Zeitschrift: Bulletin des Schweizerischen Elektrotechnischen Vereins, des Verbandes Schweizerischer Elektrizitätsunternehmen = Bulletin de l'Association suisse des électriciens, de l'Association des entreprises électriques suisses

Herausgeber: Schweizerischer Elektrotechnischer Verein ; Verband Schweizerischer Elektrizitätsunternehmen

Band: 80 (1989)

Heft: 15

Artikel: Neurale Netzwerke : eine Übersicht

Autor: Leber, J.-F. / Matthews, M. B.

DOI: https://doi.org/10.5169/seals-903696

Neurale Netzwerke: eine Übersicht

J.-F. Leber und M.B. Matthews

In der Computerwissenschaft forscht man nach neuen Methoden, die das menschliche Gehirn zum Vorbild nehmen. Diese versprechen vor allem dort Vorteile, wo es um die Erkennung von Bild und Sprache geht. Im vorliegenden Beitrag werden vier der wichtigsten Neuralen Netzwerke beschrieben und die aktuellen Trends auf diesem zukunftsträchtigen Gebiet zusammengefasst.

La science informatique recherche de nouvelles méthodes qui s'orientent au cerveau humain pour reconnaître l'image et la parole. Cet article résume les concepts actuels dans ce domaine d'avenir.

Adresse der Autoren:
J.-F. Leber, Dipl. El.-Ing. ETH, und *M.B. Matthews*, MSEE, Institut für Signal- und Informationsverarbeitung, ETH Zentrum, 8092 Zürich.

Künstliche Neurale Netzwerke gehören zur Kategorie der verteilten technischen Systemen. Sie bestehen aus vielen Elementarrechnern, die nach einem je nach Typ verschiedenen Muster oder einer Struktur miteinander verbunden sind. Charakteristisch für *alle* Neuralen Netzwerke ist, dass alle ihre Elementarrechner (Neuronen) identisch sind und eine verhältnismässig einfache (lokale) Funktion ausüben. Erst das Netzwerk als Gesamtheit aller Bausteine bewirkt ein interessantes Verhalten.

Zur Entwicklungsgeschichte dieses modernen Forschungsgebiets lässt sich sagen, dass Menschen sich schon seit langer Zeit fragen, wie das biologische Nervensystem wohl funktioniert, das für verschiedene Lebewesen eine mehr oder weniger intelligente, jedenfalls aber gewaltige Signalverarbeitungsleistung vollbringt. Wie gewaltig diese Leistung ist, zeigt sich vor allem dem, der das Auge oder Ohr nachzubilden versucht. Von den zahllosen technischen Anwendungen (zu denen auch Probleme der künstlichen Intelligenz zählen) sind die Erkennung von Bild und Sprache besonders zu erwähnen; sie gehören zu den herausforderndsten aktuellen Forschungsgebieten.

Obwohl der digitale Computer weit schneller und genauer als ein Mensch rechnen kann und vielerorts Dienste leistet, die nicht mehr wegzudenken sind (Beispiele Taschenrechner, Medizin, Kommunikationstechnik, Finanzwesen, Verkehrsmittel, Compact Disc, Simulationen), gibt es eine ganze Reihe von Problemen, deren Lösung unerwartet harzig vorangeht. Diesen Problemkindern ist meist eines gemeinsam: Nicht algorithmisch erfassbare (d.h. nicht algebraisch modellierbare) Signale müssen in einer Trainingsphase *kennengelernt*, zum Speichern *dargestellt* und beim Wiedervorkommen *erkannt* werden. Die Hauptschwierigkeit liegt bei der *Erkennung*.

Andererseits kann bereits ein dreijähriges Kind recht gut sehen und hören, stehen, laufen usw.; sein ganzes Leben lang wird es auf Lernen und Erkennen angewiesen sein, um in unserer komplexen Umwelt zu überleben. Diese Feststellung motivierte die Forscher, vom biologischen Nervensystem zu lernen, um Teile seiner Funktionen für technische Anwendungen einzusetzen. In diesem Artikel werden die Grundkonzepte vorgestellt, zu denen neben vielen anderen vor allem vier Forscher beigetragen haben: *Rumelhart, Hopfield, Kohonen und Grossberg*.

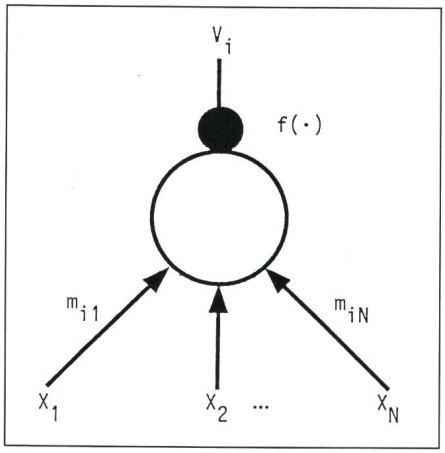
Das Neuron – Baustein aller Neuralen Netzwerke

Bevor einige der heute im Vordergrund stehenden Netzwerkstrukturen behandelt werden, müssen die Eigenschaften des Neurons, das – wie bereits gesagt – für alle Netzwerke gleich ist, vorgestellt werden. Jedes Neuron bildet eine Funktion f (·) der Summe jedes seiner Eingänge x_j (Gl.1), von denen jeder mit einem Gewichtsfaktor (auch Synapse genannt) m_j multipliziert wird.

Der lineare Teil ist damit eine Linearkombination der Eingänge und kann als Skalarprodukt des Eingangsvektors $\boldsymbol{x}^T = [x_1, x_2, \ldots, x_N]$ und Gewichtsvektor $\boldsymbol{m}^T = [m_1, m_2, \ldots, m_N]$[1] geschrieben werden (Fig. 1):

$$v = f\left(\sum_{j=1}^{N} m_j x_j\right) = f(\boldsymbol{m}^T \boldsymbol{x}) \qquad (1)$$

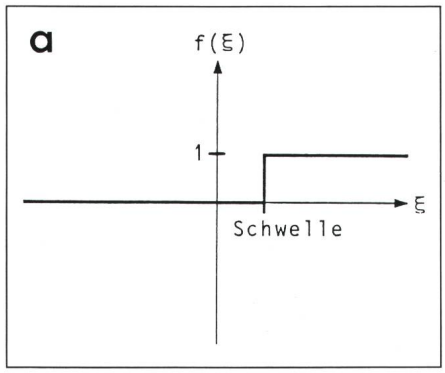
[1] Alle Vektoren sind Kolonnenvektoren. Der Exponent T bezeichnet die Transponierung.

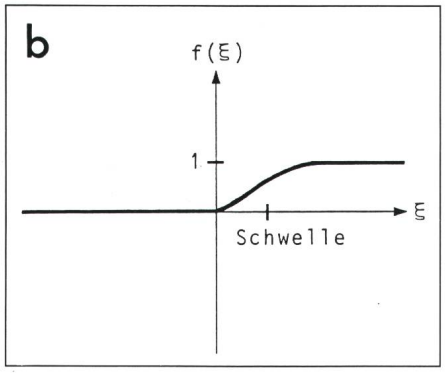


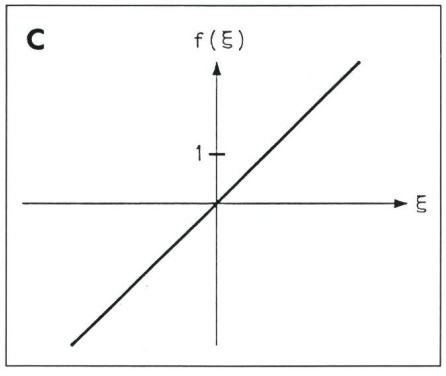
Figur 1 Funktion jedes Neurons

Jedes Neuron v_i bildet die nichtlineare Funktion $f(\cdot)$ der Summe seiner Eingänge x_j multipliziert mit dem Gewicht der Synapse m_{ij}:

$v_i = f(\Sigma_{j=1}^{N} m_{ij} x_j)$.







Figur 2 Schwellenfunktion

Die nichtlineare Funktion $f(\xi)$ kann treppenförmig (a), sigmoidförmig (b) oder linear (c) gewählt werden. Die exakte mathematische Formulierung ist nicht wesentlich.

Der nichtlineare Teil in f variiert je nach Modell zwischen treppenförmig, sigmoidförmig oder linear, wie in Figur 2 illustriert wird. Für die meisten Modelle ist wesentlich, dass der Ausgang jedes Neurons durch zwei Sättigungen begrenzt wird, wodurch drei Zustandsgebiete geschaffen werden: Ist das Skalarprodukt viel grösser als die Schwelle zwischen beiden Sättigungen, wird der Neuronausgang 1; ist das Skalarprodukt viel kleiner als die Schwelle, wird der Neuronausgang 0; bewegt sich das Skalarprodukt um die Schwelle herum, bewegt sich der Neuronausgang zwischen 0 und 1 im quasi-linearen Bereich.

In der Literatur findet man zwei leicht unterschiedliche Interpretationen eines Neurons. Bei den Perceptrons z.B. wird es als trennende Hyperebene, beim Kohonennetzwerk und anderen als Schloss dargestellt.

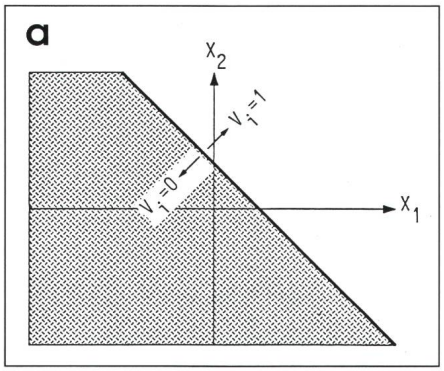
Die Funktion der trennenden Hyperebene[2] soll am Beispiel eines Neurons mit zwei Eingängen x_1, x_2 (Eingangsvektor $\boldsymbol{x}$) und dem Gewichtsvektor $\boldsymbol{m} = (m_1, m_2)$ erklärt werden (Fig. 3a). Jeder Wert von $\boldsymbol{x}$ kann als Punkt auf einer zweidimensionalen Fläche interpretiert werden. Der zweidimensionale Gewichtsvektor definiert eine zu ihm senkrechte Gerade[3] g. Bei einer treppenförmigen Schwelle wird der Neuronausgang den Wert 1 annehmen für alle Punkte von x, die auf der einen Seite der Geraden liegen, und den Wert 0 für alle Punkte, die auf der anderen Seite der Geraden liegen.

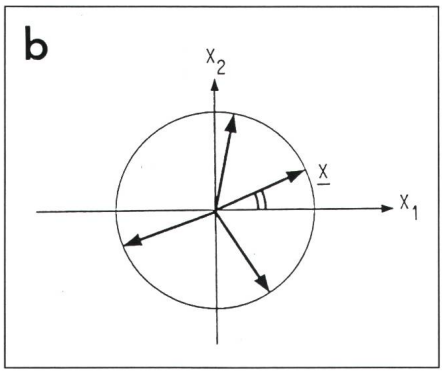
$$v = f(m_1 x_1 + m_2 x_2) \qquad (2)$$

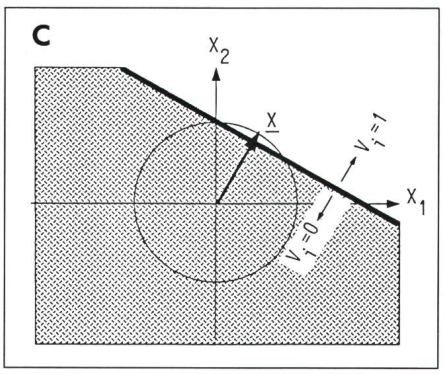
$$v = \begin{cases} 1, & m_1x_1 + m_2x_2 \gg Schwelle \\ 0 < v < 1, & m_1x_1 + m_2x_2 \simeq Schwelle \\ 0, & m_1x_1 + m_2x_2 \ll Schwelle \end{cases} \qquad (3)$$

Dieses Verhalten lässt sich auf eine beliebige Dimension der Eingangs- und Gewichtsvektoren verallgemeinern. Das Neuron entscheidet stets, auf welcher Seite der Hyperebene – die, wie gesagt, immer senkrecht zum Gewichtsvektor steht – der aktuelle Eingangsvektor liegt, oder gibt im quasi-linearen Bereich die Nähe zur Schwelle an.

Bei der zweiten Art, die Funktion eines einzelnen Neurons zu interpretieren (z.B. bei den Kohonennetzwerken) wird die Einschränkung gemacht, dass alle Eingangsvektoren normiert







Figur 3 Mögliche Interpretationen der Neuronenfunktion

a. Jedes Neuron wirkt als Hyperebene (eine Gerade im zweidimensionalen Fall), welche den Eingangsvektorraum in zwei Hälften trennt, wobei der Neuronenausgang auf der einen Seite 1 und auf der anderen Seite 0 ist.

b. Wenn die euklidsche Länge aller Eingangsvektoren gleich ist, unterscheiden sie sich nur durch ihren Winkel.

c. Im normierten Fall öffnet das Neuron wie ein Schloss zu einem passenden Schlüssel, wobei das Skalarprodukt von Eingangsvektor und Gewichtsvektor gross genug, bzw. ihre euklidsche Distanz klein genug sein muss.

[2] Eine Hyperebene ist ein $(n-1)$-dimensionaler Raum, der den übergeordneten n-dimensionalen Raum in zwei Teilräume auftrennt. Eine Ebene z.B. trennt den 3dimensionalen euklidischen Raum auf, eine Gerade den 2dimensionalen.

[3] Die Geradengleichung lautet nämlich $\boldsymbol{m}^T \cdot \boldsymbol{x}$ = const.

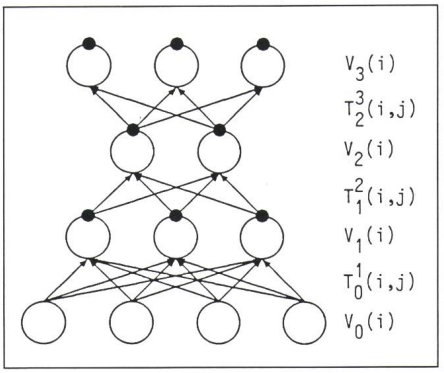
sind, d.h. dass ihre euklidsche Länge (Betrag)

$\sqrt{\Sigma_{j=1}^{N} x_j^2}$

konstant (z.B. 1) ist. Die einzelnen Vektoren unterscheiden sich, wie Figur 3b zeigt, nur durch ihren Winkel. Der Gewichtsvektor und die Schwelle können dann, wie in Figur 3c veranschaulicht, so gewählt werden, dass der Neuronausgang dann und nur dann 1 ist, wenn der Eingangsvektor in einem engen Winkelbereich um den (senkrecht zur Hyperebene liegenden) Gewichtsvektor liegt. Dann wirkt das Neuron, bzw. sein Gewichtsvektor, wie ein Schloss, das der passende Schlüssel (Eingangsvektor) öffnet (Ausgang wird 1)[4].

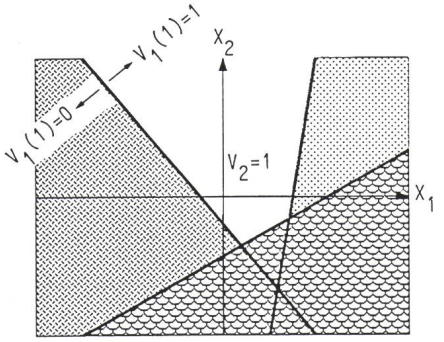
Zusammenfassend kann gesagt werden, dass die lokale Funktion *aller* Bausteine eines Neuralen Netzwerks als einfaches Skalarprodukt anzusehen ist, das mit Hilfe der nichtlinearen Schwelle entscheiden kann, ob der lokale Eingangsvektor zum Schloss passt oder auf welcher Seite der Hyperebene er steht. Dieses Konzept findet man durchwegs durch alle Modelle, auch wenn diese in den Einzelheiten davon abweichen. Versuche haben bestätigt, dass das Netzwerk viel bedeutungsvoller ist als die Details der Implementation ihrer Einzelbausteine.

In den folgenden Abschnitten werden einige Strukturen vorgestellt, die Neuronen derart vernetzen und ihre Parameter trainieren, dass ein interessantes Verhalten entsteht.



Figur 4 Die Neurale-Netzwerk-Struktur des 3-Schichten-Perceptrons

Jedes Neuron einer Schicht wird von allen Neuronen der vorigen Schicht gespeist. Schicht 0 repräsentiert die Eingangssignale, Schicht 3 die Ausgangssignale.



Figur 5 Funktion des Mehrschichten-Perceptrons

Der Ausgang des Neurons V_2 ist nur dann 1, wenn sich der Eingangsvektor x im überlappenden Gebiet derjenigen Seiten aller Hyperebenen (hier 3) der Neuronen der ersten Schicht befindet, die lokal mit einer 1 antworten (UND-Funktion).

Multilayer-Perceptron – assoziative Abbildung

Funktionsweise

Eine heute sehr beliebte Neurale-Netzwerk-Struktur ist das Multilayer (Mehrschichten)-Perceptron. Dieses besteht aus einer Anzahl aufeinanderfolgender Schichten von Neuronen, innerhalb deren keine Verbindungen bestehen. Wie am Beispiel eines 3-Schichten-Perceptrons in Figur 4 gezeigt wird, wird jedes Neuron einer Schicht von allen Neuronen der vorigen Schicht gespeist. Schicht 0 repräsentiert die Eingangssignale, Schicht 3 die Ausgangssignale. Jedem Pfeilende lässt sich ein Gewicht zuordnen, allen Pfeilenden, die auf ein bestimmtes Neuron treffen, *ein* Gewichtsvektor. Die Notation T_2^3 (3,1) steht für das Gewicht einer Verbindung, die von Schicht 2, Neuron 1 herkommt und zum Neuron 3 der dritten Schicht führt.

Die globale Funktion dieses Netzwerks kann – wie im folgenden gezeigt wird – als *assoziative Abbildung*[5] des Eingangsvektors auf den Ausgangsvektor bezeichnet werden. Legt man bei einem solchen Netzwerk einen bestimmten Vektor an den Eingang, so wird sich nach einer kurzen Durchlaufzeit der assoziierte Vektor am Ausgang einstellen. Der Zusammenhang zwischen Ein- und Ausgangsvektor liegt in den einzelnen Gewichten verborgen und kann folgendermassen erklärt werden: Jedes Neuron der ersten Schicht arbeitet, wie im vorigen Kapitel beschrieben, als trennende Hyperebene, d.h., sein Ausgang ist 0 oder 1 abhängig davon, ob der Eingangsvektor auf der einen oder anderen Seite der Hyperebene liegt. Jedes Neuron der zweiten Schicht kombiniert nun die Ergebnisse der einzelnen Neuronen der vorhergehenden Schicht. Sein Gewichtsvektor und seine Schwelle können z.B. so eingestellt werden, dass sein Ausgang nur dann 1 ist, wenn jeder seiner Eingänge 1 ist (logisches UND) oder 1, wenn ein oder mehrere Eingänge gleich 1 sind (logisches ODER). Es kann gezeigt werden, dass ein Neuron die meisten logischen Grundoperationen erzeugen kann, auch solche, die Negationen enthalten. Wenn nun ein Neuron der zweiten Schicht als UND funktioniert, bedeutet dies, dass sein Ausgang nur dann 1 ist, wenn der System-Eingangsvektor in einem Raumgebiet liegt, das für alle Hyperebenen der Neuronen der ersten Schicht positiv ist, d.h. wo diese lokal mit einer 1 antworten (Fig. 5).

Ein Neuron der dritten Schicht (Ausgangsneuron) kann nun z.B. eine ODER-Verknüpfung der Neuronenausgänge der zweiten Schicht durchführen. Falls diese – wie vorhin beschrieben – UND-Funktionen ausüben, wird das Ausgangsneuron eine 1 erzeugen, sobald der Eingangsvektor in eines der in der zweiten Schicht definierten positiven Gebiete zu liegen kommt. Eine tiefere Analyse zeigt, dass jedes Ausgangsneuron auf ein beliebiges, möglicherweise unzusammenhängendes Gebiet des Eingangsvektorraums selektiv gemacht werden kann. Und weil dies für jedes Ausgangsneuron einzeln gilt, kann ein und dasselbe Netzwerk auf jeden beliebigen Eingangsvektor aus einem vorgegebenen Satz von P assoziativen Paaren von Vektoren (V_0^p, V_n^p, $p = 1 \ldots P$, den gewünschten (assoziierten) Ausgangsvektor hervorrufen, sofern genug Neuronen vorhanden sind, um die gewünschten Gebiete zu erzeugen. Mit anderen Worten: Das Netzwerk kann die allgemeinste assoziative Abbildung durchführen[6].

[4] Die Analogie zum Schloss ist nicht perfekt, da das Neuron wegen des quasilinearen Bereichs mit einer gewissen Fehlertoleranz zu öffnen vermag.

[5] Unter Assoziation versteht man die Verknüpfung zweier oder mehrerer Informationen (Vektoren). Im vorliegenden Fall werden benachbarte Vektoren der Eingangsseite in benachbarte Vektoren der Ausgangsseite transformiert, im Spezialfall der perfekten Identifikation in denselben Vektor.

[6] Sind die Eingangssignale binär, genügen sogar zwei Schichten Neuronen für die allgemeinste assoziative Abbildung.

Als Beispiel für eine Anwendung kann eine Reihe von Bildern mit einer Reihe von Namen assoziiert werden. Jeder Bildpunkt wird durch eine Komponente des Eingangsvektors dargestellt und jeder Buchstabe entsprechend durch ein Element (1 Ausgangsneuron) des Ausgangsvektors.

In einem anderen Beispiel könnte eine Reihe von Bildern mit sich selbst assoziiert werden. Dann besteht die Aufgabe des Multilayers-Perceptrons darin, ein verrauschtes oder sonst defektes Bild zu rekonstruieren. Die Rekonstruktionsfähigkeit eines Neuralen Netzwerks ist üblicherweise recht hoch. Darauf einzugehen würde den Rahmen dieses Beitrags sprengen; es sei jedoch erwähnt, dass die Rekonstruktionsfähigkeit einerseits durch die Verteilung der Information auf die vielen Gewichte der vielen Neuronen und andererseits durch die Weichheit der sigmoidförmigen Nichtlinearität (Fangbereich) bedingt ist[7].

Training und Lernen

Nun stellt sich die Frage, wie man ein Multilayer-Perceptron trainieren kann, damit es den vorgegebenen Satz von Assoziationen möglichst beherrscht[8].

Kohonen [1] hat gezeigt, dass im Fall des 1-Schichten-Perceptrons die Gewichte und Schwellen (welche im linearen Fall wegfallen) *direkt* mit Hilfe der Matrizenrechnung berechnet werden können. *Rumelhart* [2] und andere haben den folgenden Backpropagation-Algorithmus eingeführt, der bestrebt ist, den Totalfehler, d.h. die Quadratsumme aller Abweichungen der erzielten von den gewünschten Ausgangsvektoren über alle Assoziationspaare zu minimieren. Dieser Algorithmus lautet:

1. Wähle ein Assoziationspaar ($\boldsymbol{V}_{0n}$, $\boldsymbol{V}_n$) zufällig aus dem (vorgegebenen) Satz von Assoziationen.
2. Lege den Eingangsvektor ans Netzwerk und registriere den erhaltenen Ausgangsvektor $\boldsymbol{V}_n$.
3. Berechne komponentenweise (Index i) aus dem gewünschten Ausgangsvektor $\boldsymbol{D}_n$ und dem tatsächlich erhaltenen Vektor $\boldsymbol{V}_n$ die Fehlerkoeffizienten[9]:

$$\delta_n(i) = V_n(i) \cdot (1 - V_n(i)) \cdot (D_n(i) - V_n(i)) \quad (4)$$

4. Modifiziere die Gewichte T^n_{n-1} gemäss:

$$\Delta T^n_{n-1}(i,j) = \alpha \cdot \delta_n(i) \cdot V_{n-1}(j) \quad (5)$$

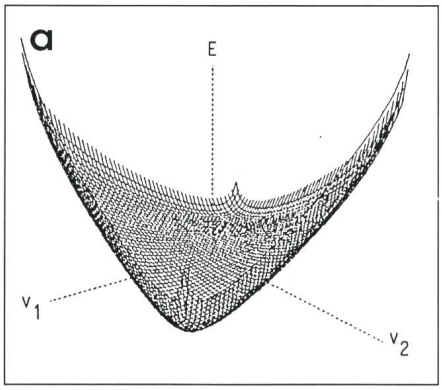
5. Berechne die Fehlerkoeffizienten der nächstunteren Schicht als gewichtete Summe aller Fehlerkoeffizienten derjenigen Neuronen der oberen Schicht, die vom betrachteten Neuron gespeist werden[10].

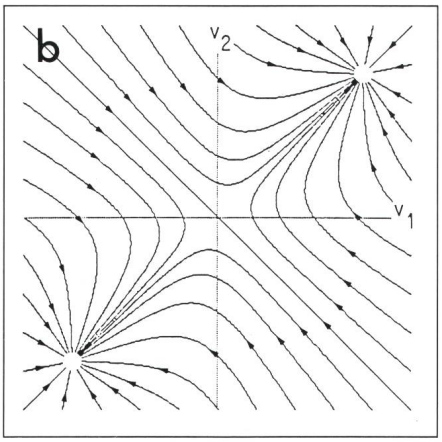
$$\delta_{n-1}(i) = \sum_{\text{alle } k} \delta_n(k) \cdot T^n_{n-1}(k,i) \quad (6)$$

6. Fahre mit der Modifizierung der Gewichte T^{n-1}_{n-2} gemäss Punkt 4 fort bis hinunter zur 1. Schicht; dann gehe zurück zu Punkt 1.

Es ist zu erwähnen, dass meist ein recht langes Training nötig ist, bis der ganze vorgegebene Satz von Assoziationen genügend genau reproduziert wird. Die Konvergenz ist nicht gewährleistet, wenn das Netzwerk zuwenig Neuronen oder weniger als 3 Schichten besitzt. Die Anzahl benötigter Neuronen hängt von der Anzahl der Trainingsvektoren und der Komplexität ihrer Ähnlichkeitsbeziehungen ab. Sie stellt stets ein Kompromiss dar, da zu viele Neuronen die Konvergenz ebenfalls verschlechtern können[11]. Nach dem Training ist das Neurale Netzwerk ein Modell für den vorgegebenen Satz von Assoziationen geworden, deren Information in den Gewichten und Schwellen[12] verteilt beinhaltet ist. Hier wäre noch darauf hinzuweisen, dass Vergessen in engem Zusammenhang zu Training und Lernen steht, da Training frühere Muster allmählich überschreibt.

Ein interessantes Experiment mit einem Stück schriftlicher englischer Sprache als Eingangsvektor und den gewünschten Sprechmuskelpositionen als Ausgangsvektor hat gezeigt, dass schon ein relativ kleines Multilayer-Perceptron (Grössenordnung 100 Neuronen) in der Lage ist, gar nicht so schlecht lesen zu lernen. Das Neurale Netzwerk hat also die Sprechregeln, die in den vorgeführten Beispielen implizit enthalten sind, extrahiert und kann recht gut verallgemeinern, d.h. neue (noch nicht erfahrene) Sätze mit unerwartet wenig Fehlern vorlesen [3].





Figur 6 Hopfield-Netzwerk
Beispiel einer dreidimensionalen Energiefläche (a) und der zugehörigen Trajektorienschar (b). Der Systemzustand konvergiert stets entlang einer Trajektorie zu einem Minimum der Energiefunktion.

[7] Mehrere Schichten und/oder Rückkopplungen können die Rekonstruktionsfähigkeit drastisch erhöhen, weil ein Neuron seine tolerante Entscheidung dann auf die toleranten Entscheidungen seiner Vorgänger gründen kann.

[8] «Möglichst» weist darauf hin, dass ein Netzwerk gegebener Grösse nach noch so langem Training nicht unbedingt fähig ist, jede assoziative Abbildung exakt oder innerhalb eines gewissen Toleranzbereichs richtig durchzuführen.

[9] Die Gleichung gilt für den Fall, dass die Sigmoidfunktion als

$$f(\xi) = \frac{1}{1 - e^{-\xi}}$$

definiert ist. Sie wird hier nicht weiter erklärt.

[10] Das betrachtete Neuron erhält somit dann einen grossen Fehlerkoeffizienten, wenn es grosse Fehler auf der nächsthöheren Schicht verursacht hat.

[11] Ein solches von Natur aus überwachtes (supervised) Training ergibt die oben erwähnte «UND» - «ODER»-Struktur nicht unbedingt; jene wird nur verwendet, um zu beweisen, dass 3 Schichten prinzipiell genügen, um jede beliebige Abbildung durchzuführen.

[12] Eine Schwelle wird zu Trainingszwecken wie ein Gewicht behandelt, das stets mit 1 gespeist wird. Ein Neuron fester Schwelle mit einem solchen Zusatzgewicht verhält sich equivalent zu einem Neuron variabler Schwelle ohne Zusatzgewicht.

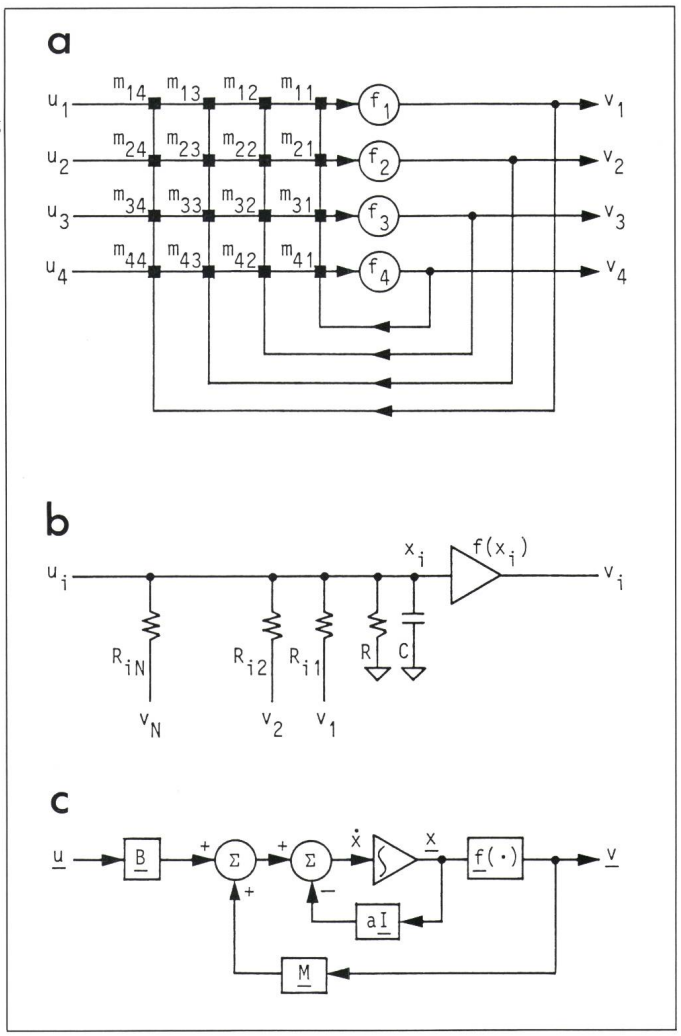
Hopfield-Netzwerk: assoziativer Speicher

Prinzip

In der Systemtheorie ist bekannt, dass ein stabiles und dynamisches System eine Menge von Anziehungspunkten (sogenannte Punkt-Attraktoren) im Zustandsvektorraum besitzt, gegen die das System strebt, um die gesamte Energie des Systems zu minimieren. Im zweidimensionalen Fall entsprechen die Punkt-Attraktoren den Minima einer Energiefläche (Fig. 6). Der Zustandsvektorraum ist ein Vektorraum, der von den verschiedenen Zustandsvariablen (Grössen, die das dynamische Verhalten eines Systems beschreiben) des Systems aufgespannt ist. Bei Neuralen Netzwerken sind dies die verschiedenen Neuronenausgänge. Jeder Attraktor im Zustandsvektorraum besitzt sein eigenes Einzugsgebiet. Gegen welchen dieser Attraktoren der Systemzustand konvergiert, hängt davon ab, in welchem Einzugsgebiet sich das System beim Prozessstart befindet. Die entsprechende Trajektorie des Systemzustandes vom Anfangspunkt bis zum Attraktor kann oft sehr vielfältig und kompliziert sein. Die Figur 6a zeigt ein Beispiel einer Energiefläche und Figur 6b eine Schar von Trajektorien, entlang welcher der Systemzustand auf ein Minimum der Energiefunktion konvergiert.

Ausserdem kann es auch eigenartige Attraktoren (sog. Strange Attractors) geben, die mit einer chaotischen, sich nicht wiederholenden Trajektorie verknüpft sind. Es ist jedoch durchaus möglich, ein dynamisches System zu konstruieren, das nur regelmässig plazierte Attraktoren beinhaltet und deren Trajektorien von einem bestimmten Anfangspunkt aus immer gegen den nächstliegenden Attraktor streben, in welchem das System stabil wird. Ein solches System stellt eine interessante Art *Informationspeicher* dar. Ein einziger Attraktor kann beispielsweise ein komplettes Muster im Zustandsraum repräsentieren und sein zugehöriges Einzugsgebiet entsprechend eine Menge unvollständiger oder verrauschter Muster. Mehrere komplette Muster können in ein einziges System eingebaut werden, da dieses mehrere Attraktoren besitzen kann. Von einem unvollständigen Muster (als Anfangsbedingung) ausgehend, setzt sich das System dann auf das am besten passende vollständige Muster

Figur 7
Hopfield-Netzwerk
a Struktur
b Elektronische Schaltungsnachbildung eines Neurons
c Systemdarstellung



nieder. Solch ein Systemverhalten nennt man Content Addressable Memory (CAM), weil die «Adresse» eines Attraktors seinem Inhalt entspricht.

Das Hopfield-Netzwerk in Figur 7 stellt ein solches CAM-System dar. Es besteht aus einer Menge einzelner Neuronen, deren N Ausgänge v_i zu allen Neuronen zurückgeführt sind. Jedes Neuron empfängt also sämtliche Ausgänge, welche zum Ausgangsvektor $\boldsymbol{v}^T = [v_1, v_2, ..., v_N]$ zusammengefasst werden können, an seinem Eingang und multipliziert diesen mit seinem Gewichtsvektor $\boldsymbol{m}_i = [m_{i1}, ..., m_{iN}]$. Das Resultat wird über eine nichtlineare Funktion $f_i(\cdot)$ zum Ausgang v_i geleitet. Jedes Neuron besitzt einen weiteren (skalaren) Eingang u_i, der als einstellbare Schwelle dient. Die gesamte Übertragungsfunktion eines Neurons lautet somit:

$$v_i(t+\Delta t) = f_i\left(\sum_{j=1}^{N} m_{ij}\, v_j(t) + u_i(t)\right) = f_i(\boldsymbol{m}_i^T\, \boldsymbol{v}(t) + u_i(t)) \qquad (7)$$

In Matrixnotation[13]:

$$\boldsymbol{v}(t+\Delta t) = \boldsymbol{f}(\boldsymbol{M}^T \boldsymbol{v}(t) + \boldsymbol{u}(t))$$

$$\text{für} \quad \begin{cases} \boldsymbol{M} = [\boldsymbol{m}_1, \boldsymbol{m}_2, ..., \boldsymbol{m}_N] \\ \boldsymbol{u} = [u_1, u_2, ..., u_N]^T \\ \boldsymbol{f}(\cdot) = [f_1(\cdot), ..., f_N(\cdot)]^T \end{cases} \qquad (8)$$

wobei die gespeicherte Information in der Rückkoppelungsmatrix $\boldsymbol{M}$ steckt[13]. Mittels einer Liapunovfunktion[14], die der Energie des Systems entspricht, kann man zeigen, dass das Netzwerk asymptotisch stabil ist, falls $\boldsymbol{M}$ symmetrisch ist (hinreichende Bedingung).

Hopfield hat nun zwei Netzwerkmodelle vorgestellt: ein binäres asynchro-

[13] Es ist zu unterscheiden zwischen $v_i(t+\Delta t)$ und $v_i(t)$, da sich das System nicht im eingeschwungenen Zustand befindet.

[14] Diese Funktionen spielen eine wichtige Rolle bei Stabilitätsbetrachtungen dynamischer Prozesse.

nes Modell und ein kontinuierliches Modell. Beim binären Modell [4] weisen sämtliche Nichtlinearitäten $f_i(\cdot)$ eine einstufige Treppenfunktion auf, wobei

$$v_i = \begin{cases} 1 \; ; \; \boldsymbol{m}_i^T \boldsymbol{v} + u_i > 0 \\ 0 \; ; \; \boldsymbol{m}_i^T \boldsymbol{v} + u_i < 0 \end{cases} \qquad (9)$$

In diesem ersten Modell prüft jedes Neuron seinen Eingang auf asynchrone Weise und ändert seinen Ausgang entsprechend Gleichung 9. Dieses asynchrone Verhalten unterscheidet das Hopfield-Netzwerk von anderen ähnlichen früheren Modellen wie zum Beispiel dem vorne beschriebenen Perceptron.

Das zweite kontinuierliche Modell [5] sieht dem biologischen Gehirn wahrscheinlich schon ähnlicher; es arbeitet mit kontinuierlichen elektrischen Potentialen und kann als elektronische Schaltung dargestellt werden (Fig. 7b). Die Kapazität C und der Widerstand R symbolisieren die Phasenverschiebung, die alle elektronischen Verstärker aufweisen. Die Funktion $f(x_i)$ wird als kontinuierliche, monoton steigende oder sigmoidförmige Funktion angenommen, deren Eingangspannung x_i mittels einer Differentialgleichung beschrieben werden kann:

$$C\dot{x}_i = \sum_{j=1}^{N} \frac{1}{R_{ij}} (v_j - x_i) - \frac{1}{R} x_i + u_i \qquad (10)$$

x_i ist eine Spannungsgrösse, u_i eine Stromgrösse. Die Stabilität dieses Systems kann mittels einer geeigneten Liapunovfunktion leicht bewiesen werden, vorausgesetzt, dass die Neuron-zu-Neuron-Kopplung symmetrisch ist (d.h. $R_{ij} = R_{ji}$) und keine Eigenrückkopplung ($R_{ii} = 0$) besteht. Das ganze vernetzte System kann, wie in der Systemtechnik üblich (Fig. 7c), durch eine Vektordifferentialgleichung beschrieben werden:

$$\begin{aligned} \dot{\boldsymbol{x}} &= a\boldsymbol{x} + \boldsymbol{M}\boldsymbol{v} + \boldsymbol{B}\boldsymbol{u} \\ \boldsymbol{v} &= \boldsymbol{f}(\boldsymbol{x}) \end{aligned} \qquad (11)$$

Der Vorteil dieses zweiten Hopfield-Modells besteht nicht nur darin, dass man es als dynamisches System betrachten und mit Hilfe der weitentwickelten Systemtheorie analysieren kann, sondern, dass man es auch tatsächlich als elektronische Schaltung hardwaremässig realisieren kann. Es sind bereits Hopfield-CAM-Chips in den Forschungslabors entwickelt worden, und mindestens eines davon ist bereits auf dem Markt!

Speichern der Information

Ist man im Besitze eines solchen CAM-Systems, so stellt sich die Frage, wie man die gewünschten Attraktoren in die Matrix $\boldsymbol{M}$ abspeichert und wieviel Information (d.h. wie viele stabile Attraktoren) das Netzwerk als Funktion der Anzahl Neuron enthalten kann. Auf der N-dimensionalen Energiefläche des Systems sind die stabilen Attraktoren einfach lokale Minima der Energiefunktion. Je mehr lokale Minima nun das System hat, desto mehr fliessen ihre Einzugsgebiete ineinander, und zwischen den Minima entsteht eine gewisse Fehlerfläche. Es sind verschiedene Möglichkeiten bekannt, wie man die Muster (Attraktoren) in der Energiefläche erzeugen kann. Aus der Art, wie man sie erzeugt, kann man angenähert schliessen, wie viele zuverlässige Attraktoren es gibt. Hopfield beispielsweise schlug vor, M als eine Art Korrelationsmatrix zu behandeln, die durch eine Menge von L (abzuspeichernden) Mustern $\boldsymbol{v}(1)$, $\boldsymbol{v}(2),\ldots,\boldsymbol{v}(L)$ (Spaltenvektoren) auf folgende Weise gebildet wird:

$$\boldsymbol{M} = \sum_{k=1}^{L} \boldsymbol{v}(k)\, \boldsymbol{v}(k)^T \qquad (12)$$

Definiert man eine Matrix $\boldsymbol{V}$ durch

$$\boldsymbol{V} = [\boldsymbol{v}(1), \quad \boldsymbol{v}(2), \ldots, \boldsymbol{v}(L)] \qquad (13)$$

mit $\boldsymbol{v}(k)$ als Spaltenvektoren, so kann man Gl. 12 auch schreiben als

$$M = [\boldsymbol{v}(1), \boldsymbol{v}(2), \ldots, \boldsymbol{v}(L)] \cdot \begin{bmatrix} \boldsymbol{v}^T(1) \\ \boldsymbol{v}^T(2) \\ \boldsymbol{v}^T(L) \end{bmatrix} = \boldsymbol{V} \cdot \boldsymbol{V}^T \qquad (14)$$

wobei $\boldsymbol{v}^T(k)$ die Zeilenvektoren darstellen.

Wenn man nun eines der Muster $\boldsymbol{v}(n)$ vorgibt, wird der Ausgang zu

$$\boldsymbol{x}_i = \boldsymbol{M}\boldsymbol{v}(n) = (\boldsymbol{V}\boldsymbol{V}^T) \cdot \boldsymbol{v}(n) = \sum_{k=1}^{L} \boldsymbol{v}(k) \left(\boldsymbol{v}(k)^T \cdot \boldsymbol{v}(n) \right) \qquad (15)$$

Wie man aus der Betrachtung der Gleichungen 14 und 15 leicht ersehen kann, wird, falls alle Vektoren $\boldsymbol{v}(1),\ldots,\boldsymbol{v}(L)$ senkrecht aufeinanderstehen, d.h. völlig unkorreliert sind, am Ausgang $\boldsymbol{x}_i = \boldsymbol{v}(n)$ erscheinen. Je korrelierter aber die Muster sind, desto mehr tauchen Unzuverlässigkeiten (als zusammengeschmolzene Punkt-Attraktoren) und verschiedene periodische und chaotische Attraktoren (d.h. kein stabiler Punkt) auf. *Hopfield* selbst gab eine empirische Kapazitätsschätzung von $L \approx 0{,}15\, N$ an, wobei N die Zahl der Ausgänge (Vektordimension) angibt. Später wurde gezeigt, dass die Korrelationsmatrixform von $\boldsymbol{M}$ eine theoretische obere Grenze von $L \leq N/(4 \log N)$ hat.

Andere, kompliziertere Strukturen von $\boldsymbol{M}$ ergeben eine Speicherdichte von $L \leq N$.

Lösung von Optimierungsproblemen

Geht man von einer bestimmten Matrix $\boldsymbol{M}$ und einem Anfangszustandsvektor $\boldsymbol{x}(0)$ aus, so wird das System gemäss der dynamischen Beschreibung von Gleichung (11) folgende Energiefunktion (Lyapunov) minimieren[15]:

$$V(x) = -\frac{1}{2} \boldsymbol{v}^T \boldsymbol{M}\boldsymbol{v} - \boldsymbol{v}^T \boldsymbol{B}\boldsymbol{u} + \sum_{i=1}^{N} a \int_0^{v_i} f_i^{-1}(\alpha)\, d\alpha \qquad (16)$$

Man kann sich nun leicht vorstellen, dass ein vorgegebenes Optimierungsproblem derart umformuliert werden kann, dass die Matrix $\boldsymbol{M}$ die entsprechenden Optimierungsparameter beinhaltet. Seinem energieminimierenden Drang folgend, wird das Netzwerk gegen eine Lösung des Optimierungsproblems im Zustandsraum streben und sich dort stabilisieren. Je nach Problemformulierung gibt es nun meist einige lokale Minima, in denen das System hängenbleiben kann (sie stellen eine suboptimale Lösung des Optimierungsproblems dar). In vielen Fällen aber ist eine solche Lösung eine gute Nähung des Optimums. Es besteht auch die Möglichkeit, das System an mehreren Anfangspunkten zu starten und die beste resultierende Lösung auszuwählen. Ein berühmtes Beispiel

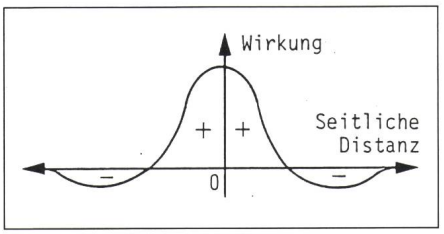
[15] Diese Gleichung wird hier ohne nähere Begründung nur der Vollständigkeit halber angegeben.

dafür ist das sogenannte Travelling Salesman Problem (TSP). Dabei geht es darum, dass ein reisender Kaufmann mehrere Städte besuchen muss und wissen will, welche Reihenfolge von Städten den kürzesten gesamten Weg ergibt. Hier wird jedem Ausgang v_i eine bestimmte Stadt zugeordnet, so dass sich eine Reihenfolge ergibt. Der Zustandsvektor $\boldsymbol{v}$ entspricht dann einem bestimmten Weg durch die Städte. Und in der Matrix $\boldsymbol{M}$ sind die verschiedenen Distanzen zwischen den Städten abgespeichert. Bei N Städten gibt es immer insgesamt $N!/2N$ mögliche Wege. *Hopfield* hat gezeigt, dass das Netzwerk bei der Lösung des TSP-Problems eine erstaunliche Leistungsfähigkeit aufweist. Bei 10 Städten zum Beispiel findet es durchschnittlich bei jedem zweiten Start den besten Weg.

Kohonens selbstorganisierende Abbildungen

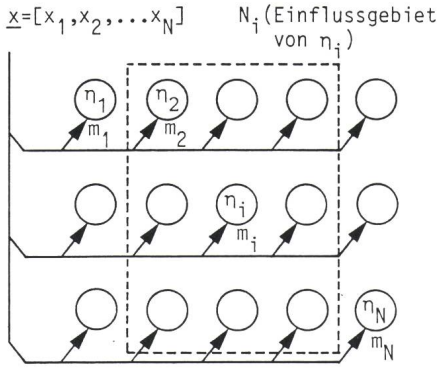
Die Hirnforschung hat in letzter Zeit viel über die Repräsentation und Speicherung von Information im Gehirn gelernt. Man glaubt, dass unser Gedächtnis im Sinne eines physischen Speichers zu einem grossen Teil aus einer Menge vereinfachter Darstellungen der Aussenwelt auf verschiedenen Abstraktionsebenen besteht, die auf zweidimensionale Schichten von Neuronen im Gehirn abgebildet sind. Diese dimensionsreduzierten Darstellungen sind die Folge eines selbstorganisierenden Verhaltens der im Gehirn vorhandenen Netzwerkstruktur, welches damit auf die Eingangssignale antwortet. Bemerkenswert an dieser Struktur sind die ausgebreiteten seitlichen Verbindungen (Lateral Feedback) zwischen den Neuronen: ein Neuron hat seitliche Verbindungen mit bis zu 10 000 benachbarten Neuronen, wobei die Kopplung anregend oder hemmend sein kann, was hauptsächlich von der Distanz abhängt (Fig. 8); nähere Zellen werden angeregt, fernere gehemmt.

Man kann sich fragen, ob eine solche Hirnstruktur bzw. ein solches selbstorganisierendes Verhalten simuliert werden könnte, um die damit verbundene dimensionsreduzierende Funktion technisch ausnützen zu können. Dazu hat Kohonen die sogenannte Topology-preserving Feature Map (topologiebewahrende Merkmalabbildung) vorgestellt. Das Kohonen-Netzwerk besteht aus einer ein- oder zweidimensionalen Ansammlung (Netz) linearer Neuronen und hat die Eigenschaft, dass bestimmte Gebiete von Neuronen eine Empfindlichkeit gegenüber bestimmten charakteristischen Merkmalen des Eingangssignals entwickeln. So ordnen sich z.B. diese spezifischen Gebiete in ungefähr derselben topologischen Ordnung auf dem Netz an wie die entsprechenden Signalmerkmale im Merkmalvektorraum: Ähnliche Eingangssignale werden also nahe beieinander abgebildet. In diesem Sinne werden die entsprechenden topologischen Beziehungen der Eingangssignale bei der Abbildung aufs Netzwerk bewahrt.



Figur 8 Anregung und Hemmung im Kohonen-Netzwerk

Die seitliche anregende und hemmende Wirkung eines aktiven Neurons gleicht einem Mexikanerhut.



Figur 9 Struktur eines zweidimensionalen Kohonen-Netzwerks

Das in Figur 9 dargestellte Netzwerk enthält N lineare Neuronen [η_1, η_2,..., η_N]. Ihnen allen wird derselbe Eingangvektor $\boldsymbol{x}$ eingeprägt, den sie mit ihrem Gewichtsvektor $\boldsymbol{m}_i$ multiplizieren, um den Ausgang v_i zu produzieren. Ausserdem besitzt jedes Neuron ein Einflussgebiet N_i um sich herum, das die seitliche Kopplung zwischen Nachbarneuronen bestimmt. Das selbstorganisierende Verhalten dieses Netzwerkes entspringt der Regel, dass ein Neuron, dessen Gewichtsvektor zu einem bestimmten Eingangsvektor am besten passt, sowohl seinen eigenen Gewichtsvektor als auch denjenigen der im Einflussgebiet liegenden Nachbarn ändert, um sich dem Eingangsvektor besser anzupassen. Das am besten passende Neuron[16], sagen wir η_j, ist dasjenige mit der kleinsten euklidschen Distanz

$$\boldsymbol{m}_j = min_i \| \boldsymbol{m}_i - \boldsymbol{x} \| \tag{17}$$

Dem Lernalgorithmus[17] entsprechend richten sich $\boldsymbol{m}_j$ und alle seine Nachbarn $\boldsymbol{m}_i$, $i \in N_j$, ein wenig nach $\boldsymbol{x}$:

$$\boldsymbol{m}_i[k+1] = \begin{cases} \boldsymbol{m}_i[k] + \alpha(\boldsymbol{x}[k] - \boldsymbol{m}_i[k]) & ; \text{für } i \in N_j \\ \boldsymbol{m}_i[k] & ; \text{sonst} \end{cases} \tag{18}$$

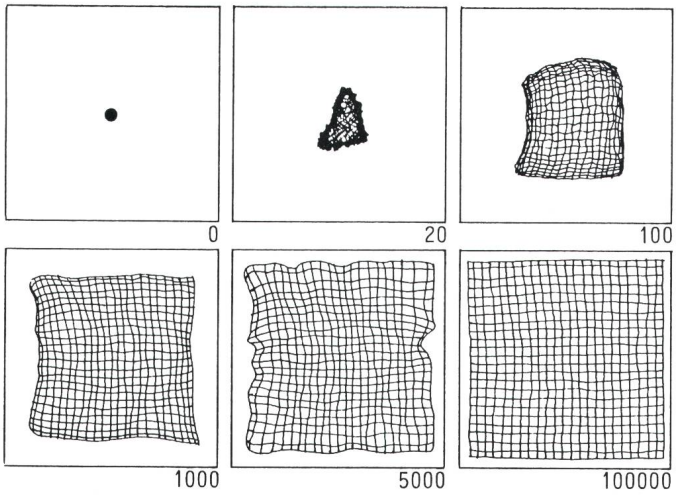
Ausserdem nehmen sowohl die Zeitkonstante α als auch die Grösse der Nachbarschaft N_j im Laufe der Zeit ab, um eine zunehmende Verfeinerung des selbstorganisierenden Verhaltens zu bewirken.

Es stellt sich nun die Frage, was die topologische Bewahrung der Merkmale des Eingangssignales genau bedeutet. Das Eingangssignal sei beispielsweise ein N-dimensionaler Zufallsprozess, der in einem entsprechenden Vektorraum verteilt ist. Merkmale dieses Prozesses betreffen z.B. die Struktur der Verteilung, wie beispielsweise die verschiedenen Grössen und Formen von Gruppierungen (Clusters) im Vektorraum. Die topologische Bewahrung der Merkmale auf einem ein- oder zweidimensionalen Neuralen Netzwerk kann nun bedeuten, dass Distanzen und Anordnungen der Gewichtsvektoren im Netz den Distanzen und Anordnungen der Merkmale im ursprünglichen Vektorraum ungefähr gleich sind.

Als Beispiel zeigt Figur 10 sechs zeitlich aufeinanderfolgende Trainingsphasen eines 25×25-Neuronen-Kohonen-Netzwerks, das von einem zweidimensionalen gleichverteilten Zufallsprozess angeregt wird. Dargestellt ist der Raum der Gewichtsvektoren, wo-

[16] Man erinnere sich der früher beschriebenen Schlossfunktion.

[17] Man beachte, dass Lernen hier einen etwas anderen Charakter hat als bei den vorher beschriebenen Netzwerken.



Figur 10
Beispiel eines zweidimensionalen Netzwerks, das mit einem zweidimensionalen gleichverteilten Zufallsprozess angeregt wird.

bei die Kreuzungen ihre Positionen bezeichnen. Da die Neuronen durch ihre Gewichtsvektoren charakterisiert sind, kann man die Knoten auch den Neuronen zuordnen. Geometrisch benachbarte Neuronen kommen auch im Vektorraum nebeneinander zu liegen. Wie erwartet, wird die Verteilung gleichmässig auf das Netzwerk abgebildet.

Will man ein deterministisches Signal auf das Netz abbilden, so muss man dessen Struktur im N-dimensionalen Vektorraum mit einem gleichverteilten Zufallsprozess abtasten.

Die Figur 11 zeigt ein solches Beispiel, nämlich die Abbildung eines 10dimensionalen Simplex[18], dessen Ecken eine Gruppierung von Punkten beinhalten. Das Simplex wurde von einem gleichverteilten Zufallsvektor abgetastet und in das Neurale Netzwerk eingespeist. Wie man in Figur 11 sehen kann, werden die «Ecken» als Clusters (gleiche Zahlen) auf die Ebene abgebildet.

Grossberg-Netzwerk: Masking Field

Die Neurale Netzwerkstruktur namens Masking Field ist zur Zeit noch wenig bekannt. Sie weist wie das Kohonen-Netzwerk eine (ein- oder zweidimensionale) Schicht von Neuronen auf und ebenfalls seitliche Verbindungen nach dem Muster des «Mexikanerhuts», d.h., unmittelbare Nachbarn werden gefördert, weitere Nachbarn werden behindert, und die ferneren Neuronen werden nicht beeinflusst.

Neu ist hier, dass die einzelnen Neuronen nicht mehr von allen Systemeingängen gespeist werden, sondern nur von einer Gruppe davon. Neu ist ferner, dass die Stärke und Tragweite der seitlichen Behinderung, die ein Neuron ausübt, proportional zur Anzahl seiner Systemeingänge ist[19].

Die Figur 12 zeigt diese Zusammenhänge skizzenhaft auf: Das mittlere Neuron $V_1(2)$ ist das grösste der drei abgebildeten Neuronen, weil es die meisten Eingänge (hier drei) kontrolliert. Das kleinste Neuron $V_1(3)$ hingegen kontrolliert bloss einen Eingang. Das grösste Neuron übt grössere «Macht» aus, indem es seine Nachbarn stärker behindert (durch die Dikke der Rückführungspfade angedeutet) und dadurch, dass die Reichweite seiner seitlichen Behinderungsfähigkeit grösser ist (nicht abgebildet).

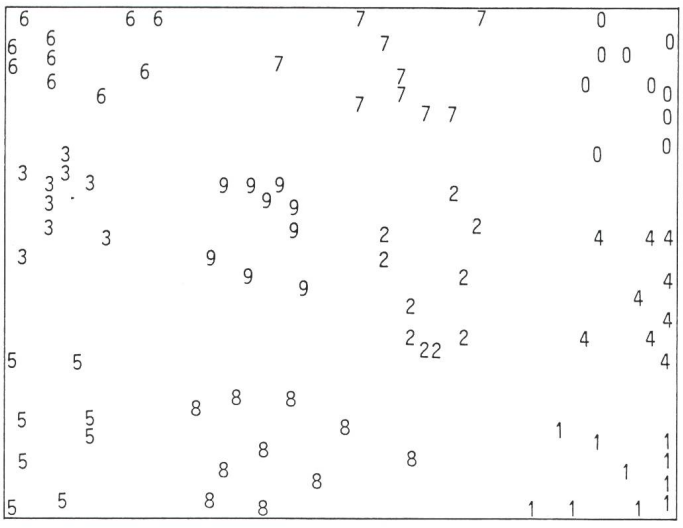
Man kann nun die Gesamtheit der Eingänge nicht mehr als Eingangsvektor betrachten, da sie für die einzelnen Neuronen unterschiedlich ist. Sie repräsentiert nun eine Art Muster, das aus vielen Einzelmustern besteht. Beispielsweise könnte das Muster ein Bild darstellen, auf das verschiedene Symbole geschrieben sind. Nun wird angenommen, das Netzwerk sei schon (unter lokaler Anwendung des Lernalgorithmus der topologischen Abbildung vom vorigen Kapitel) mit allerlei Symbolen trainiert worden. Dabei sei ein grosses Neuron auf das Symbol L selektiv geworden, d.h., es öffne, wenn man ein «L» an die Eingänge legt, weil der Schlüssel sehr gut zu seinem Schloss passt (siehe das Kapitel über das Neuron). Es sei aber ein «kleineres»[20] Neuron auf das Symbol | und ein anderes kleines Neuron auf das Symbol _ selektiv geworden (das L setzt sich aus den Symbolen | und _ zusammen). Daneben gebe es noch sehr viele weitere Neuronen, die andere, mehr oder weniger grosse und mehr oder weniger zusammenhängende Bereiche kontrollieren und auf vorkommende Symbole selektiv geworden sind.

Wie reagiert nun das Netzwerk, wenn ein L angelegt wird? Die Figur 13 erläutert die folgenden Zusammenhänge. Zunächst öffnen drei Schlösser: auf | , _ und L, weil der lokal gesehene Schlüssel passt. Nun kommen je-

[18] Ein Simplex ist eine geometrische Figur, bei dem sämtliche Distanzen zwischen den Ecken gleich sind.

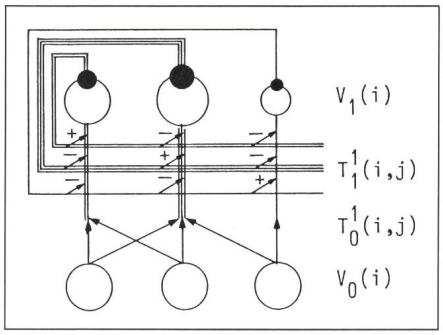
[19] Bildlich gesprochen gibt es im Netzwerk also eine Reihe von unterschiedlich grossen (hohen und breiten) Mexikanerhüten.

[20] «Kleiner» deshalb, weil sein Symbol sich aus weniger Bildpunkten zusammensetzt, es also weniger Eingänge kontrolliert.



Figur 11
Die zweidimensionale Abbildung eines zehndimensionalen Simplex.

Ein Simplex ist eine Struktur, bei der sämtliche Ecken dieselben Abstände voneinander haben. In diesem Beispiel enthält jede Ecke eine gleichverteilte Gruppierung von Punkten mit einer Varianz von 0,1. Der Abstand zwischen den Ecken ist 1,0.



Figur 12 Die Neurale-Netzwerk-Struktur des Masking Fields

Ein Neuron, das mehr Eingänge kontrolliert, bekommt mehr «Macht», d.h. es behindert seine Nachbarn stärker und über eine grössere Distanz.

doch die seitlichen Behinderungen ins Spiel. Weil das grössere Neuron mehr Macht hat, wird es im dynamischen Konkurrenzkampf, der beweisbar stets stabil ist und einer Endlösung zustrebt, siegen. Nach dem Einschwingen wird nur noch dasjenige Neuron aktiv sein, das den vollen Buchstaben dekodiert; diejenigen Neuronen, die hierarchisch untergeordnete Bestandteile des Buchstabens dekodieren, werden nicht mehr aktiv sein.

Die Funktion des Masking Fields besteht also darin, den höchsten im Eingangsmuster enthaltenen *Abstraktionsgrad* zu erkennen. Diese Neurale Netzwerk-Struktur ist von grossem Interesse, weil sie eine Funktion generiert, die für Lebewesen notwendig ist. Beispielsweise erkennt ein Mensch, wenn er die Strasse überqueren will und ein Auto auf ihn zukommen sieht, nicht die Einzelheiten des Autos. Wohl werden diese von den optischen Zellen erfasst, aber es kommt in diesem Fall darauf an, das Auto als *Ganzes* zu erkennen, damit die richtigen Folgeassoziationen ausgelöst werden (in diesem Fall z.B.: Achtung, Gefahr; Warten!). Weiteres Nachdenken führt zur Einsicht, dass solche Maskierungsvorgänge, wie sie das Masking Field erzeugt, im täglichen Leben nahezu ständig vorkommen. Natürlich kann ein Mensch sich auf Einzelheiten konzentrieren; diese Fähigkeit ist aber auf Aufmerksamkeitsprozesse zurückzuführen, welche die vermuteten Masking Fields im Gehirn steuern.

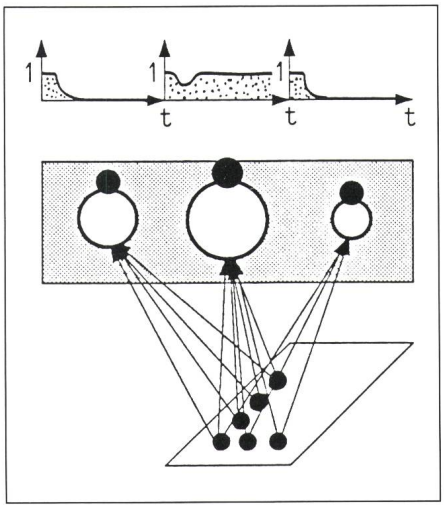
Grossberg [6] hat biologisch recht einleuchtende Wachstumsgesetze aufgestellt, nach denen ein Masking Field derart aufgebaut werden kann, dass Neuronen, die mehr Eingänge kontrollieren, auch grössere seitliche Behinderungen über eine grössere Distanz ausüben können. Ob hingegen ein Masking Field auch topologische Abbildungen erzeugt, ist unsicher, weil trotz der grossen Verwandtschaft der beiden Strukturen nicht jedes Neuron den ganzen Eingangsvektor sieht, was bei den topologischen Abbildungen jedoch vorausgesetzt wird. Weitere Forschungen sollten untersuchen, ob Masking Fields nicht eine allgemeinere Art von topologischen Abbildungen erzeugen, welche eine zusätzliche Dimension der *Abstraktionstiefe* beinhalten. Zum Beispiel könnte man erwarten, dass ähnliche Bestandteile der Abstraktion «Auto» geometrisch nahe beieinander gespeichert sind und ähnliche «Autos» als Ganze beieinander.

Zusammenfassung und Ausblick

Die vorgängig besprochenen vier Neuralen Netzwerkstrukturen zeigen die wesentlichen Erkenntnisse auf, die man bis heute auf diesem relativ jungen Gebiet gewonnen hat. *Multilayer-Perceptrons* eignen sich für assoziative Abbildungen aller Art, sofern man die Signale sinnvoll in Eingangs- und Ausgangsvektoren verpacken kann. Ein überwachtes Training erlaubt dem Netzwerk, die in den Lernbeispielen implizit enthaltenen Regelmässigkeiten herauszufinden. *Hopfield-Netzwerke* eignen sich für inhaltsadressierbare assoziative Speicher und zur raschen Lösung von Problemen aller Art, sofern man das Problem in einer «energieminimierenden» Form schreiben kann (Beispiel: Kaufmann, der seine Städte auf dem kürzesten Weg besuchen will). Kohonens topologische Abbildungen ordnen die erfahrenen Vektoren nach ihrer Ähnlichkeit auf selbstorganisierende Weise zusammen, so dass ihre stochastischen Eigenschaften (Statistik) optimal erhalten und repräsentiert werden. *Masking Fields* erkennen komplexere Objekte als Ganzes im eingegebenen Muster und maskieren deren Einzelteile.

Der Vollständigkeit halber werden hier noch zwei weitere Aspekte beleuchtet: Der erste Aspekt handelt von *zeitabhängigen* Signalen. Die erwähnten Modelle verarbeiteten vorerst nur einzelne Vektoren oder Muster. Es stellt sich nun z.B. die Frage, wie ein Masking Field zeitabhängige Signale erkennen könnte. Als Beispiel diene die Spracherkennung: Wie können die einzelnen Laute durch ganze Wörter maskiert werden? Und wie extrahiert man den Sinn aus einem ganzen Satz? Offensichtlich sind zeitliche Maskierungsvorgänge am Werk, wenn der Mensch die gesprochene Meinung seines Gegenübers zu erkennen versucht. *Grossberg* [6] versucht eine Vorverarbeitung zu entwickeln, welche die Zeitachse sozusagen in eine örtliche Achse abbildet. Dabei geht er aber nicht davon aus, dass sich zeitlich eingefrorene Muster in einem sequentiellen Speicher bewegen würden. Es werden Aufmerksamkeitsprozesse einbezogen, die das Muster der aktivierten Schlösser (Merkmaldetektoren) fortlaufend beeinflussen. Eine grössere, d.h. weniger behinderte Aktivität von Neuronen stellt dann ein aktuelleres (auf der Zeitachse später auftauchendes) Signal dar. Die Schwächung früherer Signale hat natürlich auch etwas mit Vergessen zu tun.

Der zweite Aspekt betrifft die Stabilität des Lernvorgangs. Der unüberwachte Lernalgorithmus, der im Kapitel der topologischen Abbildungen erwähnt ist und auch beim Masking Field angewendet wird, hat den Nachteil, dass die Gewichte aller Neuronen, die auch nur ein wenig aktiv sind, verändert werden. Es fehlt also ein Schutzmechanismus, der die Veränderung eines Schlosses dann verhindert,



Figur 13 Beispiel einer Masking-Field-Funktion

Das grösste Neuron, das den vollen Buchstaben «L» kodiert, bleibt nach der Einschwingphase aktiv, während die kleineren Neuronen, die Bestandteile des vollen Buchstabens kodieren, nur anfänglich aktiv sind und dann durch die seitlichen Behinderungen gehemmt werden.

wenn es nicht das zum aktuellen Schlüssel am besten passende ist. Die Kernidee von *Grossberg* [6] ist in diesem Zusammenhang, Rückkopplungen vom Schloss zum Schlüssel einzubauen. Ein aktiviertes Schloss sendet dabei seinen gelernten Schlüssel an diejenige Schicht von Neuronen zurück, die den Schlüssel repräsentieren. Aufmerksamkeitsprozesse vergleichen den aktuellen mit dem erwarteten Schlüssel und lähmen das nicht passende Schloss blitzschnell. Dies hat zur Folge, dass ein Schloss, das nicht genau passt, nicht mehr aktiv bleibt, was zwei Vorteile einbringt: Einerseits wird sein Gewichtsvektor vom nichtpassenden Schlüssel nicht verfälscht, und andererseits beeinflusst das falsche Schloss die dynamische Ermittlung des bestpassenden Schlosses nicht mehr. Das System schwingt jetzt in einer Resonanzphase auf das richtige Schloss ein: Der Schlüssel speist das Schloss, und das Schloss speist den Schlüssel. Dabei findet eine Adaptierung der Gewichtsvektoren von Schloss und erwartetem Schlüssel statt. Diesen Mechanismus nennt man «Adaptive Resonanz-Theorie». Sie kann z.B. auf ein Masking Field angewendet werden. Dieses ist damit stets bereit, neue Informationen zu lernen, während die alte (in einem anderen Kontext wesentliche) Information geschützt bleibt. Im Gegensatz zu Strukturen ohne Adaptive Resonanz kann eine neue Erkennungskategorie (ein selektives Schloss) ohne langwieriges Wiederholen des Trainingssatzes direkt eingebaut werden, sofern dem Schlüssel genügend *Aufmerksamkeit* geschenkt wird. Sein Gewichtsvektor kann sich dann stärker anpassen, als es bei weniger wichtigen Ereignissen nötig ist. Eine ähnliche Funktion haben Hormone wie Adrenalin bei Lebewesen, welche diese befähigen, unter Gefahr besonders gut zu lernen und bei einer späteren erneuten Gefährdung entsprechend rasch zu reagieren. So ausgestattete Neurale Netzwerk-Strukturen können als stabile, lernende Erkennungsmaschinen bezeichnet werden, die dem autonomen Nervensystem der Lebewesen ein gutes Stück näherkommen dürften.

Literatur

[1] *T. Kohonen:* Self-organization and associative memory. - Springer series in information science, vol. 8 - Berlin a. o., Springer-Verlag, 1984.

[2] *J. McClelland* and *D. Rumelhart:* Parallel distributed processing: Explorations in the microstructure of cognition. 2 volumes. Cambridge/Mass., Massachusetts Institute of Technology, 1986.

[3] *T. Sejnowski* and *C. Rosenberg:* NETtalk: A parallel network that learns to read aloud. Electrical engineering and computer science technical report JHU/EECS-86/01. Baltimore, John Hopkins University, 1986.

[4] *J.J. Hopfield:* Neural networks and physical systems with emergent collective computational abilities. Proceedings of the National Academy of Sciences of the USA, 79(1982)7, p. 2554...2558.

[5] *J.J. Hopfield:* Neurons with graded response have collective computational properties like those of two-state neurons. Proceedings of the National Academy of Sciences of the USA, 81(1984)10, S. 3008...3092.

[6] *S. Grossberg:* The adaptive brain. 2 volumes. Amsterdam a. o., North-Holland, 1987.