# Das Perzeptron-Konvergenztheorem

Daniel Seifert

19.12.2001

---

## 1 Einführung

Computer sind in vielen Punkten nicht so leistungsfähig wie das menschliche Gehirn. Wohl können Computer um einen bedeutenden Faktor schneller rechnen als Menschen, aber für uns so einfache Tätigkeiten wie Sprache erkennen oder Assoziationen bilden sind für die heutigen Computer nur sehr stark eingeschränkt möglich.

Bereits bei der Entwicklung der Flugzeuge hat sich der Mensch an der Natur orientiert und versucht, durch einen Nachbau der Vogelschwingen einen Flugapparat zu basteln. Auch in Fragen der „Künstlichen Intelligenz" hat sich eine Forschungsrichtung entwickelt, welche sich damit beschäftigt, das menschliche Gehirn nachzubilden. Auf diese Weise erhofft man sich, nicht nur den Geheimnissen unseres Denkens näher zu kommen, sondern auch Maschinen mit menschlichen geistigen Fähigkeiten ausstatten zu können.

### 1.1 Neuronen

Das menschliche Gehirn besteht aus unzähligen Neuronen, welche auf vielfältige Weise miteinander verknüpft sind und sich beeinflussen. Dieses Wechselspiel, wo sich Neuronen gegenseitig anfeuern oder blockieren, ist ein in höchstem Maße paralleler Prozeß mit in gewissermaßen redundanter Speicherung der Daten.

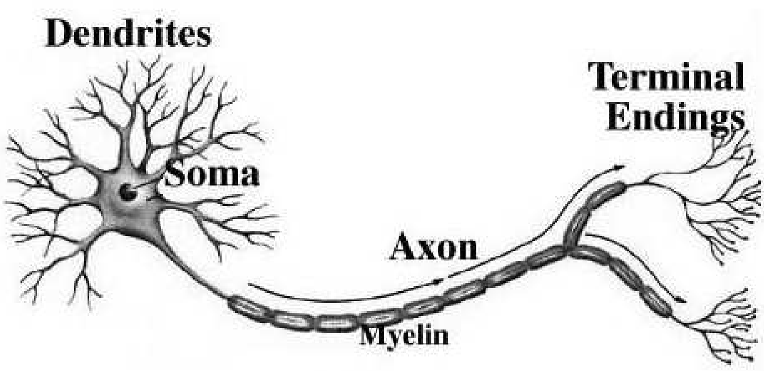


Abbildung 1: Ein Neuron[5]

Ein Neuron besteht dabei (u.a.) aus Dendriten, Soma, Axon und Endkolben. Dabei nimmt das Neuron „Werte" über die Dendriten auf und „entscheidet" im Soma ob es *feuern* soll, ob es also den Impuls über das Axon weiterleitet und über die Endkolben an andere Neuronen weiterreicht.

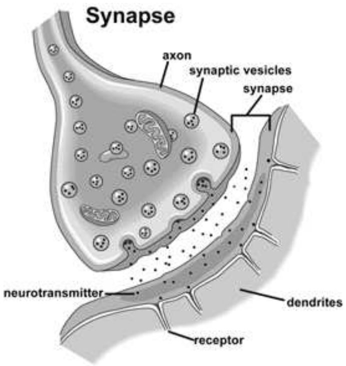


Abbildung 2: Eine Synapse[6]

Dabei kann ein bei einer Synapse ankommender Impuls das angesprochene Neuron nicht nur darin bestärken, ebenfalls einen Impuls abzufeuern (exzitatorische Synapse), sondern es kann auch sein, daß dies gehemmt wird (inhibitorische Synapse).

## 1.2 Das Gehirn

Der Begriff *Neuronales Netz* ist ein Schlagwort in der Künstlichen Intelligenzforschung. Auch, bzw. insbesondere, das Gehirn ist ein sogenanntes (biologisches) neuronales Netz: die gesamten Neuronen, deren Verbindungen und Eigenschaften. Erst in der Gesamtheit entwickeln die Neuronen die Leistung, welche die Fähigkeiten des Menschen ausmachen.

## 1.3 Die Simulation

Bereits 1943 stellten McCulloch[1] und Pitts[2] erste Ergebnisse ihrer Forschungen über Neuronen und Künstliche Neuronale Netze vor. Sie beschrieben das *McCulloch/Pitts-Neuron* (*MCP-Neuron*) und wie man MCP-Neuronen verknüpfen kann und somit jeden durch einen Computer ausführbaren Algorithmus auch durch ein Neuronales Netz aus MCP-Neuronen durchführen kann[3].

Ein MCP-Neuron ist dabei folgendermaßen aufgebaut:

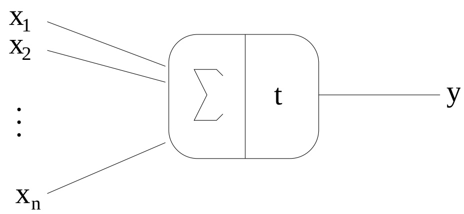


Abbildung 3: Ein McCulloch/Pitts-Neuron

Die Eingabewerte und der Ausgabewert sind hierbei binär. Die Eingabewerte $x_i$ werden aufsummiert und ein interner Schwellenwert (engl. *treshold*) gibt an, ob eine 1 oder eine 0 als Ausgabewert erzeugt werden soll. Gibt ein Neuron eine 1 aus, so sagen wir, daß es *feuert*.

[1] Warren McCulloch, 1898-1968 (http://cognet.mit.edu/MITECS/Entry/lettvin2)

[2] Walter Pitts, 1923 bis ca 1969 (http://cognet.mit.edu/MITECS/Entry/lettvin1)

[3] In [2] findet sich für den interessierten Leser eine ausführliche Darstellung eines MCP-Neuron inklusive einiger graphischer Demonstrationen, wie ein solches Neuron funktioniert.

Hier sehen wir bereits, daß die künstlichen Neuronen recht einfach aufgebaut sind. W. McCulloch und W. Pitts haben angenommen, daß ein echtes Neuron ebenso arbeitet. Es gibt aber auch andere Ansätze, ein Neuron zu modellieren.

## 2 Das Perzeptron

Frank Rosenblatt, ein amerikanischer Psychologe, beschäftigte sich um 1958 mit der Frage, ob es möglich ist, daß sich ein neuronales Netz selbständig an ein bestimmtes Problem anpassen kann. Ergebnis seiner Überlegungen war das sogenannte *Perzeptron*. Dabei handelt es sich um ein neuronales Netz, welches nicht nur seine Eingänge aufsummiert und anhand des Schwellenwertes entscheidet, ob es feuert, sondern die Eingänge auch noch gewichtet.

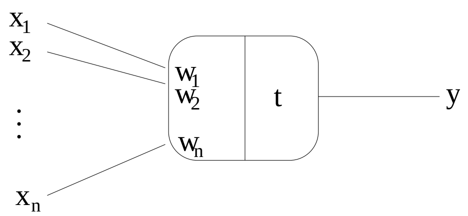


Abbildung 4: Ein Perzeptron (Grundmodell)

Im folgenden betrachten wir nur normierte Perzeptrons die wir erhalten, indem wir einen zusätzlichen Eingang mit konstant 1 anlegen und als Gewicht unseren Schwellenwert benutzen.

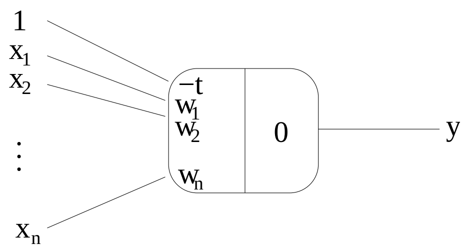


Abbildung 5: Perzeptron mit Bias

Der eigentliche Schwellenwert kann damit auf 0 gesetzt werden und wir können die Ausgabe $y$ des Perzeptrons folgendermaßen berechnen:

$$y = \begin{cases} 1, & \text{falls } wx \geq 0 \\ 0, & \text{sonst} \end{cases}$$

Dabei gelten folgende Bezeichnungen:

- $n$ ist die Zahl der Eingänge des Perzeptrons
- $w_1$ bis $w_n \in \mathbb{R}$ sind die Gewichte der jeweiligen Eingänge $x_i$. $w_0$ ist das Gewicht für den konstanten 1-Eingang.
- $w = (w_0, w_1, \ldots, w_n) \in \mathbb{R}^{n+1}$ ist der sogenannte Gewichtsvektor, $x = (1, x_1, \ldots, x_n)$ ist der Eingangsvektor mit den Eingangssignalen $x_1, \ldots, x_n$.
- $wx := \sum_{i=0}^{n} w_i x_i$ (Skalarprodukt)

## 2.1 Beispiele

### 2.1.1 AND / OR

Mit einem einzelnen Perzeptron können wir bereits einfache Boolsche Operationen berechnen, zum Beispiel die AND- und die OR-Funktion.

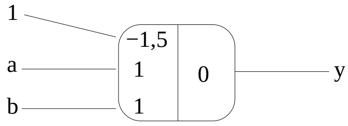


Abbildung 6: Perzeptron für die UND-Funktion

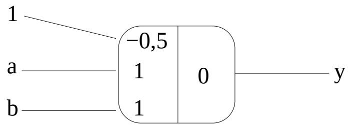


Abbildung 7: Perzeptron für die ODER-Funktion

### 2.1.2 XOR

Interessanterweise versagt das Perzeptron komplett bei einer weiteren zweistelligen boolschen Operation, dem XOR. Es ist nicht möglich, einen Gewichtsvektor zu finden, mit dem ein Perzeptron mit obigen Eigenschaften diese Funktion modellieren kann.

Warum aber geht das nicht? Nun, offensichtlich ist die Perzeptron-Funktion zur Berechnung der Ausgabe eine lineare Funktion. Dies bedeutet - anschaulich gesprochen - daß die $n-1$-dimensionale Hyperebene $\sum x_i w_i = 0$ unsere $n$-dimensionale Eingabemenge $X$ in zwei Teile zerlegt. Alle Elemente „auf der einen Seite" sind dann Werte bei denen das Neuron feuert, die Elemente „auf der anderen Seite" werden dagegen keine Reaktion auslösen. Existiert für eine Menge $X$ eine Hyperebene mit dieser Eigenschaft - das also die Menge $X$ auf diese Weise in zwei wohlunterschiedene Teilmengen zerlegt werden kann - so sagen wir, daß $X$ linear-separabel ist.

Betrachten wir die XOR-Funktion, so ist sofort offensichtlich, daß keine solche Hyperebene (in diesem Fall eine Gerade) existiert.

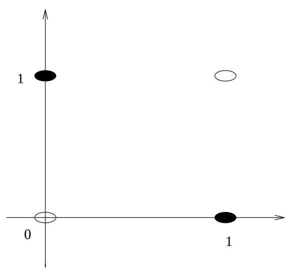


Abbildung 8: XOR

## 2.2 Einschränkungen ...

Marvin Minsky und Seymour Papert haben 1969 in „*Perceptrons*" mathematisch bewiesen, daß nur linear separierbare Funktionen durch ein solches Perzeptron wie oben definiert dargestellt werden können. Dies ist noch kein Problem, da durch Hinzufügen weiterer Schichten auch nicht-linear-separierbare Funktionen realisiert werden können - allerdings haben Minsky und Papert auch gezeigt, daß in diesem Fall kein effizienter Lernalgorithmus existiert.

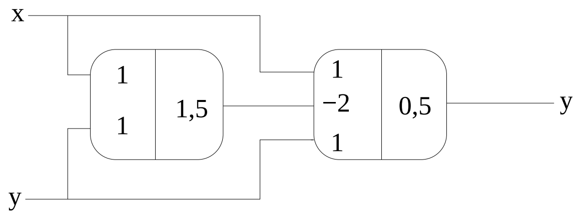


Abbildung 9: XOR mit einem two-layer Perzeptron

Als Ergebnis ihres Buches erlag in den folgenden Jahren die Forschung im Bereich der KI fast vollständig.

# 3 Lernen

## 3.1 Lernalgorithmus

Das Ergebnis von Rosenblatts Untersuchungen an Single-Layer Perzeptrons war das im nachfolgenden Abschnitt vorgestellte Perzeptron-Konvergenztheorem, welches die Existenz eines Algorithmus postulierte, mit dem ein Perzeptron in endlicher Zeit *lernen* kann, korrekte Ausgaben zu erzeugen.

Ein solcher Algorithmus, mit dem dies unter Zuhilfenahme eines externen „Bewerters" erreicht werden kann, soll im folgenden vorgestellt werden. Das dieser wirklich unsere Behauptung erfüllt, wird im Rahmen des Beweises des Perzeptron-Konvergenztheorems gezeigt.

Wie im vorigen Abschnitt besprochen, müssen wir unsere Voraussetzungen dahingehend festlegen, daß unsere Eingabemenge

$$X = \{(1, x_1, \ldots, x_n) \mid x_i \in \{0, 1\}\}$$

linear separierbar in zwei Mengen $X_0$ und $X_1$ bzgl. des Ausgabewertes ist. Unser *idealer* Gewichtsvektor $w$ hat nun die Eigenschaft, daß gilt:

$$\begin{aligned} x \in X_0 \quad &\Rightarrow \quad wx < 0 \\ x \in X_1 \quad &\Rightarrow \quad wx \geq 0 \end{aligned}$$

1. Am Anfang unseres Algorithmus starten wir mit einem beliebigen Gewichtsvektor $w_0$. Dieser wird, aller Voraussicht nach, nicht unser gesuchter Gewichtsvektor sein. Wir konstruieren nun eine unendliche Folge von Elementen aus $X$ mit der Eigenschaft, daß jedes Element aus $X$ unendlich oft in dieser sogenannten *Trainingsfolge* vorkommt.

2. Wir leiten nun nacheinander die Elemente aus der Trainingsfolge in das Perzeptron. Wir betrachten nach jeder Eingabe die Ausgabe und bilden einen neuen Gewichtsvektor $w_{i+1}$ nach folgender Regel:

$$w_{i+1} = \begin{cases} w_i + x & \text{falls } y \in X_1 \text{ und } wx < 0 \\ w_i - x & \text{falls } y \in X_0 \text{ und } wx \geq 0 \\ w_i & \text{sonst} \end{cases}$$

3. Wie wir im Abschnitt 3.2 beweisen werden, wird die Folge der $w_i$ gegen unseren gesuchten Gewichtsvektor $w$ konvergieren, und zwar in endlicher Zeit.

Die Idee bei diesem Algorithmus ist, daß der aktuelle Gewichtsvektor bei falsch klassifizierten Eingabevektoren so angepaßt wird, daß die typischen „Merkmale" dieses Eingabevektors in unserem Gewichtsvektor betont werden. Beispielsweise ordne der Gewichtsvektor $w_i$ dem Eingabevektor $x \in X_0$ fälschlicherweise den Ausgabewert 1 zu, also $wx \geq 0$. Dann wäre $w_{i+1} = w_i - x$ und es würde gelten:

$$w_{i+1}x = (w_i - x)\,x = w_i x - xx < w_i x$$

Es muß allerdings noch nicht gelten, daß $w_{i+1}x < 0$, aber wir sind diesem Ziel bereits ein Stückchen näher gekommen. Genaugenommen kann es sogar passieren, daß nach dieser Adjustierung ein anderer, von $w_i$ korrekt eingeordneter Eingabevektor, durch $w_{i+1}$ falsch eingeordnet wird. Aber wie bereits angedeutet, werden wir im folgenden Abschnitt sehen, daß nach endlich vielen solcher Schritte unsere Folge von Gewichtsvektoren gegen einen korrekten Gewichtsvektor $w$ konvergieren wird.

Unter [4] existiert ein Java-Applet, welches den Lernprozeß anhand eines Perzeptrons mit zwei Eingängen verdeutlicht, dort wird auch nochmal gezeigt wie das XOR-Problem durch Einfügen einer Zwischenschicht gelöst werden kann.

## 3.2 Perzeptron-Konvergenztheorem

> **Satz** (Perzeptron-Konvergenztheorem): Gegeben seien zwei linear separierbare Mengen von Punkten $X_0$ und $X_1$. Dann existiert ein Lern-Algorithmus, durch den ein Perzeptron in endlicher Zeit einen Gewichtsvektor erhält, so daß das Perzeptron allen Punkten in $X_0$ den Wert 0 und allen Punkten in $X_1$ den Wert 1 zuweist.

Ein Algorithmus ist zum Beispiel der im Abschnitt 3.1 vorgestellte. Wir wollen nun zeigen, daß dieser wirklich das Konvergenztheorem erfüllt, daß also nach einer endlichen Zeit ein Gewichtsvektor $w$ gefunden wird, der für alle Eingabevektoren die korrekte Ausgabe erzeugt.

Für den Beweis werden wir eine kleine Anpassung an der Menge $X_0$ vornehmen, also an der Menge der Punkte aus $X$, für welche das Perzeptron den Wert 0 ausgeben soll. Und zwar ersetzen wir $X_0$ durch $X_0^{,}$ mit der Eigenschaft

$$X_0^{,} = \{-x \mid x \in X_0\}$$

um die folgenden Notationen zu vereinfachen. o.B.d.A. gelte desweiteren, daß wir einen Lösungsvektor mit der Eigenschaft $wx \neq 0$ für alle Eingabevektoren suchen. Ein Eingabevektor aus $X_0^{,} \cup X_1$ wird damit genau dann korrekt erkannt, wenn $wx > 0$.

Wir bilden desweiteren eine Teilfolge $(w_{i_k})$ der Folge der $w_i$'s mit der Eigenschaft, daß für alle $k \in I \subseteq \mathbb{N}$ gilt: $w_{i_k} \neq w_{i_{k+1}}$. Das heißt, wir betrachten in unserer Folge der Gewichtsvektoren nur die Stellen, wo sich (aufgrund einer falschen Klassifizierung) der Gewichtsvektor geändert hat. Außerdem legen wir fest, daß wir mit $w_0 = (0, \ldots, 0)$ beginnen. Damit gilt:

$$w_{i_{k+1}} = \sum_{l=1}^{k} x_{i_l}$$

Um zu beweisen das unsere oben beschriebene Trainingsfolge konvergiert, genügt es zu zeigen, daß $I$ endlich ist. Damit ist dann auch die Teilfolge $w_{i_k}$ (also die Zahl der durchgeführten Änderungen des Gewichtsvektors) endlich.

Sei $\tilde{w}$ ein beliebiger Lösungsvektor, für den gilt, daß für alle $x \in X_0^{,} \cup X_1$: $\tilde{w}x > 0$. Dann existiert ein $\alpha \in \mathbb{R}^+$ mit $\tilde{w}x \geq \alpha > 0$ und es gilt:

$$\begin{aligned} w_{i_{k+1}}\tilde{w} &= (x_{i_1} + x_{i_2} + \ldots + x_{i_k})\,\tilde{w} \\ &= x_{i_1}\tilde{w} + \ldots x_{i_k}\tilde{w} \\ &\geq \alpha + \cdots + \alpha \\ &= k\alpha \end{aligned}$$

Wir rufen uns an dieser Stelle die Cauchy-Schwarzsche-Ungleichung in Erinnerung:

> Sei $V$ ein euklidischer Vektorraum mit der Norm $||x||_V := \sqrt{< x, x >}$. Für $x, y \in V$ gilt:
> $$| < x, y > | \leq ||x|| \cdot ||y|| \tag{1}$$

In unserem Fall handelt es sich bei $V$ um den Vektorraum $\mathbb{R}^n$ und um das Skalarprodukt

$$xy := < x, y > := \sum_{i=1}^{n} a_i b_i$$

Speziell gilt also:

$$(xy)^2 \leq xx \cdot yy$$

Wir können also folgende Abschätzung vornehmen:

$$k^2\alpha^2 \leq \left(w_{i_{k+1}}\tilde{w}\right)^2 \leq \left(w_{i_{k+1}}w_{i_{k+1}}\right) \cdot (\tilde{w}\tilde{w})$$

also

$$\frac{k^2\alpha^2}{(\tilde{w}\tilde{w})} \leq w_{i_{k+1}}w_{i_{k+1}} \tag{2}$$

Das heißt, daß $w_{i_{k+1}}w_{i_{k+1}}$ quadratisch in k wächst, also

$$w_{i_{k+1}}w_{i_{k+1}} > w_{i_k}w_{i_k} \tag{3}$$

Dieses Wachstum ist aber nach oben beschränkt:

$$\begin{aligned}
w_{i_{k+1}}w_{i_{k+1}} &= (w_{i_k} + x_k)(w_{i_k} + x_k) && (4)\\
&= w_{i_k}w_{i_k} + 2w_{i_k}x_k + x_kx_k && (5)\\
&\leq w_{i_k}w_{i_k} + x_kx_k && (6)\\
&\leq \ldots && (7)\\
&\leq \sum_{l=1}^{k} x_lx_l && (8)\\
&\leq k \cdot M && (9)
\end{aligned}$$

mit $M := sup\{xx \,|\, x \in X\}$. (6) folgt hierbei übrigens aus der Tatsache, daß $w_{i_k}x_k < 0$. Fügen wir nun unsere Erkenntnisse über untere (2) und obere (9) Schranken zusammen, so erhalten wir:

$$\frac{k^2\alpha^2}{(\tilde{w}\tilde{w})} \leq w_{i_{k+1}}w_{i_{k+1}} \leq k \cdot M$$

also

$$k \leq M \cdot \frac{\tilde{w}\tilde{w}}{\alpha^2}$$

Wir sehen also, daß $k$ durch eine Konstante $\beta = M \cdot \frac{\tilde{w}\tilde{w}}{\alpha^2}$ beschränkt ist, welche nur abhängig ist von der Menge $X$ und unserem frei gewählten Lösungsvektor $\tilde{w}$. $k \in I$ ist also nach oben beschränkt, die Folge $(w_{i_k})_{k \in I}$ ist also endlich und unser Gewichtsvektor ändert sich nach maximal $M \cdot \frac{\tilde{w}\tilde{w}}{\alpha^2}$ Änderungen nicht mehr.

q.e.d.

# Literatur


[1] Schöning, Uwe: „Perlen der Theoretischen Informatik - 9 weitere Themen", 1995

[2] „A computer model of the neuron - The McCulloch & Pitts neuron" (http://www.mind.ilstu.edu/curriculum/perception/mpneuron1.html)

[3] Hellmich, Rainer: „Einführung in intelligente Softwaretechniken", 1997

[4] F. Corbett: „Perceptron Learning Applet" (http://home.cc.umanitoba.ca/~umcorbe9/perceptron.html)

[5] Bild 1: http://www2.gasou.edu/psychology/courses/muchisky/intro/neuron.jpg

[6] Bild 2: http://www.neurogen.com/graphx/Synapse.gif