Inhalt

Neuronale Netze

von Reginald Ferber

1.1. Das Nervensystem als Vorbild paralleler Informationsverarbeitung

Betrachtet man die Leistungsfähigkeit moderner Computer, so fällt auf, daß in vielen Bereichen heute zwar Berechnungen möglich sind, die noch vor wenigen Jahrzehnten im Bereich der Utopie lagen, daß aber andererseits für alltägliche Informationsverarbeitungsaufgaben, die von Menschen (und Tieren) problemlos, schnell und oft sogar unbewußt gelöst werden, kaum Lösungen auf Rechnern existieren. Die, die existieren, sind oft sehr aufwendig, langsam und anfällig und funktionieren nur unter günstigen Bedingungen. Typische Beispiele finden sich im Bereich der Wahrnehmung: Aufgaben der Bilderkennung wie Herauslösen von Motiven aus unruhigem Hintergrund, Erkennen von Gegenständen bei verschiedener Beleuchtung, Erkennen von teilweise verdeckten Gegenständen. Besonders schwierig wird es, wenn mehrere dieser Probleme zusammen auftreten. Aber auch das Kategorisieren von wahrgenommenen Motiven, z.B. das Lesen von Handschriften, ist eine sehr schwierige Aufgabe für maschinelle Lösungen.

Wenn Bilder als Pixelmuster mit Rechnern verarbeitet werden, zeichnen sie sich durch ungeheure Datenmengen aus. Wenn diese Daten sequentiell, wie auf herkömmlichen sog. VON NEUMANN—Maschinen[1], verarbeitet werden, ergeben sich auch bei schnellen Rechnern enorme Rechenzeiten.

Die Geschwindigkeit der Signalverarbeitung in den einzelnen Zellen des Nervensystems, den Neuronen, ist im Vergleich zur Geschwindigkeit von Bauelementen moderner Rechner langsam. Trotzdem sind Organismen in der Lage, enorm schnell auf Sinneswahrnehmungen zu reagieren. Diese Geschwindigkeit beruht vermutlich zu einem großen Teil auf der parallelen Verarbeitung der einströmenden Information.

Daher versucht man Algorithmen zu entwickeln, die, von kleinen Teilbereichen der wahrgenommenen Information ausgehend, Daten parallel verarbeiten und die so gewonnenen Ergebnisse auf verschiedenen Hierarchiestufen integrieren.

In der Natur scheint es ähnliche Strukturen zu geben. Bei der Untersuchung des visuellen Systems von Katzen fand man auf der Hirnrinde in den entsprechenden Bereichen Zellen, die besonders stark erregt waren, wenn dem Versuchstier visuelle Stimuli in Form von gerichteten Geradenstücken dargeboten wurden. Dabei gab es für verschiedene Richtungen verschiedene Zellen, die am stärksten erregt waren. Da für die Wahrnehmung der Richtung mehrere Zellen der Retina benötigt werden, muß im Verlauf der Reizweiterleitung vom Auge zur Hirnrinde bereits eine Verarbeitung der Information stattgefunden haben, bei der die Geradenstücke erkannt wurden. (Eine Übersicht zu diesem Thema findet sich in [12].)

---

1) Die Namensgebung sollte nicht zu der Annahme verleiten, VON NEUMANN's Werk beschränke sich auf sequentielle Rechner. Neben vielen Ideen und Ergebnissen aus vielen anderen Bereichen stammt von ihm auch eine Pionierarbeit aus dem Bereich der parallelen Informationsverarbeitung. (Siehe [11])

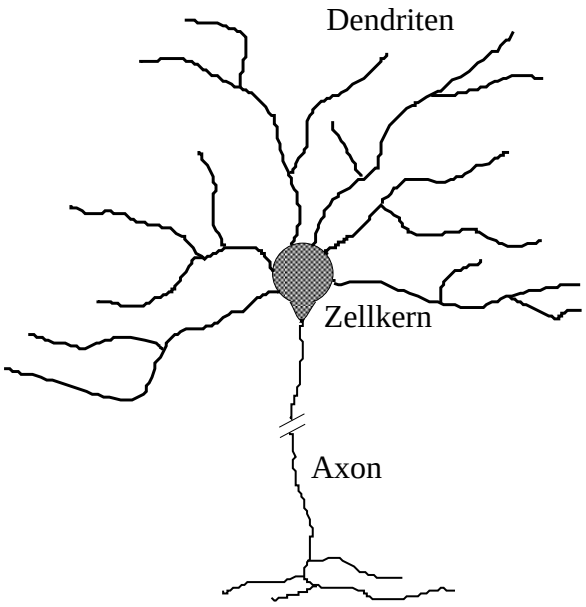

Abbildung 1.1.1. Schematisierte Darstellung einer Nervenzelle des Großhirns. An den Dendriten und dem Zellkörper befinden sich Synapsen zu anderen Zellen. Das sind Kontaktpunkte, an denen u.a. durch chemische Prozesse positive und negative elektrische Potentiale an die Zelle übergeben werden. Summieren sich in einem Zeitraum diese Potentiale zu einem genügend großen Potential, so löst dieses am Zellkörper einen oder mehrere sogenannte Spikes aus. Das sind relative elektrische Potentiale, die sich durch Depolarisierung der Zellwände entlang dem Axon (teilweise über weite Strecken) fortsetzen und ihrerseits wieder anderen Zellen elektrische Potentiale über Synapsen zuleiten. Wichtig für uns ist, daß die Weiterleitung im wesentlichen nicht als einfache elektrische Leitung geschieht, sondern durch die Fortpflanzung des Depolarisierungseffektes quasi Information weitergeleitet wird. Insgesamt ergeben sich drei wesentliche Punkte: 1. Ankommende Spikes können durch Synapsen je nach deren Art aktivierend oder hemmend wirken. 2. Die Potentiale, die eine Zelle erreichen, müssen zusammen eine Schwelle überschreiten, um weitergeleitet zu werden. 3. Die Weiterleitung von Erregung an andere Zellen verringert den Spike nicht. D.h. es wird nicht ein vorhandenes Potential aufgeteilt, sondern Information weitergegeben, ohne dadurch verbraucht zu werden.

Weitere Bereiche der Informationsverarbeitung, in denen Organismen künstlichen Systemen weit überlegen sind, sind die Verarbeitung von unsicherer, unvollständiger, zweideutiger, fehlerhafter oder widersprüchlicher Information und die Filterung von (überlebens-)wichtiger Information aus dem enormen Informationsfluß, der über die Sinnesorgane auf den Organismus einströmt. Dabei ist es entscheidend, auf frühere Erfahrungen zurückgreifen zu können. Entsprechend müssen aktuelle Erlebnisse als Erfahrungen aufgenommen werden. Der Begriff “Erfahrung” deutet hier an, daß es sich dabei um eine bisher schwer zu beschreibende Form des Gedächtnisses handelt. Es werden viele einzelne Situationen und Erlebnisse zu einem Gesamtsystem vereint, aufgrund dessen Entscheidungen gefällt werden. Dabei sind die einzelnen Erlebnisse nicht als solche präsent, lassen sich aber eventuell rekonstruieren. Das sogenannte Weltwissen, das wir im täglichen Leben brauchen, scheint in eher vager Form gespeichert zu sein. Aber gerade diese Vagheit scheint die Flexibilität, die das Verhalten in unserer komplexen Umwelt verlangt, erst zu ermöglichen.

Seit Mitte dieses Jahrhunderts werden parallele Modelle und Algorithmen entwickelt, die sich z.T. an der Struktur des Nervensystems orientieren. Dabei gab es Beiträge aus vielen

Disziplinen, vor allem aus der Medizin (Neurophysiologie) [12], Psychologie [16], [8], Biologie, Mathematik [14], [1], Physik, Informatik [5] und Chemie. Entsprechend vielfältig sind auch die Bezeichnungen, Definitionen und Methoden. Es handelt sich oft um mehr oder weniger ad hoc verwendete Methoden aus anderen Bereichen. Häufige Bezeichnungen für Theorien und Modelle aus diesem Bereich sind unter anderen: Konnektionismus, parallele verteilte Systeme (parallel distributed processing = pdp), zellulare Automaten, homogene Strukturen, iterative Arrays. Von einer geschlossenen Begriffsbildung oder Theorie kann kaum die Rede sein. Das macht die Beschäftigung mit dem Thema andererseits auch wieder spannend.

Es können hier nun nicht alle Ansätze erläutert werden. Vielmehr habe ich versucht, zunächst in 1.2 einen globalen Rahmen abzustecken, in den sich fast alle Modelle einordnen lassen. In 1.3 werden die häufigsten Merkmale, nach denen neuronale Netze unterschieden werden, vorgestellt.

In Teil 2 werden einzelne Modelle vorgestellt. Es wurden Modelle ausgewählt, die jeweils für eine ganze Gruppe von Ansätzen charakteristisch sind und an denen sich die grundlegenden Mechanismen darstellen lassen. Dabei habe ich teilweise versucht, durch verhältnismäßig genaue Darstellung von Sätzen und Beweisen einen Einblick in die verwendeten Methoden zu ermöglichen.

Zunächst wird in 2.1 auf einen der historisch grundlegenden Artikel hingewiesen. Darin wurde gezeigt, daß mit neuronalen Netzen jede logische Funktion modelliert werden kann.

2.2 beschreibt das Perceptron, ein Modell aus der Mustererkennung, dessen einfache Struktur eine weitgehende mathematische Analyse seiner Möglichkeiten und Grenzen zuläßt. Es ist gleichzeitig für die erste Phase der Forschung auf dem Gebiet der neuronalen Netze charakteristisch.

2.3 behandelt HOPFIELD Netze, ein von der Physik inspiriertes Modell eines Assoziativspeichers, dessen iterative Dynamik durch eine monoton fallende Energiefunktion gekennzeichnet ist.

In 2.4 wird der Backpropagation Algorithmus vorgestellt. Dabei handelt es sich um eine Weiterentwicklung des Perceptrons.

1.2. Definition zellularer und neuronaler Strukturen

In diesem Abschnitt sollen allgemeine Definitionen gegeben werden, die die meisten neuronalen Modelle einschließen. Sie finden sich in dieser expliziten Form selten in der Literatur, scheinen mir aber nützlich, um Gemeinsamkeiten und Unterschiede neuronaler Modelle herauszuarbeiten.

1.2.1. Definition: Zellulare Struktur

Sei W eine beliebige und I eine abzählbare Menge. Eine Abbildung $c : I \to W$ bezeichnen wir als *Konfiguration* von *Werten* aus W auf den *Zellen* aus I . $C = W^I = \{c : I \to W\}$ bezeichnet die Menge aller Konfigurationen.

Zu jeder Zelle $i \in I$ gebe es eine endliche, geordnete Teilmenge $N(i) \subset I$, die als Tupel der *Nachbarn* oder *Nachbarschaft* der Zelle i bezeichnet wird. $N = \{N(i) \mid i \in I\}$ bezeichne die Menge aller dieser Nachbarschaften.

Ferner existiere zu jedem $i \in I$ eine *lokale Funktion* $f_i : W^{N(i)} \to W$, die jeder Konfiguration von Werten auf den Zellen der Nachbarschaft von i einen neuen Wert zuweist. $f = \{f_i \mid i \in I\}$ bezeichne die Menge dieser lokalen Funktionen.

Das Tupel $Z = (I, W, N, f)$ wird als *zellulare Struktur* bezeichnet.

Ist I endlich, heißt $Z = (I, W, N, f)$ *endliche zellulare Struktur*.

Ist W endlich, heißt $Z = (I, W, N, f)$ *zellularer Automat*.

Durch gleichzeitige Anwendung der lokalen Funktionen in allen Zellen erhalten wir eine *globale Funktion* $F : C \to C$ mit $F(c)(i) := f_i(c(N(i)))$.[2]

Durch die Nachbarschaften der Zellen läßt sich ein Graph oder Netz definieren: Wir wählen die Zellen als Knoten des Graphen und verbinden jede Zelle mit ihren Nachbarn durch gerichtete Kanten. Die Richtung einer Kante geht dabei vom Nachbarn zur Zelle.

Falls I strukturiert ist, also z.B. $I = \mathbb{Z}$, $I = \mathbb{Z}^2$oder $I = \mathbb{Z}_N$ gilt, können die Nachbarschaften mit Hilfe der Struktur definiert werden. So ergibt $I = \mathbb{Z}^2$ mit $N(i) = (i, i+(0,1), i-(0,1), i+(1,0), i-(1,0))$ ein Rechteckgitter in der Ebene mit der sogenannten VON NEUMANN Umgebung (Abb. 1.2.2.b). Für $I = \mathbb{Z}_N$ mit $N(i) = (i+1 \bmod N, i-1 \bmod N)$ erhält man einen Ring mit je zwei Nachbarn pro Zelle (Abb. 1.2.2.c). Sind in diesen Fällen die lokalen Funktionen alle gleich definiert, so spricht man von homogenen zellularen Strukturen.

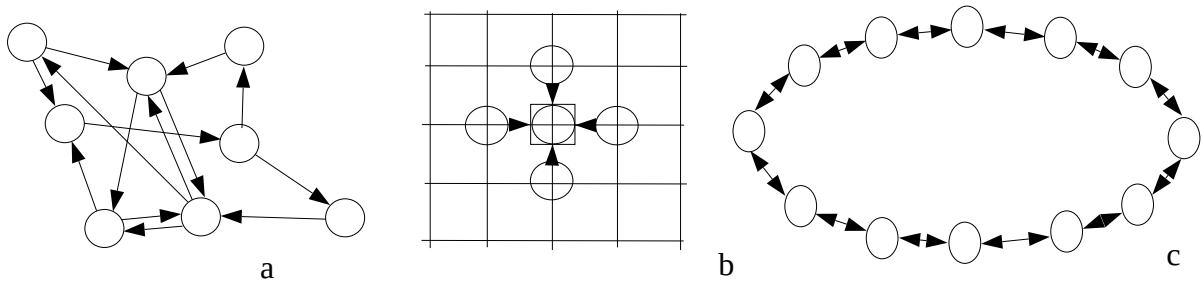

Abbildung 1.2.2. Verschiedene Netze zellularer Strukturen. a) zeigt eine beliebige Struktur (Die Pfeile deuten die Nachbarn an), b) ein Rechteckgitter mit VON NEUMANN Nachbarschaft (mit den Nachbarn der zentralen Zelle), c) ein ringförmiges Netz einer zellularen Struktur.

Nach der Definition der zellularen Struktur sind fast beliebige Netze von Zellen möglich. Wir wollen im folgenden eine Unterklasse von Netzen mit strengeren Strukturen abgrenzen:

1.2.2. Definiton: Feed forward Netz

In dem Graph, der durch die Nachbarschaftsbeziehungen N einer zellularen Struktur $Z = (I, W, N, f)$ gegeben ist, sagen wir, eine Zelle i ist in n Schritten von Zelle j aus erreichbar, wenn es Zellen $k_1, ..., k_{n-1}$ und Kanten von j nach k_1, von k_{m-1} nach k_m für $m = 2, ..., n-1$ und von k_{n-1} nach i gibt.

Die Menge $E = \{i \in I \mid N(i) = \emptyset\}$ bezeichnen wir als Menge der *Eingabezellen*.

---

2) $c(N(i))$ wird gemäß der folgenden Notation gebildet: Sei $N(i) := (j_1, ..., j_n)$ dann ist $c(N(i)) := (c(j_1), ..., c(j_n))$. Etwas verkürzend benutzen wir $N(i)$ auch als Bezeichnung für die Menge der Nachbarn und lassen dabei die Ordnung unberücksichtigt.

Sind die Mengen $S_n = \{i \in I \mid \exists e \in E$ mit i ist in n Schritten von e aus erreichbar$\}$ disjunkt und nicht leer, so nennen wir sie *Schichten* des Netzes und sprechen von einem $n+1$*–schichtigen feed forward Netz*. Die Eingabezellen bilden die nullte Schicht S_0, die Menge der Zellen, die Eingabezellen als Nachbarn haben, die erste S_1 usw. Die letzte Schicht S_n wird auch *Ausgabeschicht* genannt.

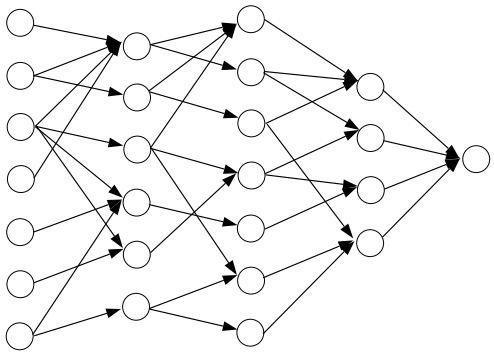

Abbildung 1.2.3. Ein feed forward Netz mit 5 Schichten.

Neben der Struktur des zugrundeliegenden Netzes können auch Eigenschaften anderer Komponenten der zellularen Struktur zur Charakterisierung benutzt werden. Ist beispielsweise W eine Gruppe und sind die lokalen Funktionen Homomorphismen, so ist die globale Funktion ein Homomorphismus der Gruppe der Konfigurationen. Zur ihrer Untersuchung steht damit die Theorie der Gruppenhomomorphismen zu Verfügung.
Faßt man den Raum der Konfigurationen mit endlicher Wertemenge W als Produkt $C = \prod_{i \in I} W$ auf, ausgestattet mit der Produkttopologie der diskreten Topologie auf W, so gilt: Eine Selbstabbildung $F : C \to C$ ist genau dann stetig, wenn sie globale Funktion eines zellularen Automaten ist (Siehe [2]).
Neuronale Netze sind zellulare Strukturen mit speziellen lokalen Funktionen, die in Analogie zum Verhalten von natürlichen Neuronen gewählt werden.

1.2.3. Definition: Neuronales Netz

1. Eine zellulare Struktur $N = (I, W, N, f)$ mit $W \subset \mathbb{R}$ und

$$f_i(c(N(i))) = \xi_i \left(\sum_{j \in N(i)} w_{i,j} c(j) \right) \text{ mit } w_{i,j} \in \mathbb{R} \tag{1.2.1}$$

und monoton wachsenden Funktionen $\xi_i : \mathbb{R} \to W$ heißt *(deterministisches) neuronales Netz*. Die Funktion ξ_i heißt *Outputfunktion* der Zelle i.
$w_{i,j}$ wird als *Gewicht* von Zelle j nach Zelle i bezeichnet.

2. Eine Funktion der Form

$$\sigma_\theta : \mathbb{R}^n \to \{0,1\};\ \sigma_\theta(x) = \zeta_\theta \left(\sum_{i=1}^{n} w_i x_i \right) = \begin{cases} 1 & falls \ \sum_{i=1}^{n} w_i x_i - \theta \geq 0 \\ 0 & sonst \end{cases}$$

heißt *lineare Schwellwertfunktion* mit Gewichten $w_1, ..., w_n$ und Schwellwert θ. Sie ist durch das $(n+1)$–Tupel $w = (w_1, ..., w_n, -\theta) \in \mathbb{R}^{n+1}$ vollständig charakterisiert.

Eine lokale Funktion einer Zelle eines neuronalen Netzes bildet also die gewichtete Summe der Werte ihrer Nachbarzellen. Auf diese Summe wird die Outputfunktion ξ_i angewendet, um den neuen Wert der Zelle zu berechnen. Die Bezeichnung "linear" in 1.2.3.2 bezieht sich auf die Summation der Werte der Nachbarzellen, die Bezeichnung "Schwellwert" auf die Sprungfunktion ζ_θ.

Für eine Zelle mit linearer Schwellwertfunktion als lokaler Funktion gilt: Überschreitet die Summe der Werte der Nachbarzellen einen Grenzwert, so nimmt die Zelle den Wert 1 (aktiv) an, anderenfalls den Wert 0 (passiv). (In manchen Modellen werden auch die Werte 1 und -1 gewählt). Die Analogie zum Verhalten von natürlichen Nervenzellen (Siehe Abbildung 1.1.1) besteht aus dem diskreten An/Aus Verhalten in positiver oder negativer Abhängigkeit von den von außen einwirkenden Impulsen.

Man kann die Gewichte $w_{i,j}$, $j = 1, ..., \#N(i)$[3] als Parameter der lokalen Funktion f_i der Zelle i eines neuronalen Netzes ansehen. Wenn I endlich ist und $N(i) = I$ gilt, ist es aber auch üblich, sie als Zeilen einer Matrix zu speichern. Man spricht dann von symmetrischen Gewichten, wenn die Matrix symmetrisch ist, wenn also gilt: $w_{i,j} = w_{j,i}\ \forall i, j \in \{1, ..., n\}$. Wählt man im Fall $N(i) = I$ die Identität als Funktion ξ_i in 1.2.1, erhält man als globale Funktion gerade die lineare Abbildung, die durch die Matrix gegeben ist. Die Konfiguration ist ja gerade ein Vektor, und die Summation entspricht der Multiplikation einer Zeile der Matrix mit diesem Vektor.

Eine weitere Möglichkeit, die Gewichte der linearen Schwellwertfunktion in die Definition eines neuronalen Netzes einzubinden, besteht darin, die Elemente in W so aufzubauen, daß sie die Gewichte enthalten, z.B. in der Form $W = \{0, 1\} \times \mathbb{R}^{n+1}$. Dann müssen die lokalen Funktionen entsprechend definiert werden. Diese Form ist formal komplizierter, ermöglicht dafür aber die Veränderung der Gewichte durch die lokalen Funktionen, ein Effekt, der bei kontinuierlich lernenden Systemen erwünscht ist. Ich werde darauf aber hier nicht weiter eingehen.

Solche Überlegungen sind allerding mehr formaler Art. In der Praxis werden die Algorithmen meist ad hoc definiert, ohne sie in einen theoretischen Rahmen einzupassen. (Der auch nur dann nützlich ist, wenn er die Anwendung von bekannten Ergebnissen ermöglicht oder zur Klärung der Struktur beiträgt.)

1.3. Typisierung neuronaler Netze

Für eine Vielzahl von Aufgaben werden Lösungsansätze mit neuronalen Netzen untersucht. Es lassen sich verschiedene Aufgabentypen unterscheiden:

— Assoziative Speicher (Assotiative memory): In einem System werden Bitmuster gespeichert. Wird dem System ein Teil eines Musters dargeboten, soll es in der Lage sein, das gesamte Muster zu reproduzieren.
— Inhaltsadressierbare Speicher (Content adressable memory): Auf einen dargebotenen Stimulus soll das System in einer bestimmten Weise reagieren. (z. B. die Vergangenheits-

[3] $\#X$ bezeichnet die Mächtigkeit der Menge X

form eines regelmäßigen oder unregelmäßigen englischen Verbs bilden (Siehe [8] Kapitel 18)).

— Kategorisierung: Dargebotene Bitmuster sollen in Klassen eingeteilt werden. (z. B. sollen Pixelmuster als Buchstaben erkannt werden).

Bei diesen Aufgaben soll auch eine gewisse Fehlertoleranz gelten. Sind die dargebotenen Stimuli leicht verändert oder "verrauscht", soll trotzdem noch das "richtige" Muster produziert werden. Dabei vermischen sich die Aufgaben des Kategorisierens und des assoziativen Speichers.

Ein wichtiges Merkmal neuronaler Netze ist, wieder in Analogie zu biologischen Systemen, der Versuch, das System aus Beispielen lernen zu lassen. Dieses "Wissen" wird in den Gewichten "gespeichert". Man unterscheidet zwischen zwei Arten des Lernens:

— Überwachtes Lernen (Supervised learning): Das System soll Stimulus-Response-Paare lernen. Dazu werden in der Regel zunächst in einem separaten Verfahren Gewichte aus den Stimulus-Response-Paaren berechnet, die dann in der "Arbeitsphase" verwendet werden.

— Unüberwachtes Lernen (Unsupervised learning): In diesem Fall werden dem System nur Stimuli dargeboten. Aus ihrer Häufigkeit, ihrem gemeinsamen Auftreten und ihrer Ähnlichkeit untereinander werden Informationen gewonnen. Diese Art des Lernens ist mehr an psychologischen und physiologischen Theorien orientiert. So wird in der Assoziationspsychologie angenommen, daß Dinge, die häufig gemeinsam auftreten, durch starke Assoziationen verbunden sind, d.h. wenn eines auftritt, wird das andere "erinnert". Auf der neuronalen Ebene entspricht dem die HEBB'sche Regel: 'When an axon of a neuron x is near enough to help fire a neuron y and does so, some change takes place such that x becomes more effective at exciting y.' (D. O. HEBB, 1958, zitiert nach [12] Seite 92). Diese Art Lernen findet typischerweise nicht in einer separaten Phase statt, sondern das System wird den Stimuli ausgesetzt, im Laufe der Zeit werden die Gewichte verändert, und dadurch verändert sich auch das Verhalten des Systems. Typische Beispiele sind KOHONEN Modelle (Siehe z.B. [13], [6]).

2. Modelle und Beispiele

In diesem Kapitel werden Ergebnisse und Modelle in der Reihenfolge ihrer historischen Entwicklung vorgestellt. Die Modelle stehen als Prototypen für jeweils eine Vielzahl von ähnlichen Modellen.

2.1. Schwellwertlogik

1943 zeigten WARREN S. McCULLOCH und WALTER PITTS [9], daß mit Schwellwertfunktionen jede logische Funktion modelliert werden kann. Dazu genügt es zu zeigen, daß die logischen Funktionen AND und NOT mit Schwellwertfunktionen modelliert und verknüpft werden können. Die Schwellwertfunktionen σ sind als Tabellen (2.1.4) und (2.1.5) angegeben, wobei 0 für den Wahrheitswert falsch (false) und 1 für wahr (true) steht. (Wir folgen in diesem Abschnitt [12])

x	wx	θ	$\sigma(x)$
0	0	$-\frac{1}{2}$	1
1	-1	$-\frac{1}{2}$	0

Abbildung 2.1.4. Darstellung der logischen Funktion NOT $\left(-1,+\frac{1}{2}\right)$ als Schwellwertfunktion.

x_1, x_2	$\sum_{i=1}^{2} w_i x_i$	θ	$\sigma(x_1, x_2)$
$0,0$	0	$\frac{3}{2}$	0
$0,1$	1	$\frac{3}{2}$	0
$1,0$	1	$\frac{3}{2}$	0
$1,1$	2	$\frac{3}{2}$	1

x_1, x_2	$\sum_{i=1}^{2} w_i x_i$	θ	$\sigma(x_1, x_2)$
$0,0$	0	$\frac{1}{2}$	0
$0,1$	1	$\frac{1}{2}$	1
$1,0$	1	$\frac{1}{2}$	1
$1,1$	2	$\frac{1}{2}$	1

Abbildung 2.1.5. Darstellung der logischen Funktionen AND $\left(1,1,-\frac{3}{2}\right)$ (links) und OR $\left(1,1,-\frac{1}{2}\right)$ (rechts) als Schwellwertfunktionen.

Die logische Funktion XOR läßt sich nicht mehr durch eine einzelne Schwellwertfunktion modellieren, aber sie läßt sich folgendermaßen bilden: $x \operatorname{xor} y = (x \text{ and not } y)$ or $(y \text{ and not } x)$. Diese Funktion ist in Abbildung 2.1.6.a als neuronales Netz mit Schwellwertfunktionen dargestellt. Sie läßt sich noch vereinfachen. (Siehe 2.1.6.b.)

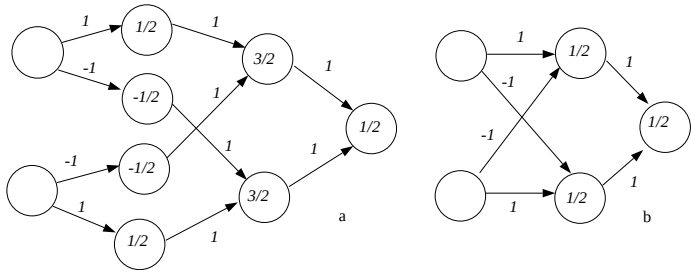

Abbildung 2.1.6. Die logische Funktion XOR läßt sich durch Verknüpfung von NOT, AND und OR darstellen (a) oder auch einfacher mit drei Schwellwertfunktionen modellieren. Die Kreise stehen für Zellen, die Pfeile für Verbindungen. Die Zahlen an den Pfeilen geben die Gewichte an, die in den Kreisen die Schwelle der Zelle. (Darstellung nach [8])

Die Vollständigkeit der Schwellwertlogik ist vor allem ein theoretisch interessantes Ergebnis. Sie besagt, daß alles, was mit einem herkömmlichen Computer berechnet werden kann, auch mit einem neuronalen Netz berechnet werden kann. Sie spielt im Bereich der neuronalen Netze kaum eine praktische Rolle, da diese ja eingesetzt werden, um Phänomene zu modellieren, die mit herkömmlichen Computern schwer zu bearbeiten sind.

2.2. Das Perceptron

Das (einfache) *Perceptron* wurde 1962 von FRANK ROSENBLATT veröffentlicht [15]. Die folgende Darstellung des Perceptrons orientiert sich im wesentlichen an [10] und [1]. Es ist

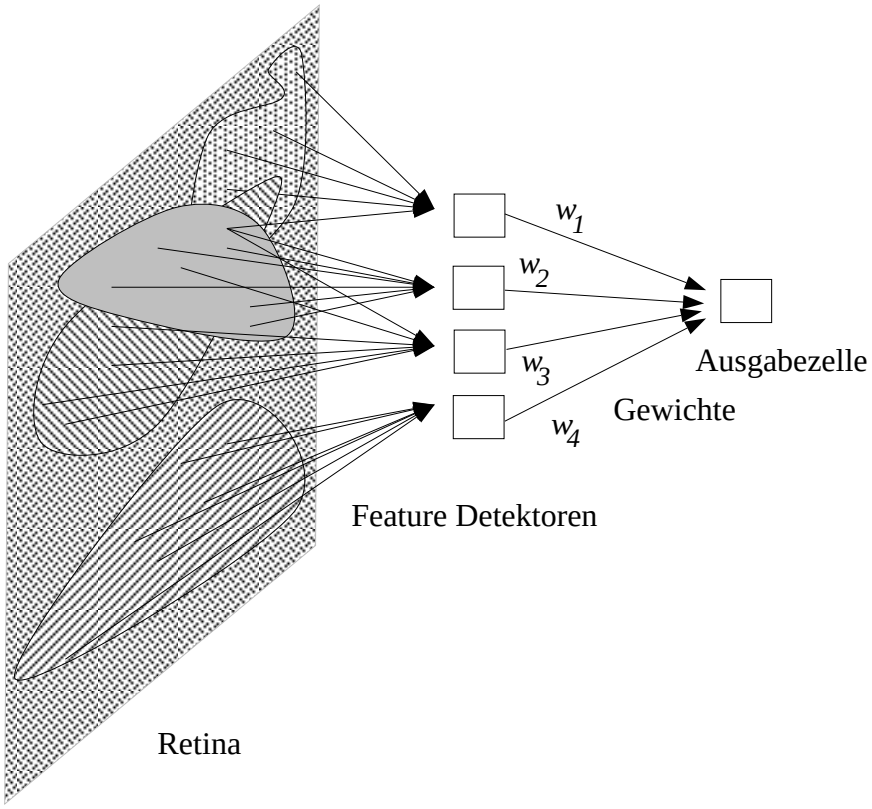

Abbildung 2.2.7. Das Perceptron. (Abbildung nach [10]). Beim Lernen werden nur die Gewichte zur Ausgabezelle hin verändert. Die Feature Detektoren bleiben unverändert.

ein feed forward Netz mit drei Schichten zur Klassifizierung von Mustern. (Siehe Abbildung 2.2.7). In der Eingabeschicht S_0, die in Anlehnung an das Auge auch *Retina* genannt wird, werden *Pixelmuster* angelegt. Die mittlere Schicht S_1 besteht aus *Feature Detektoren*. Diese Zellen haben Schwellwertfunktionen als lokale Funktionen, die Muster in einem Bereich der Retina erkennen, d. h. beim Vorliegen eines entsprechenden Musters den Wert 1 annehmen und sonst 0 sind. Diese Feature Detektoren wiederum liefern die Eingabewerte für eine Ausgabezelle a , die einzige Zelle der Ausgabeschicht S_2 .

Es werden zwei Mengen von Mustern vorgegeben. Durch einen *Lernalgorithmus* soll erreicht werden, daß die Ausgabezelle immer dann aktiv ist (d. h. den Wert 1 hat), wenn ein Muster aus der einen Menge anliegt, und immer dann inaktiv ist (d. h. den Wert 0 hat), wenn ein Muster aus der anderen Menge anliegt. Wir werden sehen, daß das genau dann erreichbar ist, wenn die beiden Mengen linear separabel sind, d. h. wenn es im Vektorraum, der durch die Werte der Feature Detektoren aufgespannt wird, eine Hyperebene gibt, die die zwei Mengen trennt.

Der zweite Aspekt, mit dem wir uns näher beschäftigen werden, ist die Frage, wie weit Parallelisierung mit dem Perceptron möglich ist. Als Maß dafür wird die maximale Anzahl der Eingaben für einen Feature Detektor genommen, die nötig ist, um eine Aufgabe zu

lösen. Ein berühmtes Ergebnis aus [10] besagt, daß, um festzustellen, ob die Anzahl der Retinapunkte, die aktiv sind, gerade oder ungerade ist, mindestens ein Feature Detektor alle Retinapunkte als Eingabe haben muß. D. h. es kann für diese Aufgabe mit dem Perceptron nicht effektiv parallelisiert werden. Dieses Ergebnis war einer der Gründe, warum Ende der 60er Jahre die Forschung auf diesem Gebiet entmutigt wurde. Dabei sagt dieses Ergebnis nichts darüber aus, ob zum Beispiel mit einem mehrschichtigen Perceptron solche Verfahren realisiert werden können. (Darauf weisen die Autoren in [10] übrigens ausdrücklich hin.) Außerdem scheint mir die Frage, ob die Anzahl aktiver Bildpunkte gerade ist, oft irrelevant. Sie wird niemanden bei einem gerasterten Fernsehbild interessieren, wohl aber die Frage, ob der Inhalt des Bildes "erkannt" werden kann.

Zunächst geben wir eine formale Definition des Perceptrons im Rahmen von 1.2.1:

2.2.1. Definition: Perceptron

Ein endliches dreischichtiges neuronales feed forward Netz $P = (I, W, N, f)$ mit einer Zerlegung in Schichten $R = S_0 \subset I$; $D = S_1 \subset I$; $\{a\} = S_2 \subset I$, $W = \{0, 1\}$ und linearen Schwellwertfunktionen als lokalen Funktionen der Zellen aus $D \cup \{a\}$ heißt (einfaches) *Perceptron*.

R wird als *Retina* bezeichnet, die Zellen in D heißen *Feature Detektoren*, a heißt *Ausgabezelle*.

Als *Eingabemuster* bezeichnen wir Konfigurationen, die auf den Zellen aus $D \cup \{a\}$ gleich 0 sind.

Ein Eingabemuster e legt das Pixelmuster eines Bildes auf der Retina fest. Wegen der feed forward Struktur des Netzes stellen die Werte der Feature Detektoren in der Konfiguration $F(e)$ den ersten Verarbeitungsschritt des Bildes dar. Der Wert der Ausgabezelle a in der Konfiguration $F^2(e) = F \circ F(e)$ liefert das Ergebnis des Erkennungsprozesses.

Um den Lernalgorithmus vorzustellen, benötigen wir einige Definitionen:

2.2.2. Bezeichnungen:

Die Symbole $\lceil . \rceil$ verwandeln einen logischen Wert in Zahlen: $\lceil true \rceil := 1$; $\lceil false \rceil := 0$

Sei $d = \#D$ die Anzahl der Feature Detektoren $j_1, ..., j_d$ eines Perceptrons und seien $w_{a,1}, ..., w_{a,d}$ die Gewichte zwischen den Feature Detektoren und der Ausgabezelle a. Dann ist die lineare Schwellwertfunktion der Ausgabezelle a durch $w := \left(w_{a,1}, ..., w_{a,d}, w_{a,d+1}\right) \in \mathbb{R}^{d+1}$ mit $w_{a,d+1} := -\theta_a$ charakterisiert. Bezeichne $x := (c(j_1), ..., c(j_d))$ den Vektor der Werte der Feature Detektor Zellen, so bezeichnen wir mit $\widetilde{x} := (c(j_1), ..., c(j_d), 1) \in \mathbb{R}^d \times \{1\} =: X^{d+1} \subset \mathbb{R}^{d+1}$ den um eine Komponente 1 verlängerten Vektor. Durch diese Verlängerung wird der Schwellwert durch eine zusätzliche Zelle mit dem konstanten Wert 1 simuliert. Auf diese Weise werden wir im folgenden häufiger annehmen, daß der Schwellwert gleich 0 ist. Die Schwellwertfunktion der Ausgabezelle a kann nun mit Hilfe des Skalarproduktes vereinfacht so geschrieben werden:

$$\sigma_a(x) = \lceil w \cdot \widetilde{x} \geq 0 \rceil \tag{2.2.1}$$

Zwei Mengen $Y_1, Y_2 \subset X^{d+1}$ heißen *linear separabel*, wenn ein $w \in \mathbb{R}^{d+1}$ existiert, so daß

$$w \cdot y \begin{cases} \geq 0 & falls\, y \in Y_1 \\ < 0 & falls\, y \in Y_2 \end{cases} \tag{2.2.2}$$

Wir sagen auch w *trennt* Y_1 und Y_2.

2.2.3. Satz: Konvergenz des Perceptron Lernalogorithmus

Mit den Bezeichnungen aus 2.2.2 gilt:

Sind $Y_1, Y_2 \subset X^{d+1}$ endlich und separabel, so konvergiert der folgende Algorithmus gegen einen Vektor $w \in \mathbb{R}^{d+1}$, der sie trennt:

1. $w^0 := (0, ..., 0) \in \mathbb{R}^{d+1}$;
2. Wähle $y \in Y_1 \cup Y_2$ beliebig.
   Falls $y \in Y_1$ and $y \cdot w^n < 0$ setze $w^{n+1} := w^n + y$
   Falls $y \in Y_2$ and $y \cdot w^n \geq 0$ setze $w^{n+1} := w^n - y$
   In allen anderen Fällen setze $w^{n+1} := w^n$.
3. Wiederhole ab 2., falls w^{n+1} Y_1 und Y_2 nicht trennt.

Beweis: (nach [1])

1. Es ist äquivalent, zu (2.2.2) $Y := Y_1 \cup \{y \mid -y \in Y_2\}$ zu setzen und zu verlangen, daß $w \cdot y \geq 0 \ \forall y \in Y$ gelten soll.

2. Sei $\left\{y^i\right\}_{i \in \mathbb{N}}$; $y^i \in Y$ eine Folge von Trainingsvektoren, in der jedes $y \in Y$ unendlich oft vorkommt. Sei $\left\{w^i\right\}_{i \in \mathbb{N}}$; $w^i \in \mathbb{R}^{d+1}$ die Folge der sich daraus nach dem Algorithmus ergebenden Gewichtsvektoren. In beiden Folgen gehen wir, ohne die Bezeichnung zu wechseln, zur Teilfolge derjenigen Elemente über, bei denen sich w^n ändert. Dann gilt:

$$w^j \cdot y^j < 0 \tag{2.2.3}$$

und wegen $w^{j+1} = w^j + y^j$ weiter

$$w^{j+1} = y^1 + y^2 + ... + y^j \tag{2.2.4}$$

3. Zu zeigen ist, daß j beschränkt ist.

Sei $w \neq (0, ..., 0)$ ein Vektor, der die Kategorien trennt (und nach Voraussetzung existiert). Wir wählen $0 \neq \alpha \leq \min\{y' \cdot w \mid y' \in Y\}$, und es folgt mit (2.2.4) $w^{j+1} \cdot w = \left(y^1 + ... + y^j\right) \cdot w \geq j\alpha$ und mit der Chauchy-Schwarz'schen Ungleichung $\left|w^{j+1}\right|^2 \geq \frac{j^2\alpha^2}{|w|^2}$.

Andererseits gilt wegen (2.2.3) für alle $j \in \mathbb{N}$, $\left|w^{j+1}\right|^2 = \left|w^j\right|^2 + 2w^j y^j + \left|y^j\right|^2 \leq \left|w^j\right|^2 + \left|y^j\right|^2 \leq \left|w^{j-1}\right|^2 + \left|y^{j-1}\right|^2 + \left|y^j\right|^2 \leq jM$ mit $M = \max\left\{|y|^2 \mid y \in Y\right\}$ folglich $\frac{j^2\alpha^2}{|w|^2} \leq \left|w^{j+1}\right|^2 \leq j|M|$ sowie $j \leq \frac{M|w|^2}{\alpha^2}$ und damit die Aussage.

□

Als zweites wollen wir das oben zitierte Ergebnis über die Grenzen der Parallelisierbarkeit mit dem Perceptron vorstellen. Dabei können aus Platzgründen nicht alle Schritte bewiesen werden, es soll aber deutlich werden, welche Mittel zur Analyse benutzt werden.

2.2.4. Definition: Prädikat

Sei C die Menge der $W = \{0,1\}$ Konfigurationen auf einer endlichen Menge R. Eine Abbildung $\Psi : C \to \{0,1\}$ heißt *Prädikat*. Äquivalent läßt sich ein Prädikat als Funktion der Teilmengen von R definieren, auf denen die Konfiguration c gleich 1 ist: $\widehat{\Psi} : \mathfrak{P}(R) \to \{0,1\}$; $\widehat{\Psi}(X) = \Psi\left(c_{_X}\right)$ mit $c_{_X}^{-1}(\{1\}) = X$.

Der *Träger* von Ψ bzw. $\widehat{\Psi}$ ist die Menge

$$S(\Psi) := S\left(\widehat{\Psi}\right) := \{i \in R \mid \exists c \in C,\, \exists t \in W \text{mit } \Psi(c) \neq \Psi(c_{i,t})\}$$

mit $c_{i,t}(i) = t$ und $c_{i,t}(j) = c(j)$ für $j \neq i$.

Für $X \subset R$ heißt ein Prädikat $\varphi_{_X}$ bzw. $\widehat{\varphi_{_X}}$ mit

$$\widehat{\varphi_{_X}}(Y) = \begin{cases} 1 & \text{falls } X \subset Y \\ 0 & \text{sonst} \end{cases}$$

Maske von X.

Masken lassen sich als Produkte schreiben: $\varphi_{_X}(c) = \prod_{i \in X} c(i)$.

Ich werde im folgenden die beiden Definitionen eines Prädikates parallel benutzen, ohne sie in der Schreibweise von einander zu unterscheiden. Die jeweilige Interpretation ergibt sich aus dem Zusammenhang. Konfigurationen werden (als Abbildungen) mit kleinen Buchstaben notiert, Teilmengen der Retina mit großen.

2.2.5. Satz: Normalform eines Prädikates

Sei C die Menge der $\{0,1\}$ Konfigurationen auf einer endlichen Menge R und M die Menge der Masken auf R. Zu jedem Prädikat Ψ gibt es eine eindeutige Darstellung der Form

$$\Psi(c) = \sum_{\varphi \in M} \alpha_\varphi \varphi(c) \text{ mit } \alpha_\varphi \in \mathbb{Z} \tag{2.2.7}$$

Beweisskizze:

Für jedes $t \in \Psi^{-1}(\{1\})$ bildet man die Funktion

$$D_t(c) := \left(\prod_{i \in t^{-1}(\{1\})} c(i)\right)\left(\prod_{i \in t^{-1}(\{0\})} (1 - c(i))\right)$$

die genau für $c = t$ den Wert 1 annimmt und sonst 0 ist. Dann gilt

$$\Psi(c) = \sum_{t \in \Psi^{-1}(\{1\})} D_t(c)$$

und mit geeigneten Umformungen läßt sich die Existenz und Eindeutigkeit der Normalform zeigen.

□

Aus diesem Satz geht hervor, daß sich jedes Prädikat mit genügend vielen Masken in einem Perceptron implementieren läßt. Allerdings steigt die Anzahl der Teilmengen einer Menge mit der Anzahl ihrer Elemente mit 2^n. Es wird also sehr bald Schranken der Implementierbarkeit geben. Eine andere Frage ist, wie groß die benutzten Masken sein müssen, um ein Prädikat zu implementieren. Um diese Frage zu beantworten, definieren wir die Ordung eines Prädikates:

2.2.6. Definition: Ordung eines Prädikates

Sei $\Psi : \mathfrak{P}(R) \rightarrow \{0,1\}$ ein Prädikat auf einer endlichen Menge R. Ψ wird als *lineare Schwellwertfunktion bezüglich einer Familie* F *von Prädikaten* $\varphi : \mathfrak{P}(R) \rightarrow \{0,1\}$ bezeichnet (Notation: $\Psi \in L(F)$), wenn es sich als Schwellwertfunktion einer gewichteten Summe dieser Prädikate darstellen läßt:

$$\Psi(X) = \left[\sum_{\varphi \in F} \alpha_\varphi \varphi(X) \geq \theta \right] \text{ mit } \theta, \alpha_\varphi \in \mathbb{R} \tag{2.2.10}$$

Als *Ordnung* des Prädikates bezeichnen wir das kleinste k, für das es eine Familie F von Prädikaten gibt, so daß $\Psi \in L(F)$ ist und kein Träger eines $\varphi \in F$ mehr als k Elemente hat.

Die Ordnung eines Prädikates ist also der größte Bereich der Retina, den ein einzelner Feature Detektor "sehen" können muß, um das Prädikat als Perceptron zu implementieren.

Als nächstes benötigen wir einen Satz über die Invarianz von Prädikaten bei Transformation der Werte der Zellen der Retina. Dazu einige Notationen:

2.2.7. Notation:

Sei R eine Menge. Die Bijektionen auf R bezeichnen wir als *Transformationen*. Sie bilden mit der Hintereinanderausführung eine Gruppe G. Zwei Prädikate Ψ und Φ auf R heißen *äquivalent* ($\Phi \underset{H}{\equiv} \Psi$) bezüglich einer Untergruppe $H \subset G$, falls ein $g \in H$ mit $\Psi(gX) = \Phi(X)\ \forall X \subset R$ existiert. Eine Menge von Prädikaten F heißt *abgeschlossen* unter H, wenn für jedes $g \in H$ und jedes $\Phi \in F$ gilt $\Phi g \in F$. Ein Prädikat Ψ heißt *invariant* unter einer Untergruppe $H \subset G$, wenn für jedes $g \in H$ gilt: $\Psi g = \Psi$.

2.2.8. Satz: Invarianz unter Transformationen

Sei R eine endliche Menge und G eine Gruppe von Transformationen darauf. Sei F eine unter G abgeschlossene Familie von Prädikaten auf R. Sei Ψ ein Prädikat, das eine lineare Schwellwertfunktion bezüglich F ist, dann existiert eine Darstellung

$$\Psi(X) = \left[\sum_{\varphi \in F} \beta_\varphi \varphi(X) \geq 0 \right]$$

deren Koeffizienten β_φ nur von der G–Äquivalenzklasse von φ abhängen, für die also gilt: $\varphi \underset{G}{\equiv} \varphi' \quad \Rightarrow \quad \beta_\varphi = \beta_{\varphi'}$.

Beweisskizze:

Aus (2.2.10) wird für beliebiges X mit $\Psi(X) = 1$ in einer Art Mittelung die Formel

$$\Psi(X) = \left[\sum_{\varphi \in F} \left(\sum_{g \in G} \alpha_{\varphi \circ g} \right) \varphi(X) \geq 0 \right]$$

hergeleitet und $\beta_\varphi := \sum_{g \in G} \alpha_{\varphi \circ g}$ gesetzt. Analog verfährt man für X mit $\Psi(X) = 0$

Mit $\varphi = \varphi' h$ folgt dann $\beta_\varphi = \sum_{g \in G} \alpha_{\varphi \circ g} = \sum_{g \in G} \alpha_{\varphi' h \circ g} = \sum_{g \in G} \alpha_{\varphi' \circ g} = \beta_{\varphi'}$

□

2.2.9. Korrolar:

Ist $F = F_1 \cup ... \cup F_m$ eine Zerlegung in Äquivalenzklassen bezüglich $\underset{G}{\equiv}$ dann gilt

$$\Psi(X) = \left\lceil \sum_{i=1}^{m} \alpha_i N_i(X) \geq 0 \right\rceil \text{ mit } N_i(X) = \#\{(\varphi \in F_i \mid \varphi(X) = 1)\}$$

Beweis:

$$\sum_{\varphi \in F} \alpha_\varphi \varphi(X) = \sum_{i=1}^{m} \sum_{\varphi \in F_i} \alpha_\varphi \varphi(X) = \sum_{i=1}^{m} \alpha_i \sum_{\varphi \in F_i} \varphi(X) = \sum_{i=1}^{m} \alpha_i N_i(X)$$

□

2.2.10. Satz: Ordnung von Ψ_{gerade}

Sei R endlich. Das Prädikat

$$\Psi : \mathfrak{P}(R) \to \{0,1\};\ \Psi(X) := \Psi_{gerade}(X) = \lceil \#X \text{ ist gerade Zahl} \rceil$$

hat die Ordnung $\#R$.

Beweis:

1. Ψ ist invariant unter allen Transformationen. Sei k die Ordnung von Ψ, dann gibt es nach 2.2.8 eine Darstellung von Ψ mit $\varphi \in F$ Masken, bei der die Koeffizienten nur von $\#S(\varphi)$, der Mächtigkeit des Trägers von φ, abhängen:

$$\Psi(X) = \left\lceil \sum_{j=0}^{k} \alpha_j N_j(X) \geq 0 \right\rceil$$

mit

$$N_j(X) = \sum_{\varphi \in F_j} \varphi(X) = \#\{Y \subset X \mid \#Y = j\} \text{ da } \varphi(X) = 1 \Leftrightarrow S(\varphi) \subset X$$

Folglich ist

$$N_j(X) = \binom{\#X}{j} = \frac{\#X(\#X-1)...(\#X-j+1)}{j!}$$

ein Polynom in $\#X$ vom Grade $j \leq k$. Damit ist auch $\sum_{j=0}^{k} \alpha_j N_j(X)$ ein Polynom in $\#X$ vom Grade $j \leq k$. Nach Definition gilt aber

$$\Psi(X) = \left\lceil \sum_{j=0}^{k} \alpha_j N_j(X) \geq 0 \right\rceil = \begin{cases} 1 & falls\ \#X = 0, 2, 4, ... \\ 0 & falls\ \#X = 1, 3, 5, ... \end{cases}$$

Als Polynom in $\#X$ ändert die Summe also mit einer entsprechenden Folge $X_0, X_1, ..., X_{\#R}$ mindestens $\#R$ mal das Vorzeichen und muß deshalb als Polynom mindestens vom Grade $\#R$ sein. Folglich hat Ψ die Ordnung $\#R$.

□

2.3. HOPFIELD Netze

HOPFIELD Netze [4] sind im Gegensatz zu Schwellwertlogikelementen und dem Perceptron keine feed forward Netze, sondern vollvernetzt, d.h. jede Zelle ist mit jeder anderen verbunden. Außerdem sind sie im strengen Sinne keine deterministischen neuronalen Netze, da sie eine Zufallskomponente aufweisen: Statt der simultanen Erneuerung aller Zellwerte durch die globale Funktion wird in jedem Schritt eine Zelle zufällig ausgewählt und ihre lokale Funktion ausgeführt. Durch dieses sukzessive Vorgehen kann man zeigen, daß eine geeignet gewählte *Energiefunktion* für das gesamte Netz monoton fällt. Wegen der Endlichkeit des Systems garantiert das die Konvergenz des Netzes gegen eine Konfiguration, die ein lokales Minimum der Energiefunktion ist.

HOPFIELD Netze sind assoziative Speicher bzw. dienen zur Kategorisierung von Mustern: Eine vorgegebene Menge von Konfigurationen soll im Netz gespeichert werden. Wird dann eine beliebige Konfigurtion eingegeben, so soll sich das Netz zu einer ähnlichen Konfiguration aus den vorgegebenen hin entwickeln.

Ein HOPFIELD Netz ist durch seine Gewichtsmatrix festgelegt. (Siehe 1.2) Der Algorithmus sieht folgendermaßen aus:

2.3.1. Definition: HOPFIELD Netz

Sei G eine symmetrische $N \times N$ Matrix mit Nullen in der Hauptdiagonalen, also $G = \{g_{ij}\};\ g_{ij} = g_{ji};\ g_{ii} = 0$ für $i, j = 1...N$.

Sei $H = (I = \{1, ..., N\}, W = \{0, 1\}, N, f)$ ein neuronales Netz mit $N(i) = I$ und Gewichten $g_{i,j}$. Die lokalen Funktionen f_i seien lineare Schwellwertfunktionen mit Schwellen $\theta = 0$.

Für eine zufällig ausgewählte Zelle $i \in \{1...N\}$ wird aus einer Konfiguration $v \in \{0,1\}^N$ ein Nachfolger $\Phi_i(v)$ berechnet, indem mit der lokalen Funktion f_i ein neuer Wert erzeugt wird:

$$\Phi_i : C \to C;\ \ \Phi_i(v)(k) = \begin{cases} f_i(v(I)) = \zeta_0\left(\sum_{j=1}^{N} g_{kj} v(j)\right) & falls\ \ k = i \\ v(k) & sonst \end{cases} \tag{2.3.1}$$

Dieses Verfahren wird iterativ fortgesetzt, bis $v = \Phi_i(v)\ \forall i \in I$ gilt.

Es gilt nun die Endlichkeit dieses Verfahrens nachzuweisen. Dabei kommt es, wie wir sehen werden, auf die Stelle, an der die Änderung vorgenommen wird, nicht an. Allerdings kann die Konfiguration, die schließlich als Ruhezustand erreicht wird, von der Zufallsreihenfolge abhängen, mit der die Zellen gewählt werden.

HOPFIELD definiert in Analogie zur Physik einen *Energiezustand* des Systems nach der Formel:

$$E(v) = -\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} g_{ij} v(i) v(j) \tag{2.3.2}$$

Wegen der Symmetrie der Matrix G kann man die Energiedifferenz, die sich beim Übergang von v nach $\Phi_i(v)$ ergibt, in die Formel

$$\Delta E := E(\Phi_i(v)) - E(v) = -\Delta v(i) \sum_{i=1}^{N} g_{ij} v(j) = -\Delta v(i) \eta_i \tag{2.3.3}$$

zusammenfassen, wobei $\Delta v(i) := \Phi_i(v)(i) - v(i)$ gesetzt ist und $\eta_i = \sum_{j=1}^{N} g_{ij} v(j)$ gerade die Summe ist, die in der Schwellwertfunktion darüber entscheidet, ob $\Phi_i(v)(i)$ gleich 0 oder 1 ist.

Durch die geschickte Wahl der Energiefunktion (2.3.2), läßt sich die Endlichkeit des Algorithmus' dadurch zeigen, daß für alle $v \in \{0,1\}^N$ $\Delta E(v) \leq 0$ gilt, d. h. die Energie im Verlauf des Algorithmus' nicht wächst. Über die Endlichkeit von $\{0,1\}^N$ ergibt sich damit die Endlichkeit des Algorithmus'.

Wir zeigen:

1. $\Delta E(v) \leq 0 \ \forall \, v \in \{0,1\}^N$

   a) $\eta_i(v) < 0 \Rightarrow \Phi_i(v)(i) = 0 \Rightarrow \Delta v(i) \leq 0 \Rightarrow \Delta E(v) \leq 0$

   b) $\eta_i(v) \geq 0 \Rightarrow \Phi_i(v)(i) = 1 \Rightarrow \Delta v(i) \geq 0 \Rightarrow \Delta E(v) \leq 0$

2. Falls $\Delta E(v) = 0$ und $v \neq \Phi_i(v)$ ist, gilt $v(i) = 0$ und $\Phi_i(v)(i) = 1$:

   Aus $\Delta E(v) = 0$ und $v(i) \neq \Phi_i(v)(i)$ folgt $\eta_i = 0$. Daher muß $\Phi_i(v)(i) = 1$ sein und folglich $v(i) = 0$.

Es muß also in jedem Fall, in dem $E(v) = E(\Phi_i(v))$ und $v \neq \Phi_i(v)$ gilt, in einer Zelle der Wert von 0 auf 1 gesetzt werden. Da nur endlich viele Zellen existieren, muß irgendwann ein Fixpunkt erreicht werden.(4)

Ist eine Konfiguration erreicht, bei der jede Änderung des Wertes einer einzelnen Zelle die Energie erhöht, ist dies ein lokales Minimum der Energiefunktion und ein Fixpunkt der Iteration.(5)

HOPFIELD versucht durch die Konstruktion der Matrix G, eine vorgegebene Menge S von Konfigurationen zu lokalen Minima der Energiefunktion und damit zu Fixpunkten der Iteration des Algorithmus' zu machen. Dazu verwendet er die Formel

$$g_{i,j} = \begin{cases} \sum_{s \in S} (2s(i)-1)(2s(j)-1) & falls \ \ i \neq j \\ 0 & sonst \end{cases} \tag{2.3.4}$$

Die Summanden sind gerade gleich 1 bzw -1, je nach dem, ob $s(i) = s(j)$ oder $s(i) \neq s(j)$ ist. Sind also die Werte der Zellen i und j in der Mehrzahl der Konfigurationen aus S gleich, so ergibt sich ein positives Gewicht g_{ij} ansonsten ist g_{ij} negativ bzw. Null.

Zu der Energie (2.3.2) tragen nur die Gewichte bei, die Zellen verbinden, deren Werte in der Konfiguration beide eins sind. Der Beitrag ist negativ, wenn in der Mehrzahl der Konfigurationen aus S die Werte der Zellen gleich sind, sonst ist er Null oder positiv. Setzt

---

4) Da die Auswahl der Zelle i zufällig war, müßte man genau genommen davon ausgehen, daß zuerst eine Zufallsfolge von Elementen aus $\{1,...,N\}$ gewählt wird, in der jedes $i \in \{1,...,N\}$ unendlich oft vorkommt. Für jede solche Folge wäre dann die Konvergenz gegen einen Fixpunkt bewiesen.

5) Als Abstandsmaß zwischen den Konfigurationen wird die "Hamming distance" gewählt, das ist die Anzahl der Zellen, in denen sich die Werte zweier Konfigurationen unterscheiden.

man in die Formel (2.3.2) die Formel (2.3.4) ein, ergibt sich

$$E(v) = -\frac{1}{2}\left(\sum_{i=1}^{N}\sum_{\substack{j=1\\ i\neq j}}^{N}\left(\sum_{s\in S}(2s(i)-1)(2s(j)-1)\right)v(i)v(j)\right)$$
$$= -\frac{1}{2}\left(\sum_{s\in S}\sum_{i=1}^{N}\sum_{\substack{j=1\\ i\neq j}}^{N}(2s(i)-1)(2s(j)-1)v(i)v(j)\right)$$

Für ein spezielles $z \in S$ kann man diese Summe noch aufspalten:

$$\begin{aligned} E(v) = &-\frac{1}{2}\sum_{\substack{s\in S\\ z\neq s}}\sum_{i=1}^{N}\sum_{\substack{j=1\\ i\neq j}}^{N}(2s(i)-1)(2s(j)-1)v(i)v(j) \\ &-\frac{1}{2}\sum_{i=1}^{N}\sum_{\substack{j=1\\ i\neq j}}^{N}(2z(i)-1)v(i)(2z(j)-1)v(j) \end{aligned} \tag{2.3.6}$$

Die Hoffnung, daß die Elemente aus S zu lokalen Minima der Energiefunktion und damit zu Fixpunkten werden, beruht auf dem zweiten Teil von (2.3.6). Dieser wird betragsmäßig groß, wenn $v = z$ oder $v = 1 - z$ (die Konfiguration, bei der die Werte 0 und 1 vertauscht sind) ist. Falls $z(i) = 1 \Rightarrow v(i) = 1$ gilt, aber nicht die Umkehrung davon, kann es vorkommen, daß in diesem zweiten Teil negative Summanden auftreten, die, wegen des negativen Vorzeichens der Summe, die Energie von v wieder erhöhen.

Wie HOPFIELD "experimentell" beobachtete, tritt Konvergenz gegen einen nahen Fixpunkt auf, wenn die Anzahl der Konfigurationen in S klein ist im Vergleich zu N (für ungefähr $\#S < 0,15 \cdot N$) und wenn die Abstände zwischen den Konfigurationen aus S groß sind.

Diese Ergebnisse lassen sich durch die Formel (2.3.6) erklären:

— Der Einfluß einer einzelnen Konfiguration aus S auf (2.3.6) wird größer, je weniger Elemente S hat.

— Eine Konfiguration v liegt nahe bei z, wenn die Werte in vielen Zellen gleich sind. Wenn die Abstände zwischen den Konfigurationen aus S groß sind, sollten kleine Veränderungen von v nur einen geringen Einfluß auf den ersten Teil von (2.3.6) haben, da im Mittel nur für wenige Paare (i, j), für die $z(i) = z(j)$ gilt, für beliebiges $s \in S,\ s \neq z$ auch $s(i) = s(j)$ gilt. Der zweite Teil von (2.3.6) wird daher überproportional zu (2.3.6) beitragen. Zugleich ist die Wahrscheinlichkeit groß, daß bei der Veränderung des Wertes einer Zelle j mit $v(j) \neq z(j)$ die Anzahl der Paare (j, k) mit $v(j) = v(k) = 1$ und $z(j) = z(k)$ größer ist als für andere $s \in S$. Dies führt dazu, daß eine Veränderung in Richtung auf z hin wahrscheinlicher wird.

Es sei noch einmal angemerkt, daß bei der Berechnung der Gewichte nach (2.3.4) auch lokale Minima der Energiefunktion entstehen können, die nicht in S liegen. (So unterscheidet die Formel (2.3.4) nicht zwischen $s \in S$ und $1 - s$.) Auch braucht nicht jedes $s \in S$ ein lokales Minimum der Energiefunktion zu sein. Es läßt sich also allgemein die Konvergenz gegen Fixpunkte zeigen, nicht aber die Menge der Fixpunkte angeben. Außerdem können durch die zufällige Auswahl der Zelle, die in jedem Schritt bearbeitet wird, in verschiedenen Durchläufen von gleichen Startkonfigurationen aus verschiedene Fixpunkte erreicht werden.

2.4. Der Backpropagation Algorithmus

Wie wir in Satz 2.2.5 gesehen haben, läßt sich jedes Prädikat auf $\{0,1\}^N$ durch ein Perceptron darstellen, wenn alle Masken als Feature Detektoren vorhanden sind. Dieses Ergebnis läßt sich sofort auf Netze verallgemeinern, bei denen die Ausgabeschicht aus mehr als einer Zelle besteht. Ein solches Netz implementiert dann kein Prädikat mehr, sondern einen inhaltsadressierbaren Speicher, oder mathematisch gesprochen eine Abbildung von $\{0,1\}^N$ nach $\{0,1\}^M$.[6)]

Die Forderung, daß alle Masken als Feature Detektoren vorhanden sein müssen, macht dieses Ergebnis in der Praxis allerdings wenig nützlich. Es wäre daher gut, nur die Feature Detektoren zu implementieren, die tatsächlich gebraucht werden. Dazu benötigt man eine Methode, mit der die Gewichte bei feed forward Netzen mit mehr als zwei Schichten berechnet werden können.

Dafür wurde der sogenannte Backpropagation Algorithmus oder auch Error Backpropagation Algorithmus entwickelt. Dabei werden in einer überwachten Lernphase (siehe 1.3) mit einem Gradientenabstiegsverfahren Gewichte aus Stimulus-Response-Paaren berechnet. Er ist heute eine der am meisten verwendeten Methoden. Allerdings konnte nicht wie beim Perceptron allgemein gezeigt werden, ob und unter welchen Bedingungen er Gewichte erzeugt, die Lösungen implementieren.

Der Algorithmus soll hier aus Platzgründen nur informell dargestellt werden. Eine ausfühliche Darstellung findet sich in [8] Kapitel 8.

Der Backpropagation Algorithmus arbeitet auf einem endlichen n–schichtigen neuronalen feed forward Netz $B = (I, W, N, f)$ mit Schichten $S_0, ..., S_{n-1}$, $W = (0,1)$ und differenzierbaren Outputfunktionen $\xi_i = \xi$, $\forall i \in I$. In der Regel wird $\xi(x) = \left(1 + e^{-\alpha x}\right)^{-1}$ gewählt. Für $\alpha \to \infty$ konvergiert diese Funktion auf $\mathbb{R} \setminus \{0\}$ punktweise gegen die Sprungfunktion ζ_0. Der Übergang von einer Sprungfunktion zu einer differenzierbaren Outputfunktion ist vor allem technisch motiviert, da man für das Gradientenabstiegsverfahren eine differenzierbare Funktion benötigt.

Die Gewichte werden zunächst zufällig gewählt und dann nach dem folgenden Verfahren verändert: Es wird eine Konfiguration v mit den Werten eines der Stimuli auf den Eingabezellen angelegt und $F^{n-1}(v)$ berechnet. Nun werden die Werte $F^{n-1}(v)(i)$ der Ausgabezellen $i \in S_{n-1}$ mit den gewünschten Ergebnissen $t(i)$ verglichen. Stimmen die Werte in einer Zelle i nicht überein, wird ein Fehlerwert $\delta_i := \left(t(i) - F^{n-1}(v)(i)\right) \cdot \xi'\left(\sum_{j \in N(i)} w_{i,j} \cdot F^{n-2}(v)(j)\right)$ berechnet. Das Gewicht von einem $j \in N(i)$ nach i wird proportional zu dem Produkt $-\delta_i F^{n-2}(v)(j)$ verändert. Außerdem wird ein Fehlerwert an die Nachbarn zurück gegeben. Mit diesen Fehlerwerten werden nun analog die Gewichte zwischen der vor-vorletzten Schicht

---

6) Eine solche Abbildung, die einem Stimulusvektor s^i einen Responsevektor r^i zuordnet, läßt sich leicht angeben, wenn die Stimulusvektoren $s^1, ..., s^K$ orthonormal sind. Man wählt die lineare Abbildung, die durch die Matrix $M := \left\{\sum_{k=1}^{K} r_j^k s_i^k\right\}_{i,j} = \sum_{k=1}^{K} r^k \cdot \left(s^k\right)^t$, also die Summe der Matrizen, die sich als Vektorprodukte der Stimulus-Response-Paare ergeben. Denn dann gilt: $M \cdot s^i = \sum_{k=1}^{K} r^k \cdot \left(s^k\right)^t \cdot s^i = r^i$.

S_{n-3} und der vorletzten Schicht S_{n-2} verändert. Auf diese Weise wird bis zu den Gewichten zwischen den Eingabezellen und der ersten Schicht fortgefahren. Während der Lernphase werden also immer zunächst von den Eingabezellen aus die Ergebnisse des Netzes berechnet und dann in einer zweiten Phase von der Ausgabeschicht aus rückwärts die Gewichte korrigiert.

Berechnet man einen Gesamtfehler als Summe der Quadrate der Differenz über alle Ausgabezellen und alle Stimulus-Response-Paare, so ist die oben angegebene Änderung eines Gewichtes proportional zur partiellen Ableitung dieses Gesamtfehlers nach dem Gewicht. Eine Änderung der Gewichte nach einem Durchgang durch alle Stimulus-Response-Paare, kann also, wenn die Schrittweite der Änderung klein genug gewählt ist, den Gesamtfehler verringern. In der Praxis werden häufig nach jedem Paar die Gewichte um sehr kleine Beträge verändert.

Auch hier, wie beim HOPFIELD Netz, ist nicht garantiert, daß das Verfahren der Gewichtsänderung gegen die "richtigen" Minima der Fehlerfunktion konvergiert.

Experimentelle Ergebnisse deuten aber darauf hin, daß in vielen Fällen mit dem Backpropagation Algorithmus gute Ergebnisse erzielt werden.

2.5. Schlußbemerkung

In diesem Kapitel wurden verschiedene Typen neuronaler Netzte vorgestellt. Während bei der Schwellwertlogik der theoretische Aspekt der Berechnungsuniversalität im Vordergrund steht, sind die anderen Modelle Versuche, Informationsverarbeitungsaufgaben parallel zu lösen. Dabei nimmt mit der Entwicklung immer leistungsfähigerer Rechner die Komplexität der Modelle zu. Dadurch nimmt die Möglichkeit ihrer theoretisch-mathematischen Analyse und der Vorhersagbarkeit ihres Verhaltens ab. Das Perceptron wird in [10] mit Hilfe der linearen Algebra weitgehend aufgrund der Definitionen und zugrunde liegenden Strukturen analysiert. HOPFIELD verwendet eine geschickt gewählte Bewertungsfunktion, um die Konvergenz seines Algorithmus' zu beweisen. Beim Backpropagation Algorithmus ergibt sich die Konvergenz aus der feed forward Struktur des zugrunde liegenden Netzes. Über die Gewichte und das iterative Verhalten des Netzes lassen sich aber kaum allgemeine theoretische Ausagen machen.

Geht man von der Definiton einer zellularen Struktur $S = (I, W, N, f)$ aus, so kann man an jeder der Komponenten mit Untersuchungen ansetzen. Die Menge I der Zellen kann endlich oder unendlich sein. Ferner kann sie eine algebraische Struktur wie z.B. eine Gruppe sein. Werden die Nachbarschaften $N(i)$ mithilfe einer solchen Struktur definiert, können sich daraus Regelmäßigkeiten der iterativen Struktur der globalen Funktion ergeben. (Siehe z.B. [3])

Umgekehrt ergeben sich aus Forderungen an eine Funktion, die durch ein neuronales Netz implementiert werden soll, Konsequenzen für die algebraische Struktur der Menge der Zellen, wie wir am Beispiel von Satz 2.2.10 gesehen haben.

Die Wertemenge W kann ebenfalls endlich oder unendlich sein. Das hat zusammen mit der Menge der Zellen Einfluß auf die Endlichkeit des gesamten Systems. Das oben erwähnte Ergebnis über die Stetigkeit von globalen Funktionen zellularer Strukturen gilt nur für endliche Wertemengen. Ist W eine Gruppe, so kann diese Struktur auf die Menge der Konfigurationen übertragen werden.

Von zentraler Bedeutung für die Untersuchung von neuronalen Netzen sind die lokalen Funktionen. Sind sie linear, so werden neuronale Netze zu linearen Abbildungen. Sind sie

Gruppenhomomorphismen, so ist auch die globale Funktion ein Gruppenhomomorphismus. Dann können entsprechende Ergebnisse angewendet werden. (Siehe z.B. [7])

Diese Ansätze gehen von den Definitionen aus und versuchen auf diesem Weg, die Eigenschaften von neuronalen Netzen zu untersuchen.

Der überwiegende Teil der Veröffentlichungen zum Thema behandelt aber Anwendungen, die meist weniger als mehr theoretisch analysiert werden. Oft werden Heuristiken für umfangreiche Computersimulationen genutzt. Dabei sind die Ergebnisse, die berichtet werden, oft beeindruckend. Gerade in den Bereichen, in denen die klassische künstliche Intelligenz mit Symbolverarbeitungsansätzen oft Schwierigkeiten hat, wie z.B. bei der Verarbeitung von unsicherem oder widersprüchlichem Wissen, bei der Klassifikation von mehrdeutigen oder "verrauschten" Informationen oder bei der Repräsentation von "Weltwissen", scheinen neuronale Netze ihre Stärken zu haben.

Diesen nachzuspüren und sie auch theoretisch zu fundieren scheint mir im Moment eine spannende Aufgabe zu sein, gerade weil die Beiträge zu dem Gebiet aus so vielen und unterschiedlichen Disziplinen stammen.

Adresse des Autors:
Dr. Reginald Ferber
Fachbereich 2
Universität - GH - Paderborn
Postfach 1621
4790 Paderborn

Literaturverzeichnis

[1] Arbib, M. A. *Brains, Machines, and Mathematics, first edition 1965, expanded edition 1987*. Springer, 1987.

[2] Ferber, R. Räumliche und zeitliche Regelmäßigkeiten zellularer Automaten. Dissertation Universität Marburg, 1989.

[3] Hedlund, G. A. Endomorphisms and automorphisms of the shift dynamical system. *Math. System Theory 3* (1969), 320–375.

[4] Hopfield, J. J. Neural networks and physical systems with emergent collective computational abilities. *Proc. Natl. Acad. Sci. USA* (1982).

[5] Kemke, C. Der neuere Konnektionismus. *Informatik Spektrum 11* (1988), 143 – 162.

[6] Kohonen, T. Self-organized formation of topologically correct feature maps. *Biol. Cybern., 43:59 - 69,* (1982).

[7] Lind, D. A. Applications of ergodic theory and sofic systems to cellular automata. In *Physica 10D* (1984), pp. 36–44.

[8] McClelland, J. L., Rumelhart, D. E., and the PDP Research Group. *Parallel Distributed Processing Explorations in the Microstructure of Cognition*. The MIT Press, Cambridge Massachusetts, 1986.

[9] McCulloch, W. S., and Pitts, W. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics, vol. 5* (1943).

[10] Minsky, M. L., and Papert, S. A. *Perceptrons, first edition 1969, expanded edition 1988*. The MIT Press, Cambridge, Ma., 1988.

[11] Müller, H. Selbstreproduktion in zellularen Netzen. In *Jahrbuch Überblicke Mathematik*. Bibliographisches Institut AG, 1978, pp. 67–86.

[12] Palm, G. *Neural Assemblies An alternative Approach to Artificial Intelligence*. Springer, 1982.

[13] Ritter, H., Martinez, T., and Schulten, K. *Neuronale Netze*. Addison-Wesley, 1990.

[14] Robert, F. *Discrete Iteration*. Springer, 1986.

[15] Rosenblatt, F. *Principles of Perceptrons*. Spartan, Washington, DC, 1962.

[16] Strube, G. Neokonnektionismus: Eine neue Basis für die Theorie und Modellierung menschlicher Kognition? *Psychologische Rundschau 41* (1990), 129–143.

Stichwortverzeichnis