Einführung in neuronale Netze

Florian Wenzel

Neurorobotik
Institut für Informatik
Humboldt-Universität zu Berlin

1. Mai 2012

Überblick

Überblick

Überblick

Überblick

Überblick

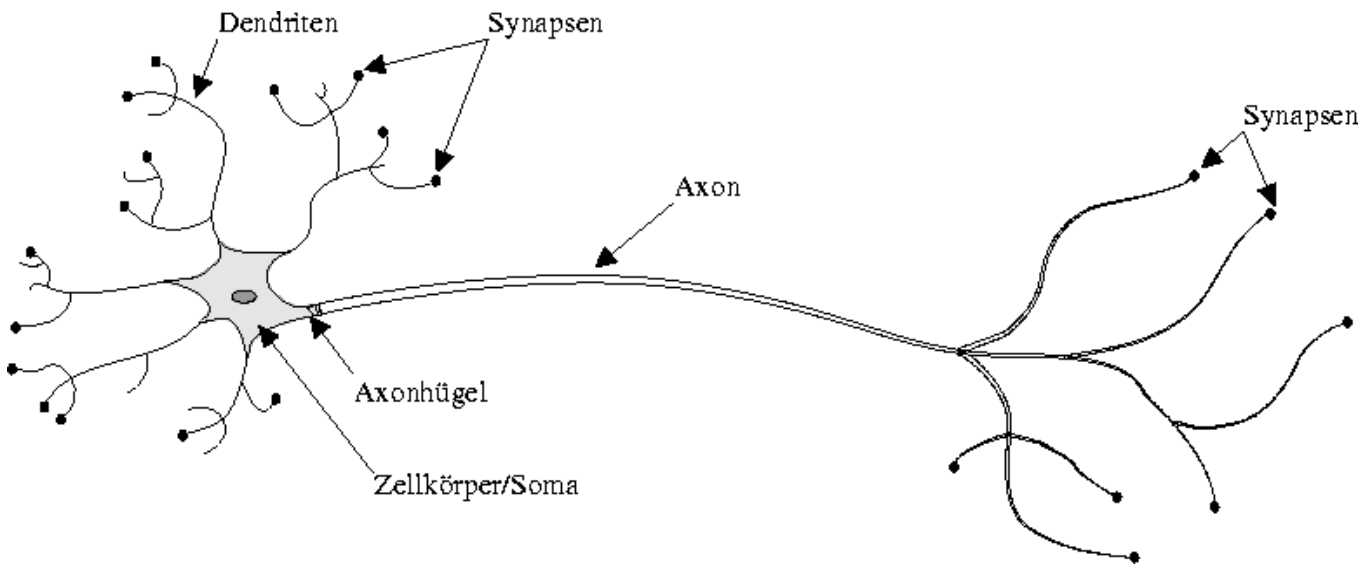
Überblick

Überblick

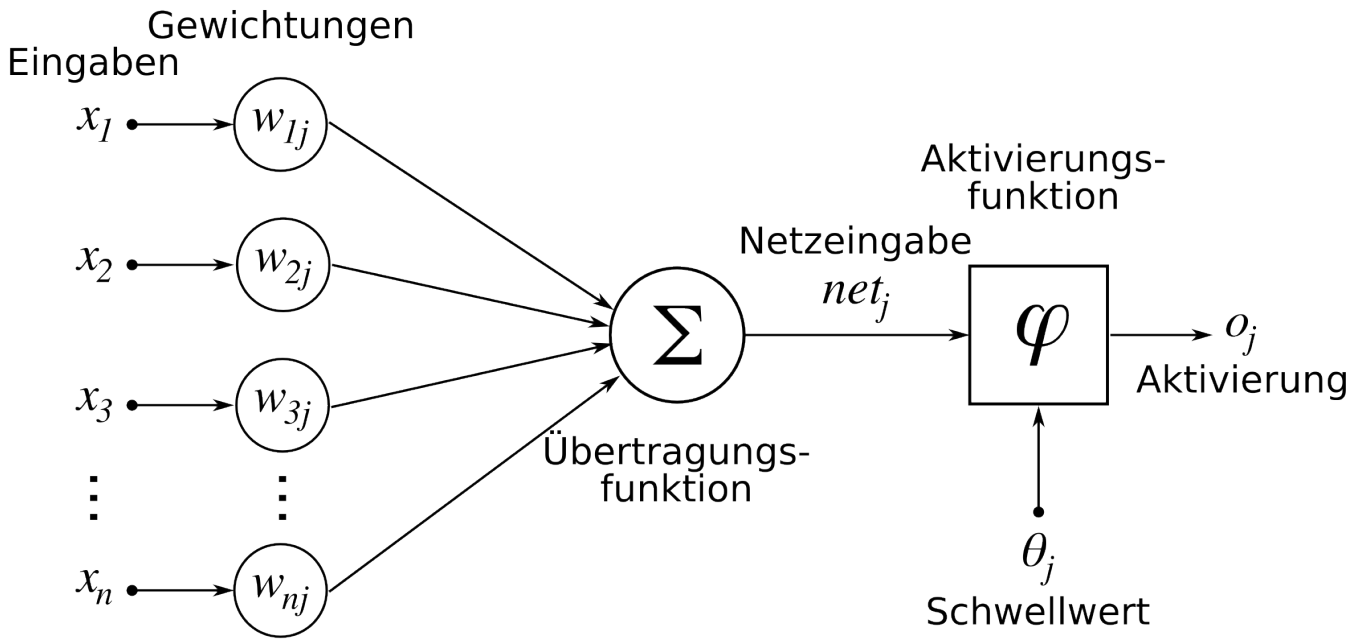
Motivation

- von Natur inspirierte Methode um Computer lernfähig zu machen
- ermöglicht auf Unbekanntes zu reagieren
- Anwendungen:
  - Mustererkennung
  - Frühwarnsysteme
  - Optimierung
  - Nachbildung von biologischen neuronalen Netzen

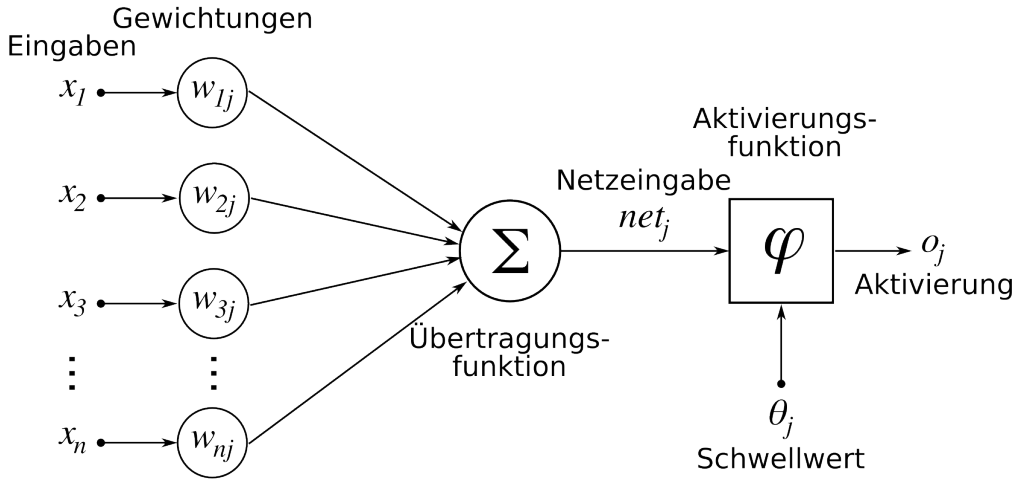
Nervenzelle



Modelliertes Neuron

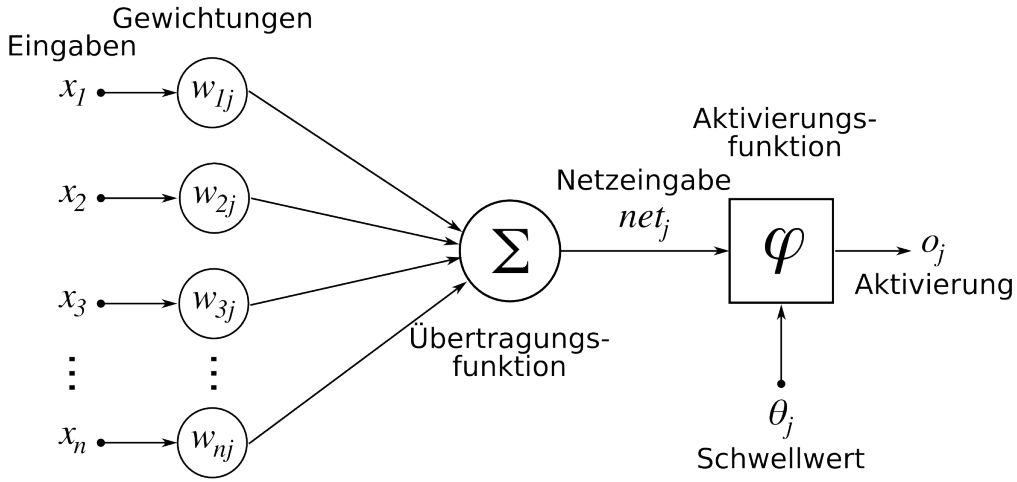


Eingangsfunktion



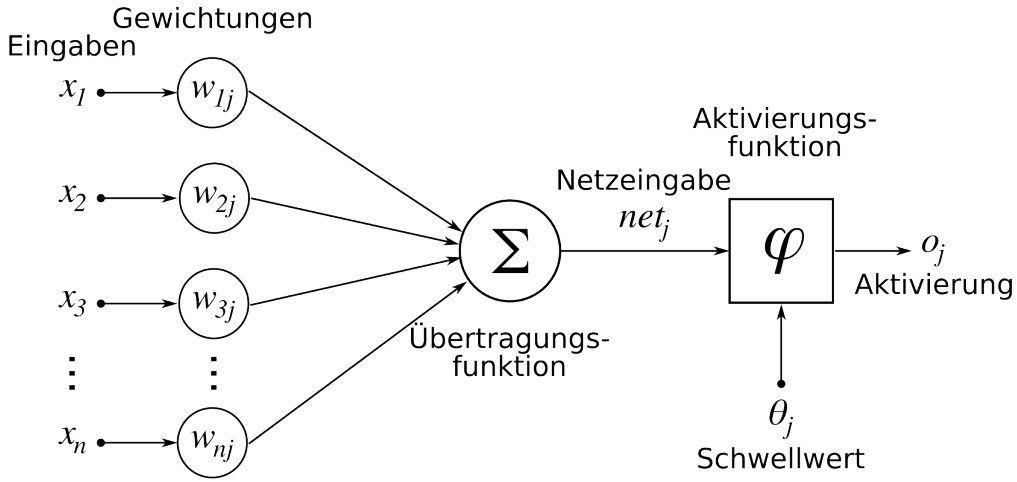
- Eingängssignale x_1 bis x_n liegen an
- Jeder Eingang hat ein bestimmtes Gewicht w_1 bis w_n
- Berechne effektiven Eingang mittels einer Eingangsfunktion ε
- ε hängt von den Eingängssignalen und Gewichten ab
- meistens $\varepsilon(\vec{x}, \vec{w}_j) := \langle \vec{x}, \vec{w} \rangle = \sum_{i=1}^{n} x_i w_i$

Aktivierungsfunktion



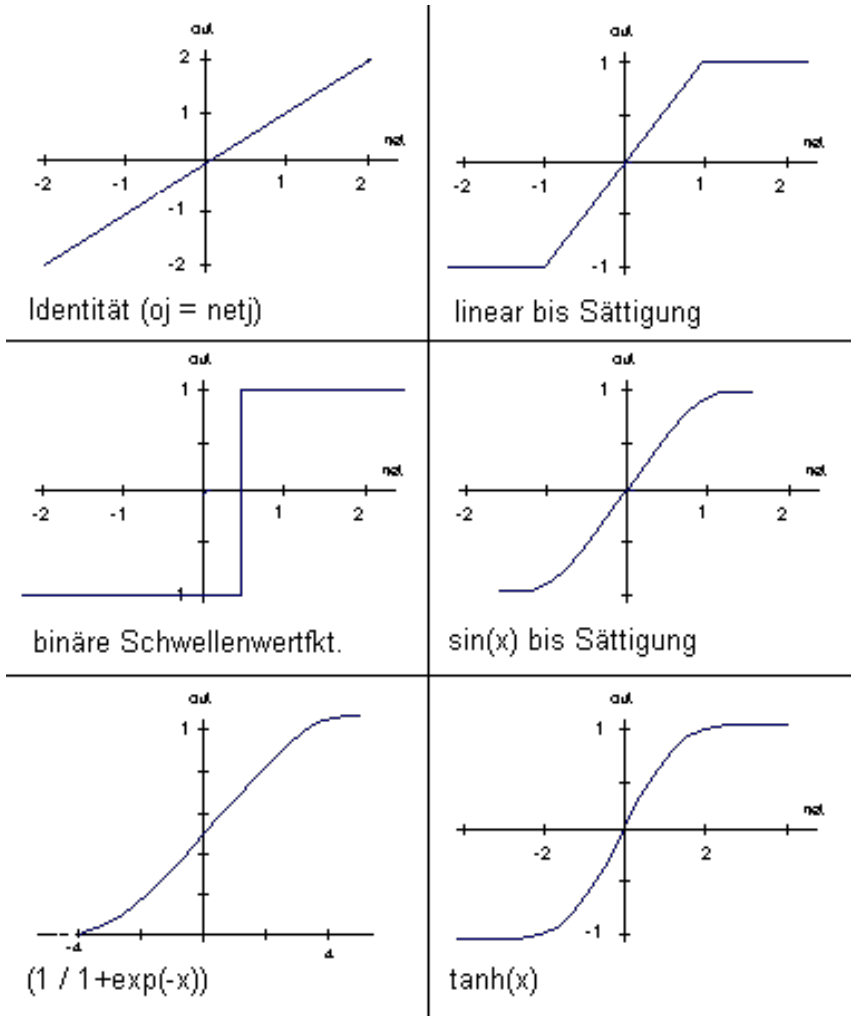
- c hängt vom effektiven Eingang ε ab
- meistens ist c die Identität

Ausgangsfunktion



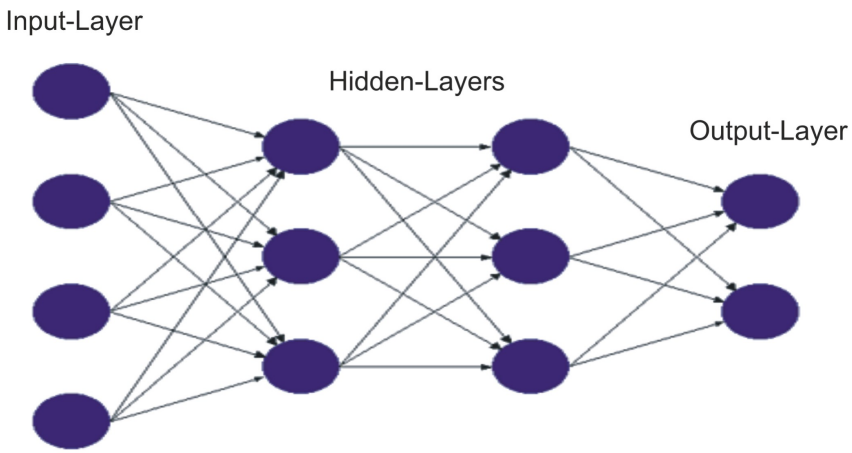
- a hängt von der Aktivität c ab
- muss monton wachsend sein
- Ausgangswertebereich wesentliches Kennzeichen für Neuronenmodell
- Natur: Schwellenwertfunktion

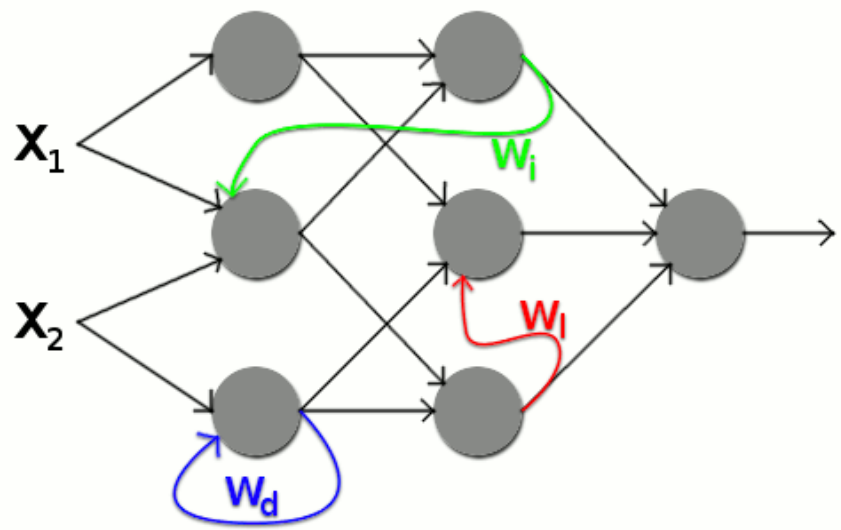
Ausgangsfunktion



Aufbau des Netzes

- Konvergenz- / Divergenzregeln zum Zusammenschluss der Neuronen
- es gibt Eingangsneuronen, versteckte Neuronen, Ausgangsneuronen
- Nummerierung der Neuronen
- Strukturierung in Schichten
- wesentliches Charakteristikum: Signalfluss
  - vorwärts (feedforward)
  - Rückkopplung (rekkurentes Netz)





Reproduktion

- es werden Werte an Netzeingängen angelegt und daraus die Netzausgänge berechnet
- bei computersimulierten Netzen ändern sich die Zustände in diskreten Zeitschritten
- bei **feedforward-Netzen** kann der Signalfluss ohne Probleme berechnet werden (von Eingangs- zur Ausgangsschicht vorarbeiten)
- bei **rekkurenten Netzen** muss folgendes Struktogramm befolgt werden:

- Eingänge (und Aktivitäten auf Startwerte setzen)
- Eingangswerte ans Netz anlegen (nicht an Neuronen weitergeben)
  - Neuronenausgänge berechnen (nicht weitergeben)
  - Neuroneneingänge mit neuen Werten belegen
- Wiederholen, bis Abbruchkriterium erfüllt

Lernen

- gewöhnlich werden nur die Gewichte geändert
- Stabilität- / Plastizitätdilemna
- es gibt 3 Lerntypen:

überwachtes Lernen

- dem Netz werden bekannte Eingabe- und Ausgabevektoren (Muster) vorgegeben
- das Netz kann dann unbekannten Eingabevektoren plausible Ausgabevektoren zuordnen
- z.B.: Backpropagation, Hebbsche Lernregel

Belohnung

- es gibt für den erzeugten Ausgangsvektor positive oder negative Rückmeldung
- das Netz soll Strategie (Gewichte) finden, sodass zu jeder Situation die Summe der Rückmeldung maximiert wird

unüberwachtes Lernen

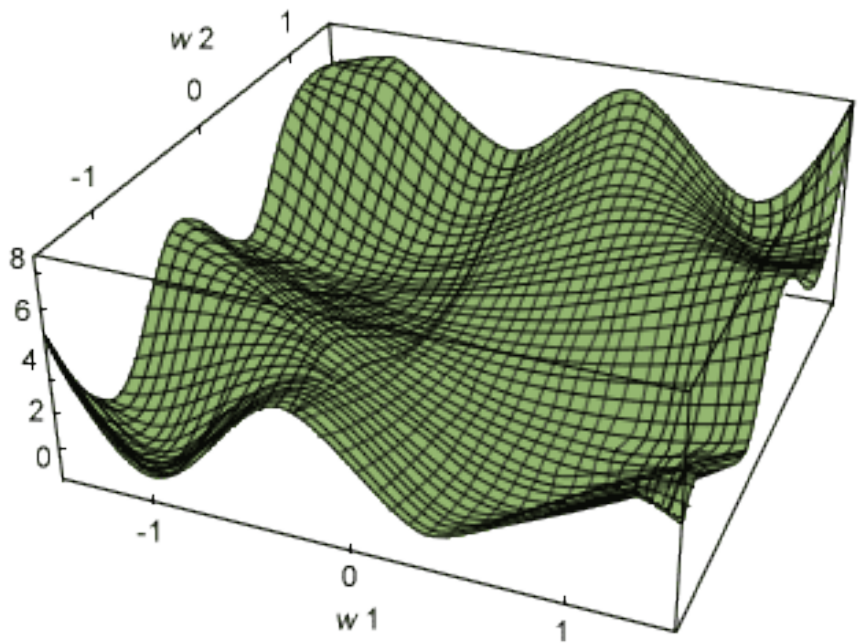
- dem Netz werden nur Eingabevektoren vorgegeben
- das Netz erkennt selbstständig Klassen
- der Natur am nächsten

Backpropagation

- häufig verwendetes Verfahren beim überwachten Lernen
- Minimierung des des Fehlers zwischen erwartetem und realem output durch Gradientenabstiegsverfahren
- Sei dazu n die Anzahl der Gewichte ($n \leq (\text{Anz. Neuronen})^2$)
- Fehlerfunktion $E : \mathbb{R}^n \to \mathbb{R}$ mit $E(\vec{w}) = E(w_1, \ldots, w_n)$
- als Fehlerfunktion wird oft der quadratische Abstand zwischen erwartetem und realem output verwendet (dabei wird über jedes bekannte Muster summiert)
- Sei N die Anzahl der Muster (bekanntes Paar: input $\vec{t}$ und erwarteter output $\vec{o}$)
- $s(\vec{t})$ ist der von Netz tatsächlich generierte output für einen input-Vektor $\vec{t}$
- Fehler des Muster p ist $E_p = \frac{1}{2} \sum_i (s(\vec{t_p})_i - o_{pi})^2$

Backpropagation

- wir haben gesehen Fehler des Muster p ist $E_p = \frac{1}{2}\sum_{i=1}(s(\vec{t_p})_i - o_{pi})^2$
- damit ergibt sich der Gesamtfehler über alle Muster: $E = \sum_{p=1}^{N} E_p$
- E ist somit nur von den Gewichten $\vec{w}$ abhängig, nun sollen Gewichte gefunden werden, sodass der Fehler minimal wird
- Gradientenabstiegstiegsverfahren findet (oft) lokale Minima



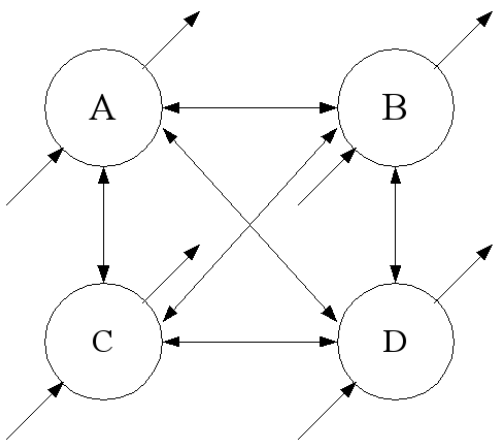
Backpropagation

- Gradient zeigt immer in Richtung des steilsten Anstiegs
- bestimme in jedem Schritt neue Gewichte Δw_i
- $\Delta \vec{w} = -\eta \nabla E(\vec{w})$
- $\Rightarrow \mathbf{\Delta w_i = -\eta \frac{\partial}{\partial w_i} E(\vec{w})}$
- wobei η die "Lernrate" ist (Skalierung des Gradienten)

- η darf nicht zu groß sein, sonst kann das Verfahren aus Senken springen
- darf aber auch nicht zu klein sein, sonst konvergiert das Verfahren zu langsam
- Backpropagation eignet sich in dieser Form nur für feedforward-Netze (kann für rekkurente Netze modifiziert werden)

Mustererkennung - Hopfieldnetz

- rekkurentes Netz
- es gibt nur eine Schicht, ist gleichzeitig Ein- und Ausgabeschicht
- jedes Neuron ist mit jedem, ausgenommen mit sich selbst, verbunden
- die Neuronen nehmen entweder den Wert -1 oder 1 (entspricht “feuert nicht“, “feuert“) an
- Gewichtsmatrix ist symmetrisch, d.h. $w_{ij} = w_{ji}$



Mustererkennung - Lernen

- jedes Pixel eines schwarzweiß Bildes entspricht genau einem Neuron q_i (1, wenn Pixel an und -1, wenn Pixel aus)
- Lernen über Hebbsche Lernregel (wenn zwei Neuronen gleichzeitig aktiv sind, wird ihre Verbindung verstärkt)
- es werden N Muster angelegt, wobei q_{pi} der Zustand des Neurons (Pixels) i vom Muster p ist
- die Muster werden gelernt, d.h. die Gewichte berechnet:
- $$\mathbf{w_{ij}} = \sum_{\mathbf{p=1}}^{\mathbf{N}} \mathbf{q_{pi}} \mathbf{q_{pj}}$$

Mustererkennung - Eingabe eines Testmusters

- nun wird ein Testmuster mit Werten $\tilde{q}_i$ angelegt.
- die Zustände der Neuronen werden asynchron aktualisiert (zufällig ausgewählt)
- $\tilde{q}_i = \begin{cases} 1, & \text{wenn } \sum_j \tilde{q}_j w_{ij} \geq 0 \\ -1, & \text{sonst} \end{cases}$
- bei wenig verrauschten Eingabemustern ist das Verfahren ziemlich gut
- Verfahren konvergiert manchmal auch gegen “Artefakte“

Quellen

- Hoffmann, “Kleines Handbuch Neuronale Netze“, 1993 Vieweg
- Zell, “Simulation neuronale Netze“, 1997 R. Oldenbourg Verlag
- http://de.wikipedia.org/wiki/Hopfield-Netz
- Bilder:
  - http://pi.informatik.uni-siegen.de/Arbeitsgebiete/ci/images/neuronen.jpg
  - http://pille.iwr.uni-heidelberg.de/~ocr01/images2/nnet.png
  - http://www.chemgapedia.de/vsengine/media/vsc/de/ch/13/anc/daten/neuronalenetze/images/snn_117.gif
  - http://cs.uni-muenster.de/Studieren/Scripten/Lippe/wwwnnscript/bilder/aktivfkt.gif
  - http://commons.wikimedia.org/wiki/File:ArtificialNeuronModel_english.png
  - http://programmingwiki.de/images/thumb/f/fa/Hopfield_net.png/200px-Hopfield_net.png
  - http://upload.wikimedia.org/wikipedia/de/4/4c/Neuronal-Networks-Feedback.png

Vielen Dank für eure Aufmerksamkeit!