2.4 Hopfield-Netze

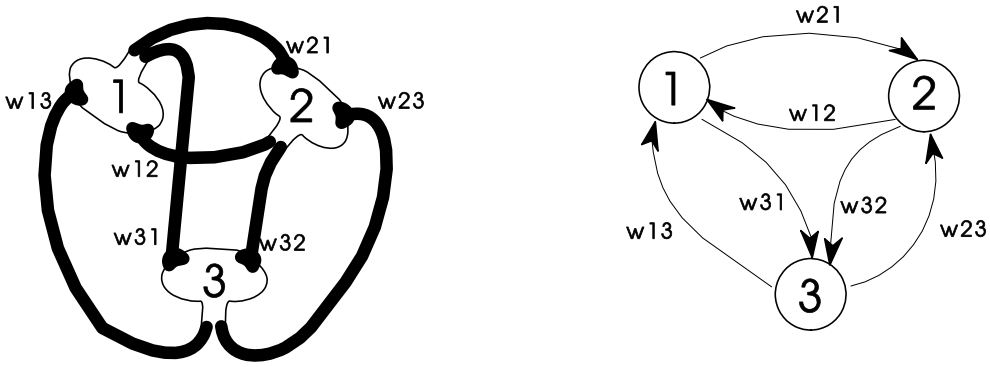


Abb. 7: Einfaches Hopfield-Netz mit drei Neuronen.

Charakteristisch für Hopfield-Netze ist, daß jede Einheit (Neuron) mit jeder anderen Einheit des Systems "symmetrisch" verbunden ist (aber nicht mit sich selbst). Für die *Verbindungsmatrix* wij gilt: wij =wji , $w_{ii} = 0$. Diese Bedingungen sichern ein "stabiles Verhalten" für das System, sind aber biologisch nicht motiviert.

Aktivierungsausbreitung (*Updating*)

Vorausgesetzt werden binäre Zustände: S = {-1, 1}.

Synchrones *Updating*:

$r_i(s_1, ..., s_n) = f(\Sigma_j w_{ij} s_j)$ für i = 1, …, n

dabei ist f die Schwellenfunktion

$$f(x) = \begin{cases} 1, \text{ falls } x \geq 0 \\ -1, \text{ sonst} \end{cases}$$

Berechnungsbeispiel (verwende Matrixschreibweise):

$$f\left(\begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}\right) = f\left(\begin{pmatrix} 2 \\ 2 \\ 2 \end{pmatrix}\right) = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

$$f\left(\begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}\begin{pmatrix} -1 \\ -1 \\ 1 \end{pmatrix}\right) = f\left(\begin{pmatrix} 0 \\ 0 \\ -2 \end{pmatrix}\right) = \begin{pmatrix} 1 \\ 1 \\ -1 \end{pmatrix}$$

$$f\left(\begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}\begin{pmatrix} 1 \\ 1 \\ -1 \end{pmatrix}\right) = f\left(\begin{pmatrix} 0 \\ 0 \\ 2 \end{pmatrix}\right) = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

Asynchrones *Updating:*

$$r_i(s_1, ..., s_n) = \begin{cases} f(\Sigma_j w_{ij} \; s_j), \text{ falls } i = \text{random}(1, n) \\ r_i(s_1, ..., s_n), \text{ sonst} \end{cases}$$

Die Veränderungen der Aktivierung eines Knotens werden sofort berücksichtigt und können die Ausbreitung der Aktivierung unmittelbar beeinflussen.

Lernstrategie

Seien $\mathbf{s}^\alpha$ ($\alpha = 1, ..., N$) die Inputvektoren. Hopfield benutzte die generalisierte Hebbsche Regel als Lernstrategie. Betrachte die j. Synapse des i. Neurons:

$\Delta w_{ij} = \mu \, s^\alpha_j \, r^\alpha_i = \mu \, s^\alpha_j \, s^\alpha_i$ (setze $r^\alpha_i = s^\alpha_i$)

Die im wesentlichen eindeutig bestimmte Verbindungsmatrix läßt sich direkt berechnen:

$w_{ij} = \Sigma_\alpha \, s^\alpha_i \; s^\alpha_j$

Beispiel: Sei nur ein Inputvektor gegeben, nämlich (1 1 1). Dann ergibt sich folgende Verbindungsmatrix:

$$\begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$$

Übung 3: Belehre ein Hopfield-System mit den Mustern (1 1 1) und (-1 -1 -1). Unterscheidet sich das Verhalten des Netzes von dem oben angeführten Beispiel? Bleibt das Muster (-1 -1 -1) bei synchronem *Updating* stabil?

Eigenschaften von Hopfield-Netzen

Hopfield-Netze können als "intelligente" Speicher angesehen werden. In der "Lernphase" eingegebene Muster werden gespeichert (es dürfen nicht zu viele Muster sein und sie müssen sich hinreichend deutlich voneinander unterscheiden). In der Kann-Phase werden diese Muster "wiedererkannt" (System anwortet mit identischer Ausgabe: Resonanz). Das System besitzt ein assoziatives Gedächtnis n unvollständige oder gestörte Muster werden in Hinblick auf die gespeicherten Muster korrigiert (*Mustervervollständigung*).

Abb. 8: Mustervervollständigung: WORR oder WORK?

Intermezzo: Stabilitätstheorie (Ljapunov 1892)

Betrachte ein *dynamisches System* [**S**, $\mathbf{r}^n$]. Dabei ist **S** ein Zustandsraum und die Funktion $\mathbf{r}^n(\mathbf{x})$ beschreibt die zeitliche Entwicklung der Zustände $\mathbf{x} \in \mathbf{S}$, d.h. $\mathbf{r}^n(\mathbf{x})$ beschreibt den Zustand, der zur Zeit n eingetreten ist.

- Ein Zustand **s** heißt *Gleichgewichtszustand*, falls sich **s** zeitlich nicht mehr ändert.
- Ein Gleichgewichtszustand heißt *stabil*, falls eine kleine Änderung dieses Zustands keinen beträchtlichen Einfluß auf dessen zeitliche Entwicklung nimmt.
- Ein Gleichgewichtszustand heißt *asymptotisch* stabil, wenn nach einer kleine Änderung dieses Zustands der Zustand sich zeitlich-asymptotisch wieder dem ursprünglichen Zustand annähert.

Abb. 9: Stabile, asymptotisch stabile und instabile Zustände

DEFINITION: Sei [**S**, $\mathbf{r}^{n]}$ ein dynamisches System. Sei **s** ein Gleichgewichtszustand, d.h. $\mathbf{r}^n(\mathbf{s}) = \mathbf{s}$ für alle Zeitpunkte $n \geq 0$

(a) $\mathbf{s} \in \mathbf{S}$ heißt stabil gdw. für jedes $\varepsilon > 0$ ein $0 < \delta \leq \varepsilon$ existiert, sodaß für einen beliebigen Zustand $\mathbf{s}' \in \mathbf{S}$ mit $|\mathbf{s}'-\mathbf{s}| < \delta$ gilt: $|\mathbf{r}^n(\mathbf{s}')-\mathbf{s}| < \varepsilon$ (für alle Zeitpunkte $n \geq 0$).

(b) $\mathbf{s} \in \mathbf{S}$ heißt asymptotisch stabil gdw. **s** ist stabil und ist folgende Bedingung erfüllt: für ein gemäß (a) existierendes δ und ein beliebiges **s'** mit $|\mathbf{s}'-\mathbf{s}| < \delta$ gilt: $\lim_{n\to\infty} \mathbf{r}^n(\mathbf{s}') = \mathbf{s}$.

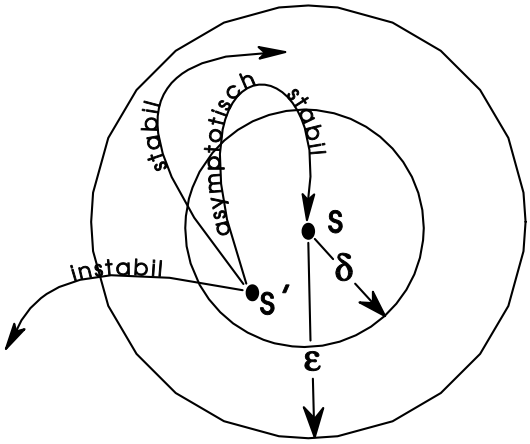
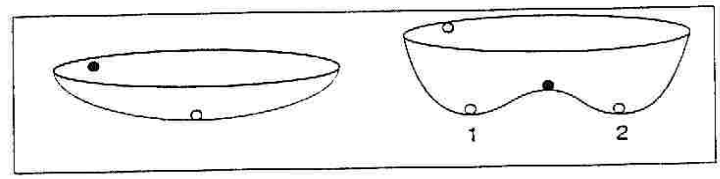
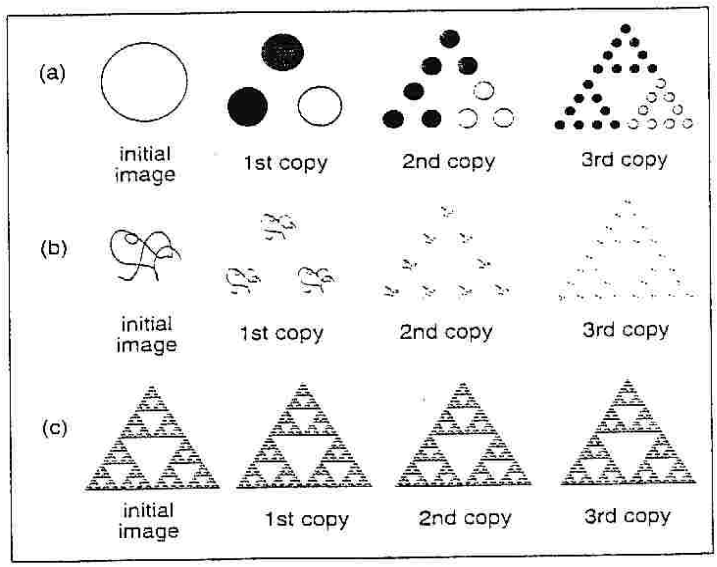


Abb. 10: Zur Definition stabiler, asymptotisch stabiler und instabiler Zustände

Asymptotisch stabile Gleichgewichtszustände heißen auch *Resonanzen* oder *Attraktoren*. Der Begriff Resonanz wird vorwiegend in der konnektionistischen Literatur verwendet, der Begriff Attraktor in der Theorie nichtlinearer dynamischer Systeme (Chaostheorie, fraktale Geometrie). Dort wird der Begriff wesentlich verallgemeinert, um die sogen. "strange attractors" zu erfassen, d.h. Attraktoren, die eine fraktale Struktur besitzen.





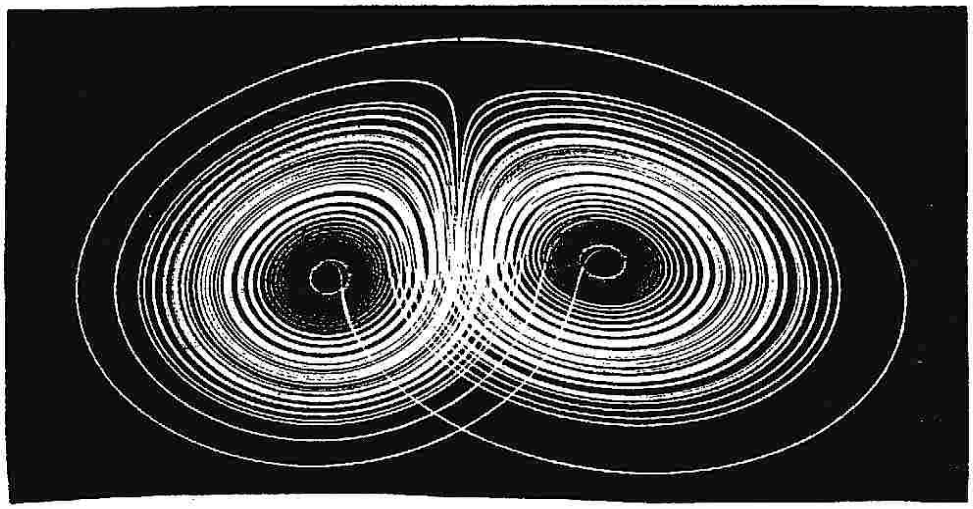


Abb.11: Beispiele für Attraktoren (Oben: Schalen mit einem bzw. zwei Attraktoren; Mitte: Beispiel für Sierpinski-Dichtung; Unten: Beispiel für "seltsamen" Attraktor (Lorenz-Attraktor)

DEFINITION: Sei $[\mathbf{S}, \mathbf{r}^{n]}$ ein dynamisches System. Eine Funktion E von **S** in die reelen Zahlen heißt *Ljapunovfunktion* für das dynamische System gdw. $E(\mathbf{r}^n(\mathbf{s}))$ monoton mit wachsender Zeit fällt, d.h. für beliebige Zustände **s** und Zeiten n, n'gilt: falls n'>n, dann $E(\mathbf{r}^{n'}(\mathbf{s})) \leq E(\mathbf{r}^n(\mathbf{s}))$.

STABILITÄTSTHEOREM VON LJAPUNOV: Sei E eine Ljapunovfunktion für das dynamische System $[\mathbf{S}, \mathbf{r}^n]$. Besitzt die Funktion E für **s** ein lokales Minimum, dann ist **s** stabil.

Einige mathematische und experimentelle Ergebnisse für Hopfield-Netze

Sei $\mathbf{S} = S^n$ der Zustandsraum, S = {-1, 1}. Sei $\mathbf{r}(\mathbf{s})$ die Input-Output-Funktion des Netzes. $\mathbf{r}^1(\mathbf{s}) =_{def} \mathbf{r}(\mathbf{s})$, $\mathbf{r}^n(\mathbf{s}) =_{def} \mathbf{r}^{n-1}(\mathbf{s})$. Dann heißt $[\mathbf{S}, \mathbf{r}^{n]}$ ein Hopfield-System.

(1) Die Funktion $E(s_1, ..., s_n) = - \frac{1}{2} \Sigma_{ij} w_{ij} s_i s_j$ ist eine Ljapunovfunktion für das Hopfield-System.

Beweis: Betrachte asynchrones Updating; untersuche Updating an Einheit 1:

(i) $r_1(s_1, ..., s_n) = f(\Sigma_j w_{1j} s_j)$

Die Aktivierungen der übrigen Einheiten ändern sich nicht.

Fall 1: $s_1 = 1$ und der Netzinput für die 1. Einheit $\Sigma_j w_{1j} s_j$ ist positiv (oder gleich Null). Dann ändert sich s_1 nicht. Auch die Ljapunov-Funktion

(ii) $E(s_1, ..., s_n) = - \frac{1}{2} \Sigma_{ij} w_{ij} s_i s_j$

bleibt unverändert.

Fall 2: $s_1 = -1$ und der Netzinput für die 1. Einheit $\Sigma_j w_{1j} s_j$ ist negativ. Dann ändert sich s_1 nicht und auch die Ljapunov-Funktion bleibt unverändert.

Fall 3: $s_1 = 1$ und der Netzinput für die 1. Einheit $\Sigma_j w_{1j} s_j$ ist negativ. Die Aktivierung der 1. Einheit ändert sich von +1 auf -1. Alle übrigen Einheiten bleiben unverändert. Die Veränderung der Ljapunov-Funktion kann nur von einer Veränderung der Summe $- \frac{1}{2} (\Sigma_j w_{1j} s_1 s_j + \Sigma_i w_{i1} s_i s_1) = - s_1 \Sigma_j w_{1j} s_j$ herrühren.
Da sich s_1 von +1 auf -1 ändert, aber der Ausdruck $\Sigma_j w_{1j} s_j$ unverändert bleibt (nur js mit j>1 tragen zur Summe bei!), ergibt sich eine Veränderung der Ljapunov-Funktion um den Wert $+2 \Sigma_j w_{1j} s_j$. Da der Summand negativ ist, verringert sich der Wert der Ljapunov-Funktion.

Fall 4: $s_1 = -1$ und der Netzinput für die 1. Einheit $\Sigma_j w_{1j} s_j$ ist positiv (oder gleich Null). Diesmal ändert sich die Aktivierung der 1. Einheit von -1 auf +1. Die Veränderung der Ljapunov-Funktion kann nur von einer Veränderung der Summe $- \frac{1}{2} (\Sigma_j w_{1j} s_1 s_j + \Sigma_i w_{i1} s_i s_1) = -s_1 \Sigma_j w_{1j} s_j$ herrühren.
Da sich s_1 von -1 auf +1 ändert, aber der Ausdruck $\Sigma_j w_{1j} s_j$ unverändert bleibt, ergibt sich eine Veränderung der Ljapunov-Funktion um den Wert $- \frac{1}{2} \Sigma_j w_{1j} s_j$. Da der Summand positiv (oder gleich Null) ist, verringert sich der Wert der Ljapunov-Funktion (oder bleibt unverändert).

(2) Die Muster $\mathbf{s}^{\alpha}$ (α = 1, ..., N), mit denen belehrt wurde (N << n), bilden Resonanzen des entstandenen Netzwerks (Beweis: Cohen & Grossberg 1983). Sie sind jedoch nicht die einzigen Resonanzen des Systems.

- Der Zustand (-1, ..., -1) ist immer eine Resonanz
- Ist **s** eine Resonanz, dann ist auch **-s** eine Resonanz

(3) Für beliebige Zustände **s'** existiert ein Endzustand der Aktivierungsausbreitung (Updating), $\lim_{n\to\infty} \mathbf{r}^n(\mathbf{s}')$. Dieser Zustand ist eine Resonanz (entweder eine "gelernte Resonanz" oder eine "Nebenresonanz" (Cohen & Grossberg 1983).
Anmerkung: Derartige Systeme heißen *Resonanzsysteme*. Neben dem Hopfield-System gibt es folgende Beispiele für Resonanzsysteme: McCulloch-Pitts (1943)-Modell, Cohen-Grossberg (1983)-Modell, Andersons (1977) BSB-Modell, Koskos (1987) BAM-Modell, Boltzmann-Maschine.

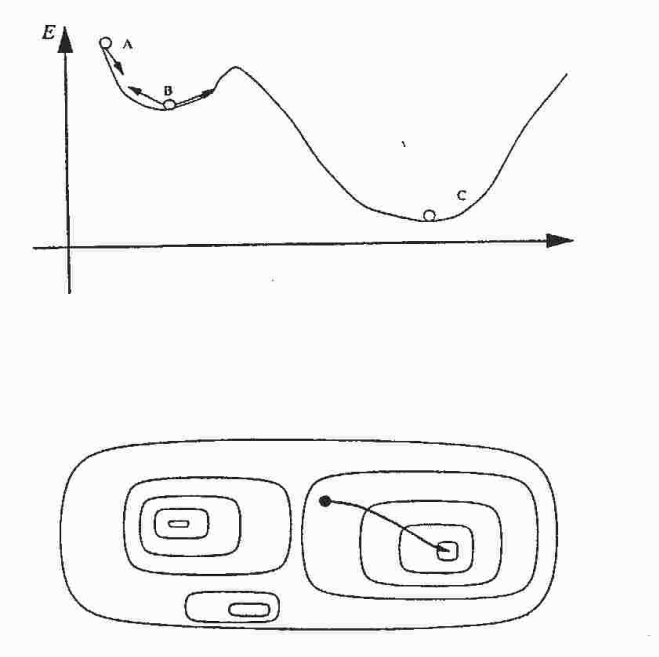


Abb. 12: *Updating* als Talwanderung

(4) Experimentelle Befunde (Computersimulation):
- Werden weniger als 0.15 n Muster gelernt, so besitzt das System in der Kann-Phase eine gute Assoziationsfähigkeit. Werden mehr als 0.15 n Muster gelernt, so ergeben sich immer mehr Nebenresonanzen, sodaß die Einzugsbereiche der gelernten Resonanzen gegen Null gehen. Entsprechende Testmuster können nur noch völlig zufällig vervollständigt werden.

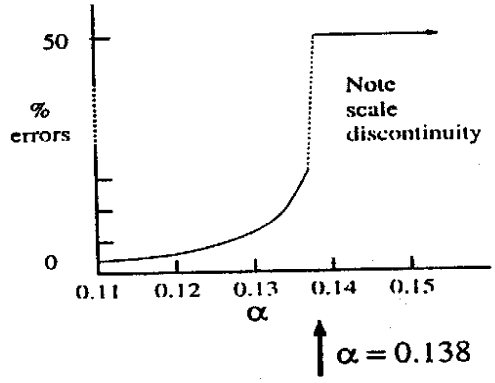


Abb. 13: Wenn zu viele Muster in einem Hopfield-Netz gespeichert werden, gibt es einen „Phasenübergang“. Der Graph zeigt die Fehlerrate in Abhängigkeit von der Zahl der gespeicherten Muster.

Boltzmann-Maschine als Weiterentwicklung von Hopfield-Systemen

- Die Topologie der Boltzmann-Maschine ist identisch mit der Topologie von Hopfield-Systemen. Jedoch besteht bei der Boltzmann-Maschine eine Unterscheidung zwischen "sichtbaren" und "verborgenen" (*hidden*) Einheiten. Die sichtbaren Einheiten sind unterschieden in Ein- und Ausgabeeinheiten. Die verborgenen Einheiten können nicht zur Ein- und Ausgabe benutzt werden.

- Verborgene Einheiten erhöhen die Abstraktionsleistungen des Netzes. Sie können strukturelle Abhängigkeiten zwischen den Komponenten der Eingangsvektoren entdecken.

- Stochastisches *Update*: An Eingabeeinheiten liegt permanent die Inputaktivierung an. Gesucht sind die Aktivierungen der Ausgabeeinheiten. Bei Fragestellungen dieser Art ist es sinnvoll nach globalen Minima der Energie-/Ljapunov-Funktion zu suchen.
  Um zu vermeiden, daß das Netz unweigerlich immer nur auf der Oberfläche der "Energielandschaft" bergab läuft, muß die Möglichkeit eingeräumt werden, daß auch gelegentlich ein Weg bergauf eingeschlagen wird. Das geschieht dadurch, daß eine Einheit, deren Netzinput über dem Schwellenwert liegt, nicht in jedem Falle feuert, sondern nur gelegentlich - mit einer gewissen Wahrscheinlichkeit p. Feuert eine Einheit nicht, obwohl der Netzinput über dem Schwellenwert liegt, dann wird ein Weg eingeschlagen, der bergauf führt.

  Die Wahrscheinlichkeit p für einen Outputwert $r_i=1$ ergibt sich aus der sogen. Boltzmann-Verteilung und ist folgendermaßen bestimmt:

  $p(r_i(s_1, ..., s_n)=1) = 1/(1+\exp(-(\Sigma_j\ w_{ij}\ s_j)/T)$ (T Konstante, "Temperatur")

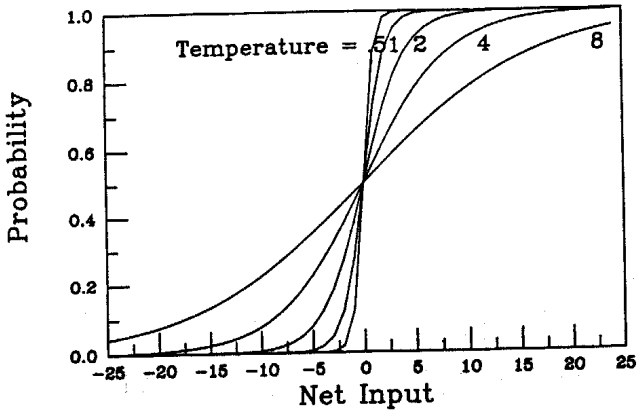


Abb. 14: Wahrscheinlichkeit für r_i = 1 bei gegebenem Netzinput $\Sigma_j\ w_{ij}\ s_j$ und verschiedenen "Temperaturen" T.

  Simuliertes Auskristallisieren: Stochastisches *Update*, wobei die "Temperatur" allmählich von einem bestimmten Ausgangswert bis auf Null herabgesetzt wird. (Für T=0 geht das stochastische *Update* in deterministisches *Update* über). Das Verfahren führt zu einem global-minimalen Outputvektor.

- Lernstrategie: Überwachtes Lernen. Einem Inputvektor ist jeweils ein *bekannter* Outputvektor zuzuordnen. Unbekannt sind jedoch die Werte für die verborgenen Einheiten. Diese Gewichte werden entsprechend eines *maximum likelyhood*-Prinzips bestimmt.

2.5 *Feedforward*-Netze

In *Feedforward*-Netzen sind die Einheiten jeweils bestimmten Ebenen zugeordnet. Die Ebenen sind linear geordnet. Nur Elemente benachbarter Ebenen sind miteinander verknüpft. Die Verbindung führt immer von der niederen zur nächsthöheren Ebene. *In Feedforward*-Netze mit N Ebenen ist der Endzustand der Aktivierung nach N-1 *Update*-Schritten erreicht.

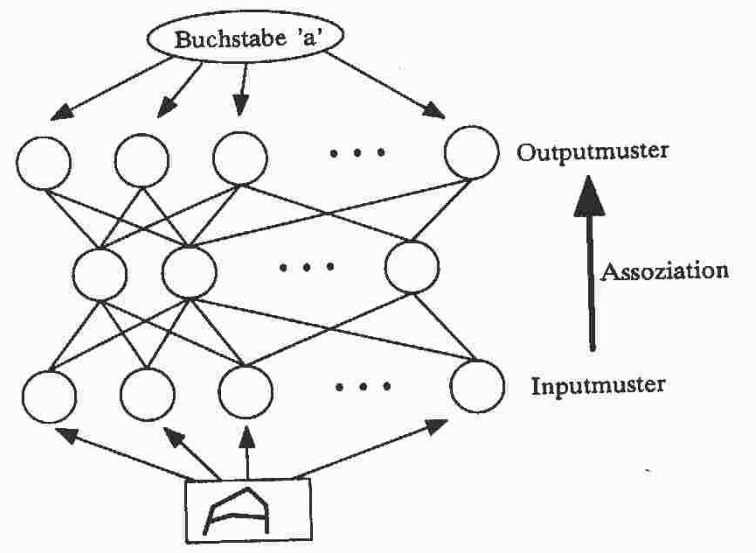


Abb. 14: *Feedforward*-Netz mit drei Ebenen von Einheiten (Inputeinheiten, verborgene Einheiten, Outputeinheiten)

Zum Verhältnis von symbolischen Elementen und Zuständen neuronaler Netze: Lokale und verteilte Repräsentation

Wie lassen sich symbolische Elemente in Netzen darstellen?
- Lokale Repräsentation: Symbole entsprechen den Zuständen einzelner Einheiten des Netzes.
- Verteilte Repräsentation: Symbole entsprechen den kollektiven Zuständen einer Gruppe von Einheiten. Die Zustände einzelner Einheiten sind nicht durch Symbole irgendeiner Art interpetierbar.

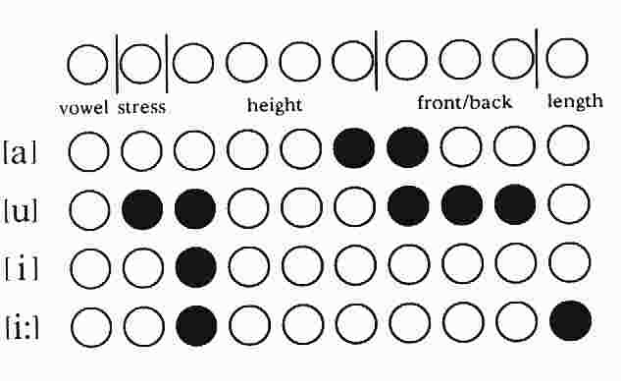


Abb. 15: Verteilte Repräsentationen für einige Vokale. Welche Einheiten lassen sich als (symbolische) Merkmalsstrukturen deuten?

Perzeptron (Rosenblatt 1958)

ARCHITEKTUR:

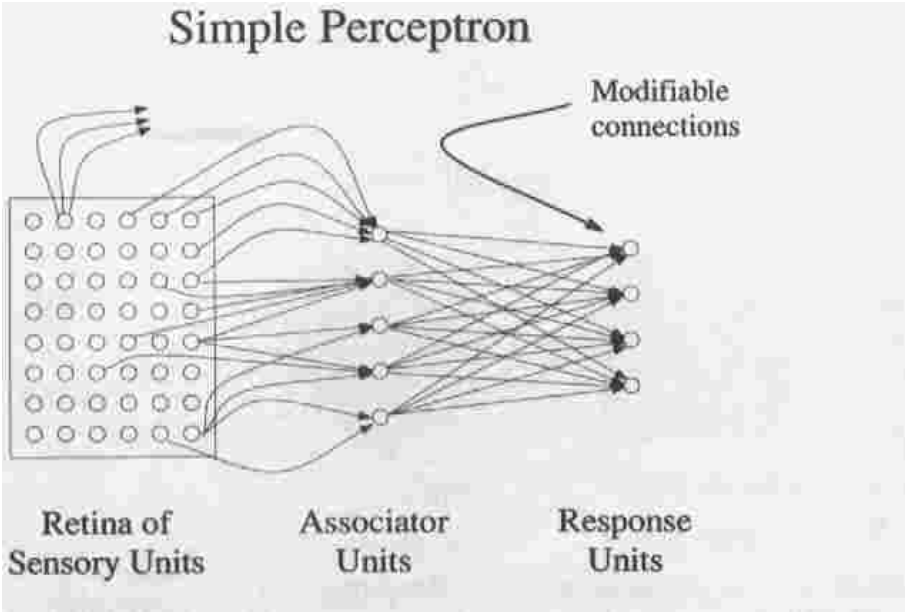


Abb. 16: Architektur der Perzeptrons (vereinfachte Grundversion)

- Feedforward-Netzwerk mit drei Ebenen: S (sensorische Ebene), A (Assoziatorebene), R (Responsebene).
- partielle Verbindung zwischen S- und A-Ebene, Vollverbindung von Ebene A zu Ebene S. Elemente der Ebene S realisieren kontinuierliche Zustände ("Grauwerte"), Elemente der Ebenen A und R realisieren diskrete Zustände (0, 1).
- Verbindungsstärken zwischen S und A fest vorgegeben. Nur die Synapsen der R-Einheiten können durch Lernen beeinflußt werden.

AKTIVIERUNGSAUSBREITUNG (*UPDATE*):

Angenommen auf A-Ebene ist ein bestimmtes Aktivierungsmuster erzeugt worden. Welches Antwort-"Muster" entsteht auf R-Ebene (beachte: Perzeptrons wird üblicherweise als Klassifikator eingesetzt, das Antwort-"Muster" ist also oft binär: ja/nein)

Betrachte n A-Einheiten und m S-Einheiten, wobei jede A-Einheit mit jeder S-Einheit verbunden ist. Sei w_{ij} die Verbindungsstärke der j. A-Einheit mit der i. R-Einheit (Anmerkung zur Numerierung: die j. A-Einheit ist an die j. Synapse der entsprechenden S-Einheit angekoppelt).

$$r_i(s_1, ..., s_n) = f(\Sigma_j w_{ij} s_j)$$

dabei ist f die Schwellenfunktion

$$f(x) = \begin{cases} 1, \text{ falls } x \geq \theta \\ 0, \text{ sonst} \end{cases}$$

LERNEN:

Überwachtes Lernen mit Lehrer: "Synapsen" der R-Einheiten werden nach Delta-Regel verändert.

$\Delta w_{ij} = \mu \, s_j \, (r'_i - r_i)$

Dabei ist ist Δw_{ij} die Veränderung des Gewichts der j. Synapse an der i. R-Einheit, hervorgerufen durch den Unterschied des Sollwerts r'_i vom Istwert r_i an der i. R-Einheit.

EIGENSCHAFTEN:

- Perzeptron als Klassifikator: Nur linear trennbare Muster können unterschieden werden.

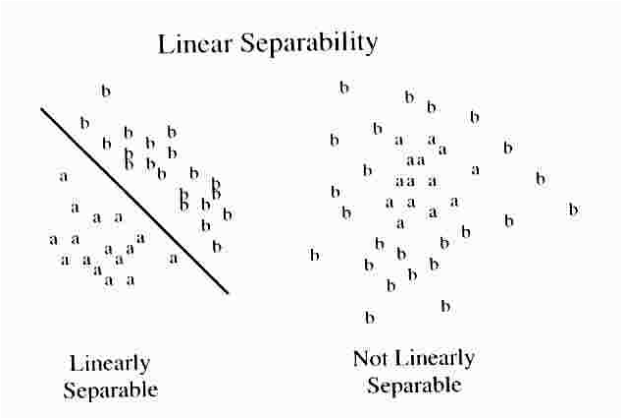


Abb. 17: Lineare Trennbarkeit

- Konvergenztheorem für Perzeptrons: Falls eine Verbindungsmatrix w_{ij} existiert, die einem gewünschten Input-Output-Verhalten entspricht, dann kann das Verhalten gemäß der Perzeptron-Lernregel erlernt werden.

- Von Perzeptrons nicht realisierbare Verhaltensweisen:
  - exklusives Oder
  - geometrisches Prädikat der Verbundenheit von Figuren

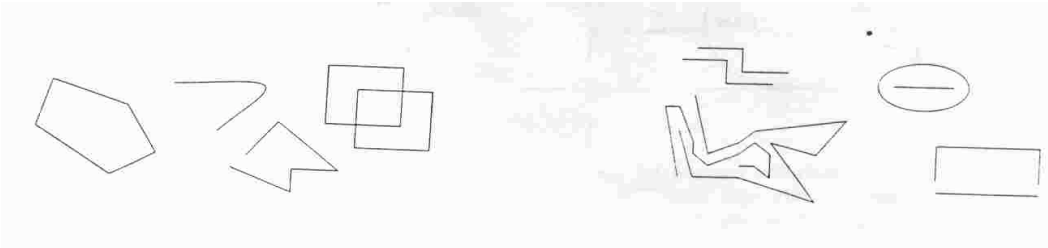


Abb. 18: Beispiele für verbundene (links) und unverbundene Figuren (rechts)

- Vermutung von Minsky & Papert (1969, Seite 231-232): "Das Problem der Erweiterung auf mehrerschichtige Perzeptrons ist nicht allein technischer Art. Es hat auch einen strategischen Aspekt. Das (klassische) Perzeptron selbst ist der

Untersuchung wert - trotz seiner ernsthaften Beschränktheit. Es hat zahlreiche attraktive Eigenschaften: seine Linearität, ein faszinierendes Lerntheorem, Einfachheit des Paradigmas als parallelverarbeitendes System. Es gibt keine Gründe anzunehmen, daß diese Vorzüge beim Übergang zu mehrschichtigen Perzeptrons erhalten bleiben. Trotzdem betrachten wir es als ein wichtiges Forschungsproblem, unsere Vermutung, daß diese Erweiterung fruchtlos ist, zu erhärten (oder zu widerlegen)."

- Diese Vermutung hat sich als falsch erwiesen.

Feedforward-Netze mit verborgenen Einheiten und *backpropagation*

Das bekanntesten Beispiel ist *Nettalk* von Sejnowski & Rosenberg (1986).
Das Assoziationsnetz zum Erlernen der Vergangenheitsformen englischer Verben (Rumelhart & McClelland 1985) enthält keine verborgenen Einheiten.

VERBORGENE EINHEITEN UND DIE XOR-FUNKTION

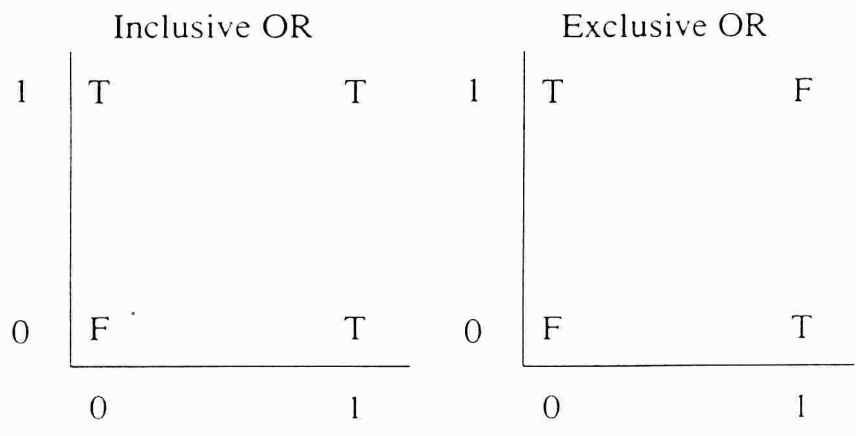


Abb. 19: Darstellung der Wahrheitsfunktionen für inklusives und exklusives Oder.

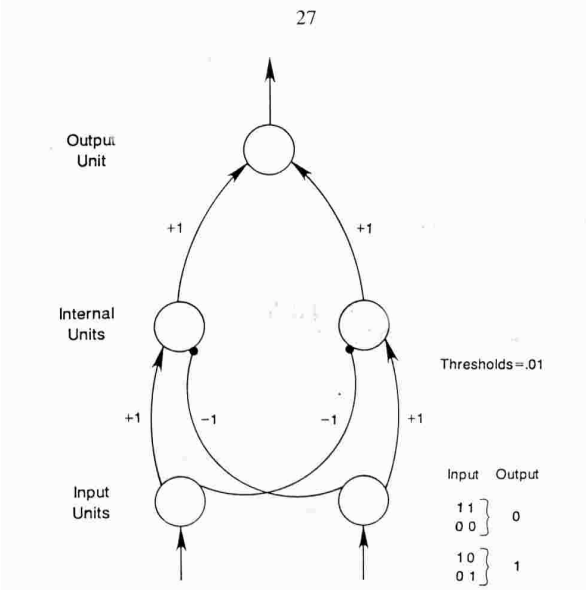
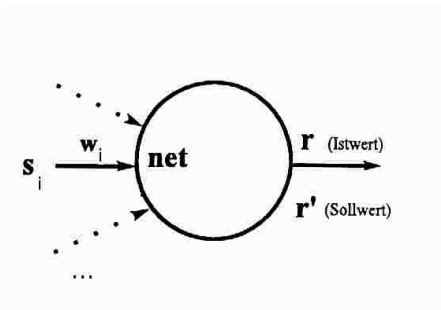


Abb. 20: Einfaches Netzwerk zur Realisierung der XOR-Funktion. Das Netz enthält zwei verborgene Einheiten.

GENERALISIERTE DELTA-REGEL (Rumelhart et al. 1986)

Bei der einfachen Delta-Regel wird für jeden Output die Abweichung vom Sollwert bestimmt. Proportional zum berechneten Fehler erfolgt die Anpassung der Gewichte.



$\Delta w_i = \mu\, s_i\, \delta$ mit $\delta = (r'-r)$

Eine verbesserte Form dieser Regel ergibt sich (Verstärkung des Korrektureffekts), wenn für folgender Ansatz verwendet wird:

$\Delta w_i = \mu\, s_i\, \delta$ mit $\delta = f'(net)\,(r'-r)$, wobei f' der Anstieg (erste Ableitung) der Funktion f ist. Für die Sigmoid-Funktion ist $f' = f(1-f)$.

Dieser Ansatz ergibt sich, wenn man die Gradientenmethode verwendet, um den Ausdruck für die Standartabweichung zu minimieren:

$e = (f(\sum_j w_j s_j) - r')^2$ $\qquad \frac{\partial e}{\partial w_i} = 2(r - r') \cdot f' \cdot s_i$

Die Gradientenmethode verändert die Gewichte in Gradientenrichtung in "kleinen Schritten", die sich aus einem Parameter μ ergeben.

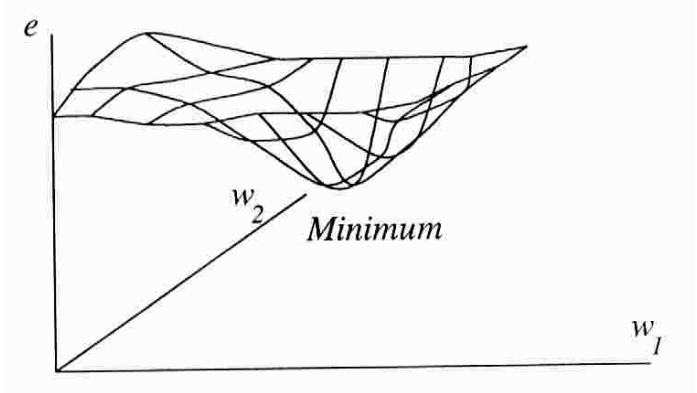
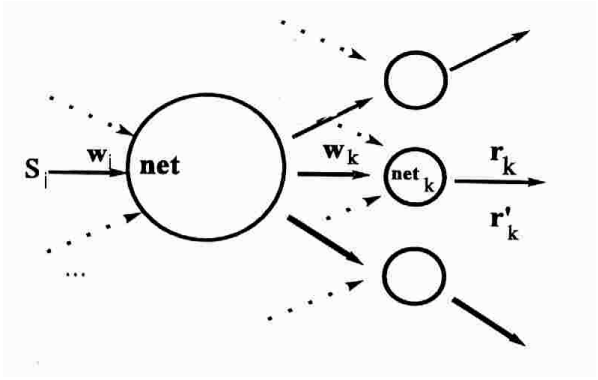


Abb. 21. Hyperfläche der Gewichte zur Illustration der Gradientenmethode

Für verborgene Einheiten ist der Sollwert nicht bekannt. Es ist jedoch möglich, die Fehler in der Output-Schicht zurückzuverfolgen (*backpropagation*) und auf Abweichungen in Elementen der verdeckten Schicht zurückzuführen. Mathematisch läuft dies ebenfalls auf das oben skizzierte Gradientenverfahren hinaus (Minimierung der Standardabweichung). Bei der generalisierten Delta-Regel wird für Elemente der verdeckten Schicht der entsprechende Fehler abgeschätzt und damit die Anpassung der Gewichte ausgeführt.



$\Delta w_i = \mu\, s_i\, \delta$ mit $\delta = f'(net) \cdot \Sigma_k\, \delta_k\, w_k$,
wobei $\delta_k = f'(net_k) \cdot (r_k'-r_k)$

2.6 Übersicht über weitere Netzwerktypen

Netzwerkgrundtypen können hinsichlich vier verschiedener Gesichtspunkte klassifiziert werden: Architektur, *Update*-Strategie, Lern-Strategie und Verhalten der einzelnen Einheiten.

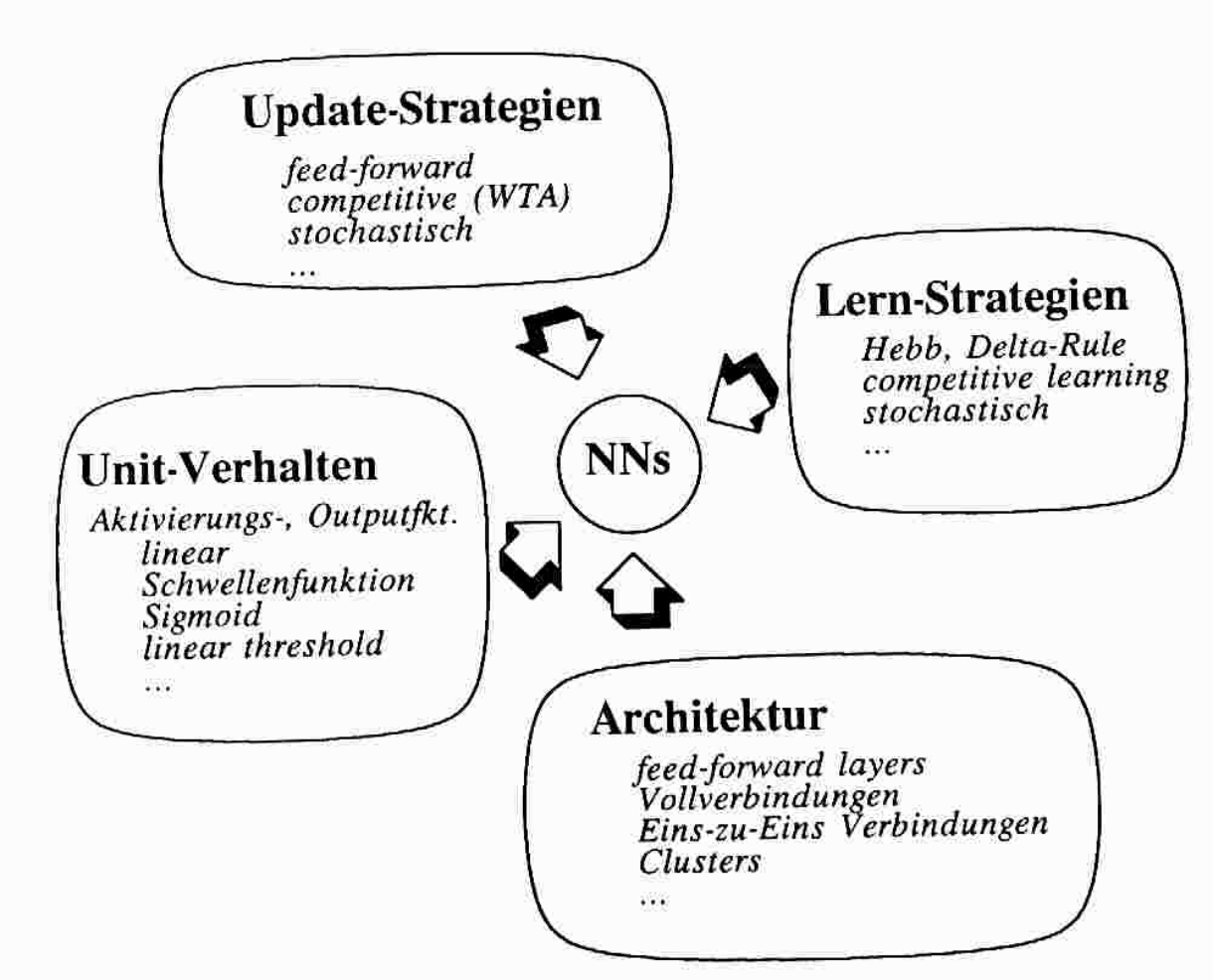


Abb. 22 : Variationen von Netzwerktypen

z.B. Hopfield-Netze:

Architektur	Vollverbindungen
Unit-Verhalten	Einheiten mit Schwellenfunktion
Update-Strategie	synchrones *Update*
Lernstrategie	Hebbsche Regel