung ist der dynamisch lernende Vektorquantisierer, der sich die erforderlichen Repräsentanten selbst allokiert, so daß diese nicht von Hand vorgegeben werden müssen, s. [45]. Der Algorithmus wurde u. a. erfolgreich in der Spracherkennung angewendet.

8.2 Selbstorganisation topografischer Abbildungen – der Algorithmus von Kohonen

Das reine Wettbewerbslernen sichert nur eine möglichst optimale Aufteilung des Inputraumes auf die Neuronen. Für eine innere Repräsentaton der Außenwelt muß die Abbildung allerdings auch die räumlichen Beziehungen zwischen den Reizen erhalten. Das würde eine geordnete "Verdrahtung" zwischen den Rezeptoren und den kortikalen Neuronen voraussetzen, die aber so zum Beginn der ontogentischen Entwicklung nicht vorhanden ist. Einfache informationstheoretische Überlegungen zeigen auch, daß die Information über die Organisation dieser Verschaltung nicht schon in den Genen enthalten sein kann. Im Gegenteil entsteht die geordnete Verschaltung, getrieben durch ständige Reizung der sensorischen Flächen, im Laufe der Entwicklung des ZNS quasi von allein. Für diesen Akt der Selbstorganisation gibt es eine Reihe mehr oder weniger realistischer Modelle, von denen wir hier nur den auf Kohonen zurückgehenden Lernalgorithmus für die synaptischen Gewichte besprechen wollen.

Unter einer Karte verstehen wir die **topografische** Abbildung eines Inputraumes V auf einen Outputraum $\mathcal{A}$. Solche Abbildungen werden durch spezielle einlagige neuronale Netze realisiert, d. h. der Outputraum $\mathcal{A}$ ist eine Neuronenschicht, wobei jedes Neuron von allen Inputunits direkt gespeist wird, s. Abbildung 8.1.

Die Topologie des Neuronenraumes $\mathcal{A}$ ist durch die Vorgabe der Orte $\vec{r}_i$ der Neuronen $i = 1, 2, ..., N$ festgelegt. Diese können die Knoten (Vertices) eines beliebigen Graphen sein, im konkreten Fall der Darstellung 8.7 bzw. der Abbildung 8.1 sitzen die Neuronen auf den Gitterpunkten eines zweidimensionalen Quadratgitters, d. h. $\vec{r}_i = (n_{i_1}, n_{i_2})\,,\ n_{i_j} \in \{0, 1, 2, ...\}$.



(8.7)

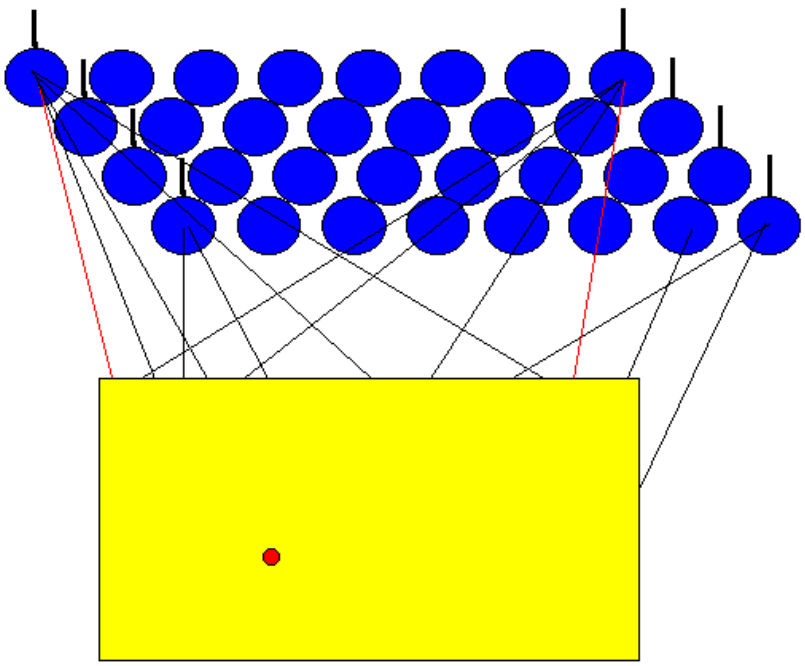


Abbildung 8.1: Abbildung einer zweidimensionalen Sinnesoberfläche (gelb) auf ein kortikales Areal, das hier durch ein zweidimensionales Gitter von Neuronen idealisiert ist. In der dargestellten postnatalen Situation sind alle Rezeptoren der Sinnesoberfläche mit allen Neuronen durch synaptische Kopplungen unterschiedlicher Stärke verbunden. Jedes Neuron sieht einen entsprechend hochdimensionalen Input. Durch Striche dargestellt sind die Kopplungen einiger weniger Neuronen mit einigen der Sinnesrezeptoren. Ontogenetisch bildet sich durch permanentes Reizen der Sinnesoberfläche eine geordnete Abbildung heraus.

Jedes Neuron an einem Gitterplatz $\vec{r}$ ist durch den Vektor seiner synaptischen Gewichte $\vec{w}_{\vec{r}} \in V$ mit den Inputunits verbunden, vgl. Abb. 8.2

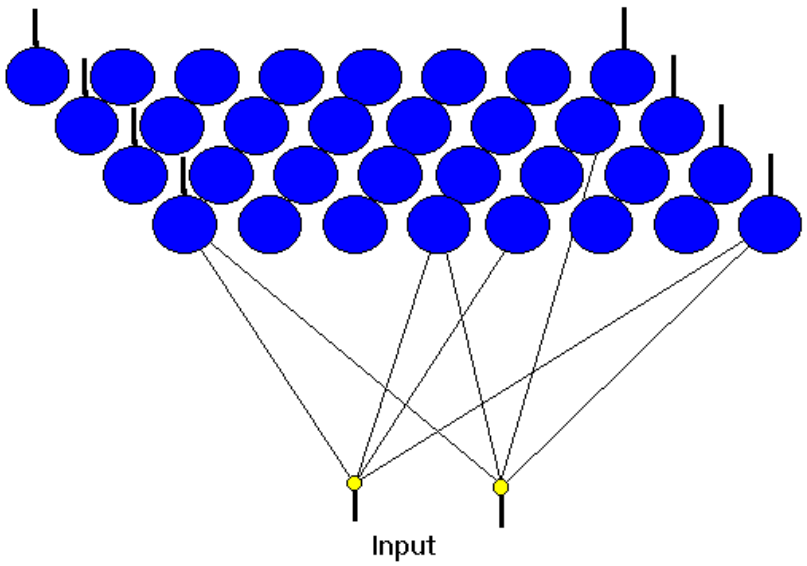


Abbildung 8.2: Für punktförmige Reize kann das in der Abbildung 8.1 dargestellte neuronale Netz vereinfacht werden. Jedes Neuron sieht nun nur zwei Inputs – die Koordinaten des Reizes auf der Sinnesoberfläche. Entsprechend ist sein synaptischer Vektor zweidimensional.

Da der Vektor $\vec{w}_{\vec{r}}$ in V liegt, ist er gleichzeitig die Abbildung des Gitterplatzes $\vec{r}$ nach V, s. Abb. 8.3. Die umgekehrte Abbildung $V \to \mathcal{A}$ ergibt sich durch die Vorschrift

$$\vec{x} \mapsto \vec{s} \quad \text{falls } ||\vec{w}_{\vec{s}} - \vec{x}|| \leq ||\vec{w}_{\vec{r}} - \vec{x}|| \quad \forall \vec{r}$$

oder

$$\vec{s}\,(\vec{x}) = \arg\min_{\vec{r}} ||\vec{w}_{\vec{r}} - \vec{x}|| \tag{8.8}$$

Der Input $\vec{x}$ wird also immer auf den ihm am nächsten gelegenen synaptischen Vektor (SV) $\vec{w}_{\vec{r}}$ abgebildet. Das entspricht der Voronoi–Parkettierung des Inputraumes, s. Abb. 8.3

Um eine **topografische** Abbildung $V \to \mathcal{A}$ zu erhalten, müssen benachbarte sensorische Inputs $\vec{x} \in V$ auf die gleichen oder benachbarte Gitterplätze abgebildet werden. Mit dem von Kohonen angegebenen Lernalgorithmus für die synaptischen Kopplungen bilden sich diese Karten in einem Prozeß der

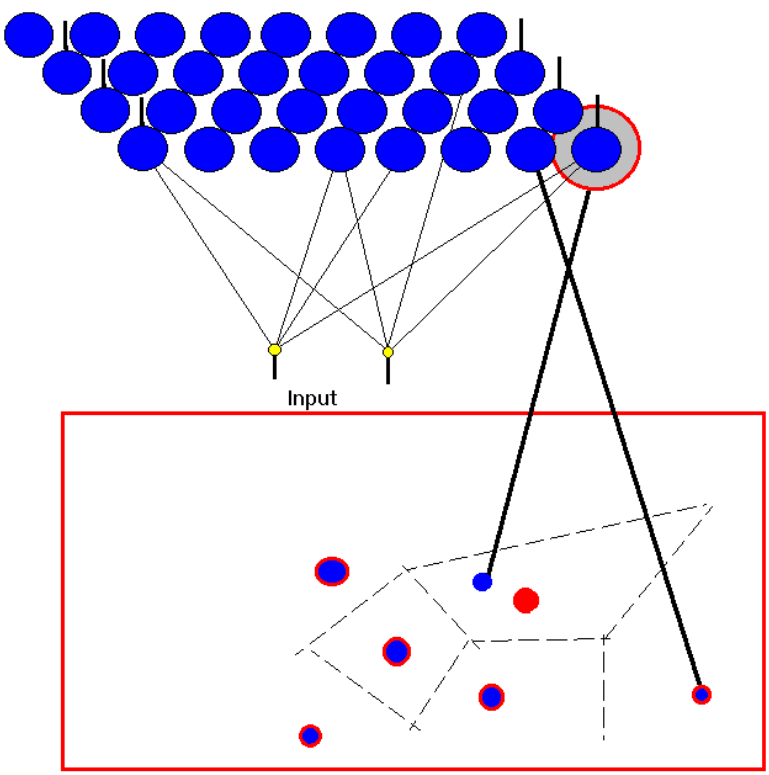


Abbildung 8.3: Für die Situation der Abbildung 8.2 sind für einige Neuronen die synaptischen Vektoren im Inputraum (blaue Punkte) zusammen mit dem Reiz (roter Punkkt) dargestellt. Die synaptischen Vektoren der Neuronen (blaue Punkte) sind die Projektionen der Neuronenorte in den Inputraum. Sie werden auch als die virtuellen Orte der Neuronen bezeichnet. Die Erregungsstärke eines Neurons bestimmt sich aus dem Euklidischen Abstand zwischen Reiz und synaptischem Vektor. Das Neuron, dessen synaptischer Vektor am besten mit dem Input übereinstimmt ist das sog. Siegerneuron (rot umrandet). Das Siegerprinzip definiert gleichzeitig eine Voronoiparkettierung des Inputraumes. Die Voronoizelle eines Neurons ist das Gebiet, dessen innere Punkte genau dieses Neuron zum Sieger machen. Die Grenzen der Voronoizellen einiger Neuronen sind gestrichelt eingezeichnet.

Selbstorganisation heraus, daher der Name selbstorganisierende Karten (englisch self–organizing maps ⇒SOMs)[25, 26]. Der Lernalgorithmus umfaßt die folgenden Schritte:

1. Initialisierung aller synaptischen Gewichte, z. B. $\vec{w} = \vec{w}_{\vec{r}} = random\ \forall \vec{r}$. Dananch werden bei jeder Präsentation eines Inputs $\vec{x}$ folgende Schritte vollzogen:

2. **Siegerbestimmung**: Bestimmung des *best matching* SV's, d. h. finde Gitterplatz $\vec{s}$ (Neuronenort), dessen SV $\vec{w}_{\vec{s}}$ dem präsentierten Input am nächsten ist, d. h.
$$\vec{s} = \arg\min_{\vec{r}} ||\vec{w}_{\vec{r}} - \vec{x}|| \tag{8.9}$$
vgl. 8.8.

3. **Lernschritt**: Dann wird für **alle** SV $\vec{w}_{\vec{r}}$ ein Lernschritt (Adaptionsschritt) durchgeführt:
$$\Delta\vec{w}_{\vec{r}} = \epsilon h_{\vec{r},\vec{s}}\ (\vec{x} - \vec{w}_{\vec{r}}) \tag{8.10}$$
wobei die Nachbarschaftsfunktion
$$h_{\vec{r},\vec{s}} = \exp\left(-\frac{d\,(\vec{r},\vec{s})^2}{2\sigma^2}\right) \tag{8.11}$$
die Mitlernstärke der Neuronen am Ort $\vec{r}$ bei Siegerneuron $\vec{s}$ und $d\,(\vec{r},\vec{s})$ der (Euklidische) Abstand zwischen dem Siegerneuron und dem betrachteten Neuron im Gitter ist, $d\,(\vec{r},\vec{s}) = \sqrt{\sum_{i=1}^{2}(r_i - s_i)^2}$ für das zweidimensionale Gitter.

"Gängige Parameter": $\epsilon = 0.1$ mit langsamer Abkühlung auf Werte $\epsilon < 0.01$. Die Nachbarschaftsreichweite σ wird im Laufe des Lernens von einem anfänglich hohen Wert (so daß die Nachbarschaft etwa ein Drittel aller Neuronen umfaßt) auf einen Wert $\sigma \cong 1$ "abgekühlt". Bei der Abkühlung der Nachbarschaftsreichweite treten eine Reihe interessanter Pänomene wie z. B. Phasenübergänge auf, vgl. [34, 33] und [13, 14].

Der Ordnungsprozeß kommt durch das Mitlernen der Nachbarschaftsneuronen zustande, s. Abb. 8.4.

Die Längenskala für die Umordnung wird dabei durch die Nachbarschaftsreichweite σ bestimmt. Für große σ wird folglich zunächst eine Grobordnung hergestellt, die sich mit der Abkühlung (Verkleinerung) von σ immer weiter verfeinert.

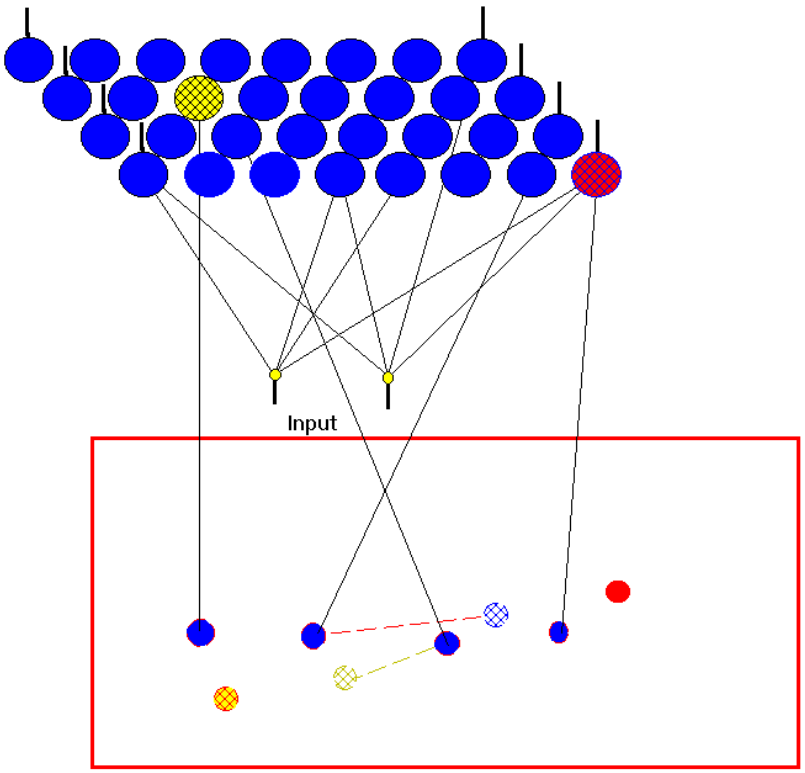


Abbildung 8.4: Der Ordnungsprozeß. Aus der Abbildung wird das selbstorganisierte Entstehen einer **topografischen** Abbildung deutlich. Dargestellt ist die Abfolge zweier Reize (rot bzw. rot schraffiert) und der Lernschritt, den ein ausgewählter Nachbar des jeweiligen Siegers vollzieht. Hierbei findet das "Entfitzen" der sich kreuzenden und damit topologieverletzenden Projektionen der beiden Nachbarneuronen statt.

8.3 Anwendungen der selbstorganisierenden Karten

Die Eigenschat der SOM zur topografischen Abbildung hochdimensionaler Räume auf solche niederer Dimension wurde in einer Vielzahl von Varianten angewendet. Hier sollen nur eingige ganz wenige kurz genannt werden, eine erschöpfende Bibliographie findet sich im Internet, [1].

8.3.1 Datenentrauschung

Messdaten sind oft redundant. Das ist insbesondere eine erwünschte Eigenschaft, wenn die Daten durch Meßfehler korrumpiert, also"verrauscht" sind. Redundanz bedeutet, daß die Datenvektoren in einem Raum höherer als der problemadäquaten Dimension "leben". Die topografische Abbildung des hochdimensionalen auf den eigentlich relevanten, niedrigdimensionalen Raum ist dann geeignet, das Rauschen zu beseitigen.

Wichtig ist dabei, die richtigen Parameter für die Nachbarschaftsreichweite zu finden, s. dazu [10, 11]

8.3.2 Selbstorganisierende Karten und kognitive Prozesse – Ein Modell des '*perceptual magnet*' Effektes

Beim Erlernen von Kategorien wird in wahrnehmungspsychologischen Experimenten ein sog. '*perceptual magnet*' Effekt beobachtet: Die Grenzen zwischen den Kategorien werden wesentlich besser aufgelöst als die eher prototypischen zentralen Regionen. Das wird z. B. bei Säuglingen in der akustischen Wahrnehmung von Vokalen beobachtet. Das Kategorienlernen (Erkennung von Vokalen) kann als die Ausbildung einer Karte verstanden werden. Allerdings haben mit dem Kohonenalgorithmus gelernte Karten eine datendichteproportionale Auflösung. Das bedeutet etwa für das Vokallernen, daß die häufiger stimulierten prototypnahen Muster besser unterschieden werden können als die seltener stimulierten Grenzregionen zwischen den Kategorien.

Trotzdem können selbstorgansierende Karten als Modelle des beobachteten Verhaltens beim Kategorienlernen und damit zur Erklärung des '*perceptual magnet*' Effektes dienen, wenn der Algorithmus von Kohonenen geeignet abgeändert wird. Wir haben in [20, 21, 4] durch eine Aufmerksamkeitssteuerung des Lernprozesses erreicht, daß sich die Auflösung der Karte einstellen läßt,

so daß sie die in den speziellen psychologischen Experimenten beobachteten Verhältnisse widerspiegeln und damit den '*perceptual magnet*' Effekt erklären kann.

8.3.3 Lösung eines Optimierungsproblems – der reisende Handelsmann

Das Problem des reisenden Handelsmannes besteht darin, auf kürzstem Weg eine Rundreise durch N vorgegebe Städte zu organisieren, wobei jede Stadt genau einmal besucht werden soll. Das kann als Abbildung der räumlichen Anordnung der Städte auf eine eindimensionale Kette aus ebenfalls N Gliedern verstanden werden. Diese Abbildung ist eine approximative Lösung des Optimierungsproblemes, wenn sie nachbarschaftserhaltend ist, d. h. wenn sie kurze Abstände zwischen den Städten auf kleine Abstände auf der Kette abbildet.

Die naive Anwendung des Kohonen–Algorithmus bringt aber keine besonders guten Lösungen. Eine sehr effektive Variante ist in [5] bzw. [6] beschrieben.

8.3.4 WEBSOM – Ein automatisches Verfahren zur semantischen Clusterung von Internetdokumenten

Bei diesem Verfahren werden Internetdokumente in jeweils einen Datenvektor umgesetzt. Der Kohonen–Algorithmus realisiert, getrieben durch diese Datenvektoren die Abbildung der Dokumente auf eine zweidimensionale Fläche. Wegen der nachbarschaftserhaltenden Eigenschaft der Kartierung mit dem Kohonen–Algorithmus werden dabei benachbarte Dokumente auch auf benachbarte Regionen der Karte abgebildet. Die Software–Realisierung bietet die Möglichkeit des Zoomens, d. h. wenn man auf der Karte der Dokumente eine interessante Region gefunden hat kann man weiter in die Tiefe gehen und sich eine Karte dieses Ausschnittes auf den Bildschirm holen. Näheres und Demos unter [2].

8.4 Messung der topologischen Eigenschaften der Karten

Wie in Kapitel 8.3 dargestellt, werden die selbstorganisierenden Karten vor allem für die dimensionsreduzierende Abbildung von Datenräumen eingesetzt, zumindest läuft die Mehrzahl der Anwendungen auf dieses Problem hinaus.

Im allgemeinen ist die wahre Dimensionalität der Datenverteilung unbekannt, die Daten bilden eine nichtlineare Mannigfaltigkeit, die außerdem verrauscht ist. Geometrisch heißt das, daß die Datenvektoren im wesentlichen (d. h. bis auf das Rauschen) auf einer gekrümmten Hyperfläche liegen, die in den hochdimensionalen Datenraum eingebettet ist.

Wenn die Dimension d der Hyperfläche bekannt und das Rauschen nicht zu stark ist, so kann die **topographische** Abbildung auf ein d–dimensionales Gitter gelingen. In der Praxis bestimmt man d durch Probieren und wegen des Rauschens kann die Topologieerhaltung nur näherungsweise gesichert werden. Damit ergibt sich die Aufgabe, die Güte der Topologieerhaltung zu messen.

Dafür sind eine Reihe von Verfahren bekannt, die aber alle schlecht für den Fall **nichtlinearer** Datenmannigfaltigkeiten taugen. Eine relativ allgemeine Lösung des Problems wurde in den Arbeiten [43, 41, 40, 42] gefunden und für eine Reihe von Anwendungsfällen getestet.

Literaturverzeichnis

[1] Bibliography on the self-organizing map (som) and learning vector quantization (lvq). http://liinwww.ira.uka.de/bibliography/Neural/SOM.LVQ.html, 1997. 2952 references up to March 1997.

[2] WEBSOM - Self-organizing map for internet exploration. http://websom.hut.fi/, 1997.

[3] N. Alon, A. K. Dewdney, and T. Ott. Efficient simulation of finite automata by neural nets. *Journal of the Association for Computing Machinery*, 38:495 – 514, 1991.

[4] H.-U. Bauer, R. Der, and M. Herrmann. Controlling the magnification factor of self-organizing feature maps. *Neural Computation*, 8(4):757–771, 1996.

[5] M. Budinich. A Self-Organizing neural network for the traveling salesman problem that is competitive with simulated annealing. *Neural Computation*, 8(2):416–424, 1996.

[6] M. Budinich and J. G. Taylor. On the ordering conditions for Self-Organizing Maps. *Neural Computation*, 7(2):284–289, 1995.

[7] E. H. Chudler. Brain facts and figures. http://weber.u.washington.edu/ chudler/facts.html, 1996.

[8] A. Cichocki and R. Unbehauen. *Neural networks for optimization and signal processing*. John Wiley, 1993.

[9] R. Der. Vorlesung Maschinelles Lernen.

[10] R. Der, G. Balzuweit, and M. Herrmann. Building nonlinear data models with self-organizing feature maps. In *Lecture Notes in Computer*

Science 1112: Artificial Neural Networks, Proc. ICANN'96 - Bochum, pages 821 – 826. Springer, 1996. Internet address: http://www.informatik.uni-leipzig.de/ der/Veroeff/bochum.ps.gz.

[11] R. Der, G. Balzuweit, and M. Herrmann. Constructing principal manifolds in sparse data sets by self-organizing maps with self-regulating neighborhood width. In *Proc. ICNN'96 Washington, IEEE*, pages 480 – 483, 1996.

[12] R. Der, H. Englisch, M. Funke, and M. Herrmann. Time series prediction using hierarchical self-organized feature maps. *Neural Networks World*, 3(6):699–704, 1993.

[13] R. Der and M. Herrmann. Critical phenomena in self-organized feature maps: A ginzburg-landau approach. *Phys. Rev. E*, 49(5):5840–5848, 1994.

[14] R. Der and M. Herrmann. Instabilities in self-organized feature maps with short neighborhood range. In *Proceedings of the European Symposium on Artificial Neural Networks (ESANN'94)*, pages 271–276, 1994.

[15] R. Der and U. Steinmetz. Scaling properties of the 'what' and 'where' conundrum (unpublished). Technical report, 1996.

[16] R. Der and U. Steinmetz. Wavelet analysis of EEG signals as a tool for the investigation of the time architecture of cognitive processes. Technical Report 4/97, Institut für Informatik, Universität Leipzig, 1997. Internet address: http://www.informatik.uni-leipzig.de/ der/Veroeff/eegwave.ps.gz.

[17] B. Efron. Bootstrap methods: Another look at the jacknife. *The Annals of Statistics*, 7:1–26, 1979.

[18] J. L. Elman. Finding structure in time. *Cognitive Science*, 14:179 – 211, 1990.

[19] J. L. Elman. *Language as a dynamical system*, chapter 8, pages 195 – 226. MIT, 1995.

[20] M. Herrmann, H.-U. Bauer, and R. Der. The "perceptual magnet" effect: A model based on self-organizing feature maps. In L. S. Smith and P. J. B. Hancock, editors, *Neural Computation and Psychology, Stirling 1994*, pages 107–116, London, UK, 1994. Springer.

[21] M. Herrmann, H.-U. Bauer, and R. Der. öptimal magnification factors in self-organizing feature maps". In *Proc. ICANN'95*, volume 1, pages 75–80, Paris, 1995. EC2 & Cie.

[22] J. Hertz, A. Krogh, and R. Palmer. *Introduction to the Theory of Neural Computation.* Addison Wesley, New York, 1991.

[23] J. Hertz, A. Krogh, and R. Palmer. *Introduction to the Theory of Neural Computation.* Addison Wesley, New York, 1991.

[24] K. Hornik, Stinchcombe, and H. M., White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359 – 366, 1989.

[25] T. Kohonen. *Self–Organization and associative memory Springer Series in Information Science 8*, volume 8 of *Springer Series in Information Science.* Springer, 1984.

[26] T. Kohonen. *The self—organizing map.* Springer, 1995.

[27] Y. Linde, A. Buzo, and R. M. Gray. An algorithm for vector quantizer design. *IEEE Trans. Comm.*, 28:48–95, 1980.

[28] W. S. McCulloch and W. Pitts. A logical calcus of ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133, 1943.

[29] J. G. Nicholis, A. R. Martin, and B. G. Wallace. *From neuron to brain.* Sinauer Associates, Sunderland, Massachusetts, 1992. 3rd edition.

[30] K. R. Popper and J. C. Eccles. *Das Ich und sein Gehirn.* Piper, München, 1982.

[31] R. Raul. *Theorie der neuronalen Netze.* Springer-Verlag, 1993.

[32] R. Raul. *Theorie der neuronalen Netze.* Springer-Verlag, 1993.

[33] H. Ritter, T. Martinetz, and K. Schulten. *Neural Computation and Self-Organizing Maps: An Introduction.* Addison-Wesley, Reading, MA, 1992.

[34] H. J. Ritter, T. M. Martinetz, and K. J. Schulten. *Neuronale Netze: Eine Einführung in die Neuroinformatik selbstorganisierender Abbildungen.* Addison-Wesley, Munich, Germany, 1990.

[35] J. G. Rueckl, K. R. Cave, and S. M. Kosslyn. Why are "what" and "where" processed by separate cortical visual systems? *Journal of Cognitive Neuroscience*, 1:171–186, 1989.

[36] H. T. Siegelman and E. D. Sontag. On the computational power of neural nets. *Journal of Computer and System Sciences*, 50:132 – 150, 1995.

[37] A. Sperduti. On the computational power of recurrent neural networks for structures. *Neural Networks*, 10(3):395 – 400, 1997.

[38] I. G. Sprinkhuisen-Kuyper and E. J. W. Boers. The error surface of the simplest XOR network has only global minima. *Neural Computation*, 8:1301 – 1320, 1996.

[39] G. Thimm, P. Moerland, and E. Fiesler. The interchangeability of learning rate and gain in backpropagation neural networks. *Neural Computation*, 8:451 – 460, 1996.

[40] T. Villmann. *Topologieerhaltung in selbstorganisierenden neuronalen Merkmalskarten – Dissertation Univ. Leipzig 1995.* Harri Deutsch, 1996.

[41] T. Villmann, R. Der, M. Herrmann, and T. Martinetz. Topology preservation in self-organizing feature maps: General definition and efficient measurement. In B. Reuch, editor, *Informatik aktuell – Fuzzy-Logic*, pages 159–166, Berlin, 1994. Springer.

[42] T. Villmann, R. Der, M. Herrmann, and T. M. Martinetz. Topology preservation in self-organizing feature maps: Exact definition and measurement. *IEEE Transactions on Neural Networks*, 8(2):256–266, Mar. 1997.

[43] T. Villmann, R. Der, and T. Martinetz. A novel approach to measure the topology preservation of feature maps. In M. Marinaro and P. G. Morasso, editors, *Proc. ICANN'94, Int. Conf. on Artificial Neural Networks*, volume I, pages 298–301, London, UK, 1994. Springer.

[44] J. Wiles. Using bottlenecks in feed—forward networks as a dimension reduction technique: An application to optimization tasks. *Neural Computation*, 8:1179—1183, 1996.

[45] A. Zell. *Simulation neuronaler Netze.* Addison–Wesley, 1994.

[46] A. Zell. SNNS - Der Stuttgarter Neuronaler Netz Simulator. http://www.informatik.uni-stuttgart.de/ipvr/bv/projekte/snns/snns.html, 1996.