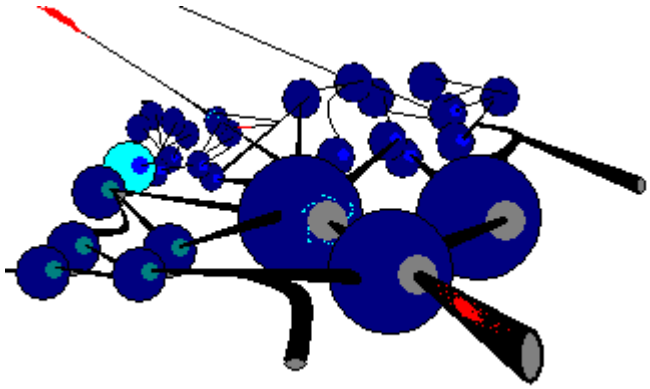
# Lernen mit neuronalen Netzen

## Am Beispiel der Backpropagation in Anwendung auf den schiefen Wurf (Kanonenschuss)

Manuskriptentwurf für
ein Schülerseminar

Gunter Heim



www.philbot.de
Aachen, 30. März 2007

## Einführung: QBasic simuliert Kanonenkugel

Mit Hilfe von Computerprogrammen lassen sich bestimmte Eigenschaften von Neuronen (Nervenzellen) simulieren. Jedes Neuron erhält Signale von anderen Neuronen, führt mit diesen eine interne Berechnung durch und sendet als Ergebnis selbst Signale an andere Nervenzellen. Indem die eingehenden Signale eines Neurons mit einem variablen Faktor gewichtet werden lassen sich neuronale Netze darauf trainieren, bestimmte Aufgaben zu verrichten. Speist man etwa den Abschusswinkel einer Kanone sowie die Mündungsgeschwindigkeit der Kanonenkugel in ein neuronales Netz ein, so kann es daraus eine Ausgabe berechnen, die man als Flugweite interpretieren kann. Mit Hilfe der Information über die Treffgenauigkeit kann nun das Netz über mehrere Lernschritte (mehr als einhundert) die Vorhersagegenauigkeit erhöhen, indem es die Wichtungsfaktoren an den einzelnen Neuronen geschickt verändert: Das Netz lernt.

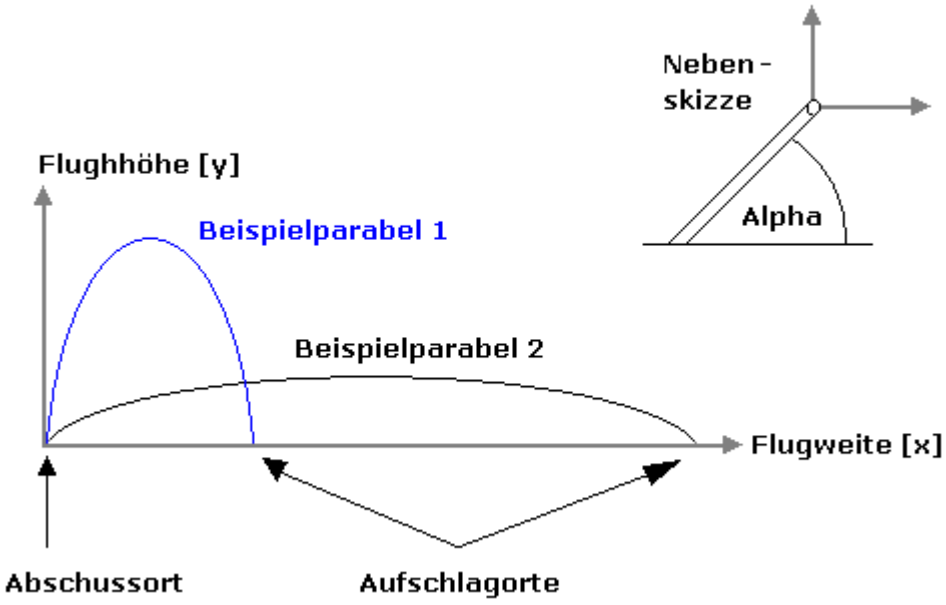
Die Idee biologische Nervenzellen über mathematische Modelle zu simulieren reicht bis in die 1940er Jahre zurück. Intensiv untersucht wurden sie in den 1960er und dann wieder ab den 1980er Jahren. Heute werden sie unter anderem zu kommerziellen Zwecken eingesetzt, etwa in der Gesichtserkennung, der Überprüfung von Handschriften und der Abschätzung der Kreditwürdigkeit von Bankkunden.

Im Folgenden wird ein als QBasic (Version 4.5) Programm realisiertes neuronales Netz dokumentiert. Das Netz mit insgesamt 8 Neuronen lernt über ein als Backpropagation bekanntes Verfahren.

Das Programm kann als Basic (bas) oder als exe-Datei aus dem Internet heruntergeladen werden über www.philbot.de/download/2007_neuronal.

## Der schiefe Wurf als Anwendungsbeispiel

Im Ursprung des abgebildeten Koordinatensystems sei eine Kanone platziert. Die Mündungsöffnung des Kanonenrohrs befinde sich genau auf Höhe der x-Achse, habe also die y-Koordinate Null.




Die Nebenskizze verdeutlch den Winkel α (alpha) der Kanone gegen die Horizontale an. Schiesst die Kanone steil ergibt sich eine parabelförmige Wurfbahn wie im Beispiel 1 oben dargestellt. Wird etwas flacher Geschossen, ergibt sich eine Wurfparabel wie im Beispiel 2. Variiert werden kann auch die Abschussgeschwindigkeit $v_0$ der Kanonenkugel. Alle hier betrachteten Rechnungen vernachlässigen den Einfluss des Luftwiderstands.

Unter "Schiefer Wurf" oder "Wurfparabel" findet man die nötigen Formeln in Physikbüchern für die Oberstufe. Hier wurden folgende Zusammenhänge zugrunde gelegt:

Flugweite x nach einer Zeit t: $x(t) = v_0 \cdot t \cdot \cos(\alpha)$
Flughöhe y nach einer Zeit t: $y(t) = v_0 \cdot t \cdot \sin(\alpha) - ½ \cdot g \cdot t^2$

"g" sei die Erdbeschleunigung. Sie wird im Programm mit 10 m/s² angesetzt. Die Flugweite, kurz "weite" bis zum Aufschlagort als Funktion der Abschusswinkels α und der Abschussgeschwindigkeit $v_0$ ist gegeben über:

Flugweite bis zum Aufschlag:

$$weite(v_o, \alpha) = \frac{v_o^2 \cdot \sin(2 \cdot \alpha)}{g}$$

Innerhalb des Basic-Programmes wurden die Größen zur Berechnung des schiefen Wurfes wie folgt benannt:

| | | |
|---|---|---|
| Abschusswinkel α: | alpha | (im Bogenmaß) |
| Abschussgeschwindigkeit $v_0$ : | veloc | (von velocity) |
| Flugweite bis zum Aufschlag: | weite | |

Intern berechnet QBasic die Winkelfunktionen Sinus und Cosinus mit Hilfe des Bogenmaßes. Dem vollen Winkel von 360° entspricht der volle Kreisumfang von 2π.

Folgende Formel wurde im Programm verwendet, um zwischen Bogen- und Winkelmaß umzurechnen:

$$Winkel\_im\_Bogenmaß = \frac{(Winkel\_im\_Gradmaß)\cdot 2\cdot \pi}{360}$$

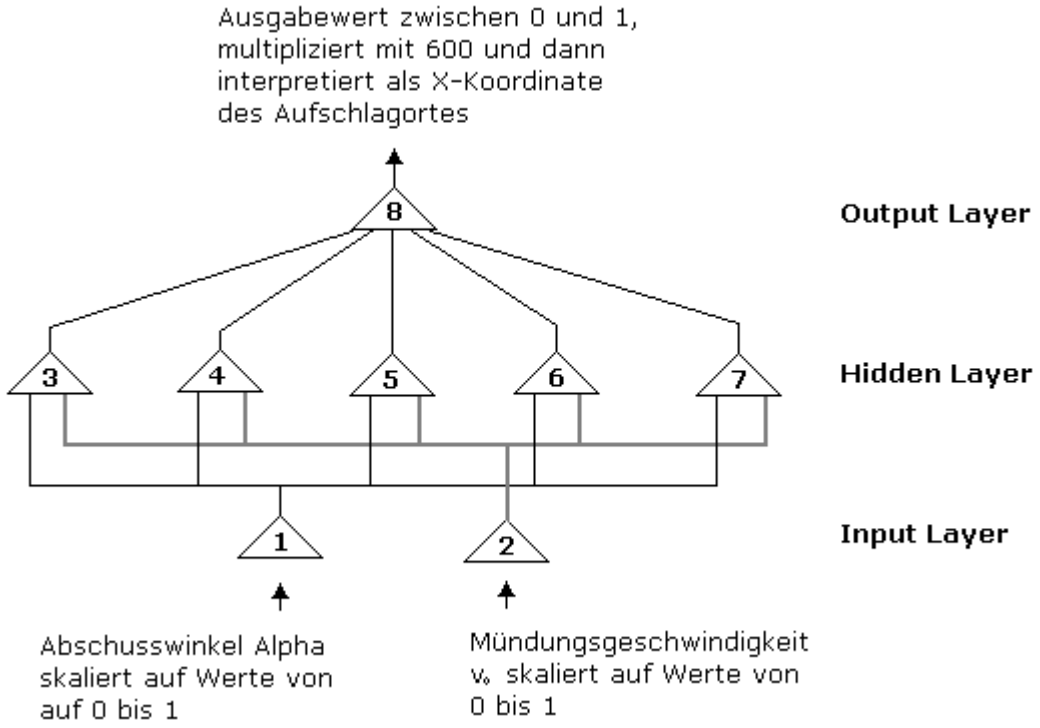
$$Winkel\_im\_Gradmaß = \frac{(Winkel\_im\_Bogenmaß)\cdot 360}{2\pi}$$

Zur grafischen Ausgabe auf dem Monitor wurde ein stilisiertes Koordinatensystem mit einer Länge der x-Achse von 0 bis 600 Pixel angenommen. Ein Pixel auf der x-Achse soll einem Meter in der Realität entsprechen. Da sich für eine Kanonenkugel die maximale Flugweite für einen Winkel α = 45° ergibt, kann man berechnen, welche maximale Abschussgeschwindigkeit $v_0$ nicht überschritten werden darf, um nicht weiter als 600m zu schießen. Dies sind 77 m/s.

Im QBasic-Programm wird der Abschusswinkel über einen Zufallsgenerator also zwischen 0° und 90° variiert und die Abschussgeschwindigkeit zwischen 0 und 77 m/s. Damit kann die Skalierung der Bildschirmkoordinaten ohne Umrechnungsfaktor direkt als Ausgabe genutzt werden. (Über den Befehl Screen 9 wird der Bildschirmmodus zur Programmlaufzeit auf eine Pixelhöhe von 350 und eine Breite von 640 gesetzt.)

## Zur Topologie des neuronalen Netzes

Eine weitverbreitete Form neuronaler Netze sind die sogenanten Feed-Forward-Netze mit einer Backpropagation als Lernalgorithmus. Ein solches Netz wurde hier in QBasic mit folgender Topologie verwendet:




Jedes Dreieck steht für ein Neuron. Die Ausgabe - der output - eines Neurons fließt durch die Spitze des Dreiecks in Richtung des nächsten Neurons nach oben. Striche an der Basis der Dreiecke stehen für Eingänge, den Input. Jedes Neuron schickt Signale an jedes Neuron der nächsthöheren Ebene. Es gibt auch Netze, in den Neuronen einer Ebene Signale austauschen oder Signale rückwärts zu tieferen Ebenen zurückgeführt werden. Solche Topologien werden hier aber nicht betrachtet.

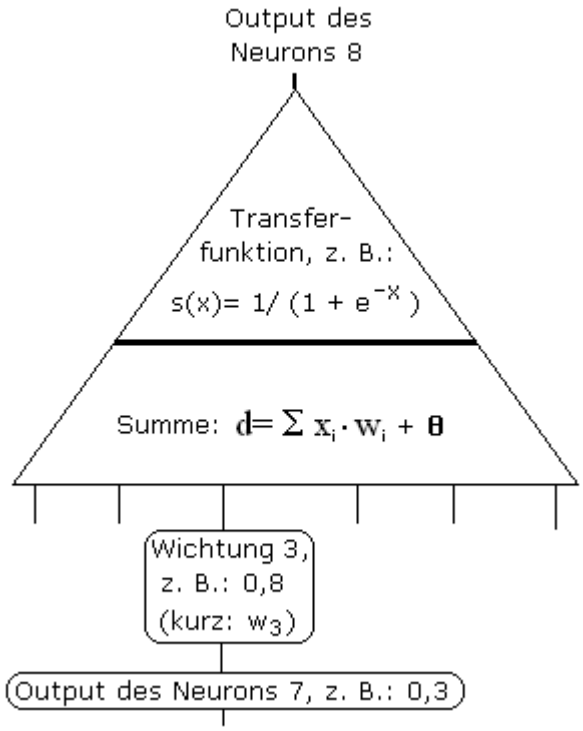
Die Neuronen im QBasic-Programm sind aufsteigend durchnummeriert. Die Neuronen 1 und 2 bilden ausschließlich die Eingangsdaten ab, sie bilden den Inputlayer. Die Neuronen 3 bis 7 bezeichnet man auch als Hidden Layer, da sie in keinerlei Wechselwirkung mit der Umwelt stehen. Sie empfangen ihren Eingang vom Inputlayer und geben ihre Ausgabe an den Outputlayer weiter. Der Outputlayer wird durch das Neuron 8 gebildet. Seine Ausgabe liegt aufgrund der mathematischen Modellierung (siehe unten) immer zwischen 0 und 1. Multipliziert man den Ausgabewert mit 600, so kann man ihn als X-Koordinate zwischen 0 und 600 interpretieren. Mit Hilfe des Ausgabewertes wird ein Schutzschild positioniert, welches im Idealfall genau über dem Aufschlagort der Kanonenkugel liegen soll. Dass dies zutrifft, muss das neuronale Netz lernen.

## Die Berechnung des Ausgabewertes

Das Netz als Ganzes kann als mathematische Funktion betrachet werden, welches Paare der Eingabewerte Abschusswinkel und Mündungsgeschwindigkeit auf dem Aufschlagort abbildet:

$f(\alpha, v_0) =$ Aufschlagort

Im Folgenden wird beschriebenen, wie aus den Eingabewerten der Neuronen 1 und 2 der Ausgabewert des Neurons 8 erzeugt wird. Wie das Netz lernt wird weiter unten beschrieben. Zunächst ist es noch wichtig, den Aufbau eines einzelnen Neurons näher zu betrachten:




Die Eingänge eines jedes Neurons sind mit Wichtungsfaktoren (weights) versehen, mit denen eingehende Werte x von tiefer liegenden Neuronen multipliziert werden. Innerhalb des Neurons werden dann alle gewichteten Eingänge gemeinsam mit einem variablen Wert, dem Bias θ (griechischer Buchstabe Theta), aufaddiert zur Summe. Der Bias wird auch threshold oder Schwellenwert genannt.

Die Summe wird dann innerhalb einer Transferfunktion umgewandelt in die Ausgabe. Für viele Zwecke als Transferfunktion geeignet ist die sogenannte Sigmoid-Funktion: $s(x)= 1/ (1 + e^{-a \cdot x})$. Diese Funktion wandelt eingehende Werte immer in eine Zahl zwischen 0 und 1 um. Der Faktor a wird hier gleich 1 gesetzt und entfällt in der Darstellung.

Um nun also die Ausgabe eines Neurons zu berechnen, müssen folgende Schritte der Reihe nach durchgeführt werden:

Zuerst wird die Summe der gewichteten Ausgänge der vorgeschalteten Neuronen und des Bias gebildet:

$$d = \Sigma x_i \cdot w_i + \theta$$

Anschließend wird auf das Ergebnis d die Sigmoid-Funktion angewendet (x=d):

$$s(d) = 1 / (1 + e^{-d})$$

Dieser Wert s(d) ist der Output des Neurons, welcher nun zu jedem einzelnen Neuron der nächst höheren Ebene weitergeleitet wird (forward).

## Der Lernalgorithmus "Backpropagation"

Um das Netzwerk zu trainieren, führt man die folgenden Schritte durch:

1. Wende die Eingabewerte auf das Netzwerk an.
2. Berechne den Ausgabewert des Netzwerkes.
3. Vergleiche die Ausgabe mit dem gewünschten Ergebnis. Der Unterschied ist der Fehler (error).
4. Verändere die Wichtungen und den Schwellenwert θ für alle Neuronen unter Verwendung des Fehlers.
5. Wiederhole diese Schritte solange, bis der Fehler ein akzeptables Maß nicht mehr überschreitet (z. B. 1%)

Die Anpassung der Wichtungen und des Schwellenwertes erfolgt für die Neuronen des Output Layers anders als für die Neuronen des Hidden Layers. Die Schwellenwerte und Wichtungen der Inputneuronen bleiben unverändert.

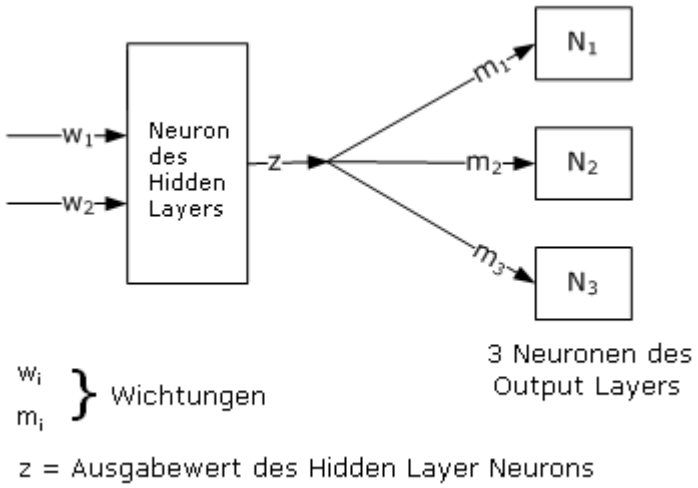
Bevor das Verfahren erklärt wird, sei noch die Lernrate λ eingeführt. Für den Output Layer und den Hidden Layer wird jeweils eine unterschiedliche Lernrate definiert. Das Lernergebnis hängt wesentlich von der geeigneten Auswahl der beiden Werte ab. Für das hier dokumentierte neuronale Netz hat sich eine Lernrate für den Outputlayer von 4 und für den Hidden Layer von 8 bewährt.

Das Lernen für die Neuronen des Output Layers (hier nur das Neuron 8) wird wie folgt durchgeführt:

Berechnung des Fehlers: $e = z * (1 - z) * (y - z)$
Berechnung der Korrektur der Wichtung $w_i$: $\Delta w_i = \lambda \cdot e \cdot x_i$

e = Fehler
z = tatsächlicher Ausgabewert des Ausgabeneurons
y = Wert der hätte ausgegeben werden sollen (Zielwert)
λ = Lernate

Zu jeder Wichtung der Output Neuronen wird dann der Wert $\Delta w_i$ hinzuaddiert. Das Lernen für die Neuronen des Hidden Layers sieht etwas anders aus:




1. z sei der Ausgabewert des Hidden Layer Neurons wie oben gezeigt.
2. $m_i$ sei die Wichtung des Neurons $N_i$ in dem Layer oberhalb des Hidden Layers. Dies ist also die Wichtung mit der die Ausgabe des Hidden Layer Neurons multipliziert wird.
3. $e_i$ sei der Fehlerwert des Neurons $N_i$.
4. r sei die Anzahl der Neuronen in dem Layer oberhalb des Hidden Layers. (In dem Bild oben ist r=3.)

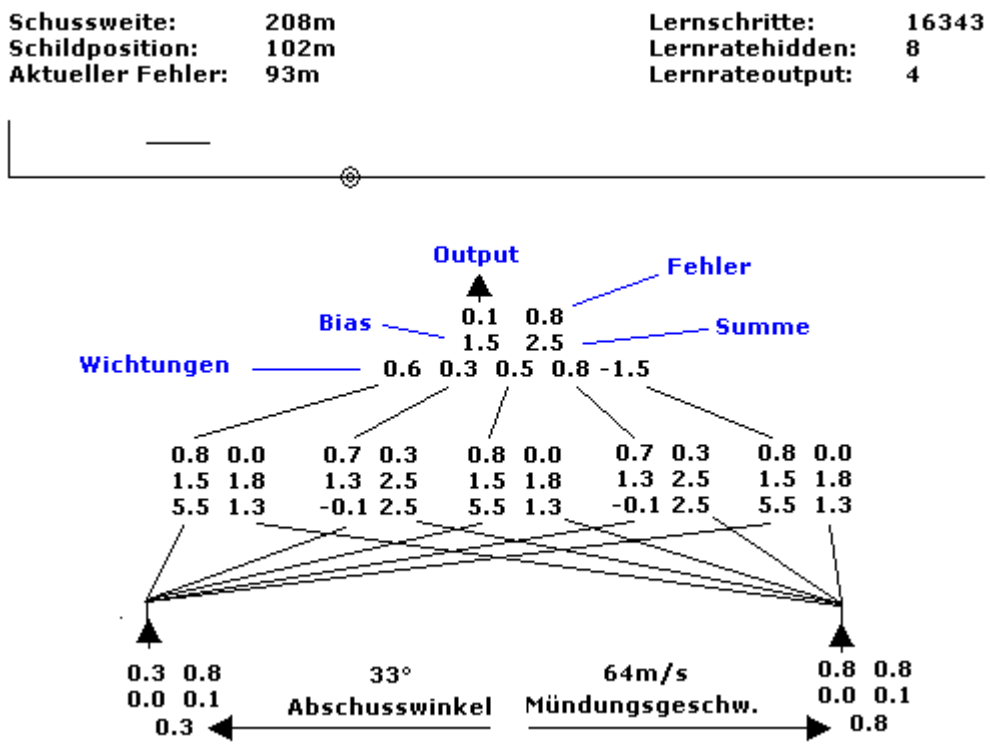
| | | |
|---|---|---|
| Fehler des Hidden Layer Neurons: | $e = z \cdot (1 - z) \cdot \Sigma m_i \cdot e_i$ | (i=1 bis r) |
| Korrektur des Bias | $\Delta\theta = \lambda \cdot e$ | |
| Korrektur der Wichtung | $\Delta w_i = \Delta\theta \cdot x_i$ | |

Zum Bias (Schwellenwert) und zur Wichtung werden die jeweiligen Korrekturen hinzuaddiert.

Es ist zu beachten, dass die Berechnung der Korrekturen rückwärts vom Output Layer über die oder den Hidden Layer in Richtung Input Layer zu erfolgen hat (Backpropagation). Zuerst werden die Fehler aller Neuronen berechnet. Erst dann werden die Wichtungen und die Schwellenwerte aller Neuronen gemeinsam für das ganze Netzwerk aktualisiert.

## Der Bildschirm während der Laufzeit

Die folgende Abbildung zeigt den Bildschirm zur Laufzeit des Programms. (Die Zahlenwerte sind willkürlich gewählt und stellen keinen tatsächlichen Rechengang dar.)




Die Schussweite wird mit Hilfe der Formeln zum schiefen Wurf berechnet und in Metern angezeigt. Die Schildposition ist die durch das neuronale Netz vermutete Schussweite in Metern. Um diesen Wert zu erhalten wird der aktuelle Output des Netzes (aufgrund der Sigmoid Funktion immer zwischen 0 und 1) mit 600 multipliziert. Der aktuelle Fehler stellt den über die letzten 10 Schüsse gemittelten Betrag des Abstandes des Schildes vom Aufschlagort (Schussweite) der Kugel dar.

Die Anzahl der Lernschritte wird oben rechts angezeigt. Sichtbare Erfolge stellen sich nach einigen tausend Lernschritten ein. Am Anfang liegen die mittleren Fehler oft bei 100 bis 200m. Nach 20.000 Schritten sollten Fehler über 100m die Ausnahme sein.

Unterhalb des Textblockes werden die Schussweite und die Schildposition auf einer X-Achse mit der Länge 600 Pixel angezeigt.

In der unteren Bildschirmhälfte werden die einzelnen Neuronen in Anlehnung an das Netzwerkschema 2-5-1 angezeigt.

Der Programmlauf kann durch Drücken der Taste q (quit) unterbrochen werden.

## Zum Stand der aktuellen Version und Download

Die Anwendung kann als exe- (47 kB) oder als bas-Datei (11 kB) heruntergeladen werden über www.philbot.de/download/2007_neuronal. Die bas-Datei benötigt eine QBasic Entwicklungsumgebung, um sie laufen zu lassen. Der Vorteil der bas-Version ist, dass der Quelltext einsehbar und editierbar ist. Die exe-Datei läuft ohne Installation und sonstigen Hilfsdateien, sie zeigt aber nicht den Quelltext an.

Der Entwicklungsstand vom 19. Februar 2007 ist über folgende Punkte gekennzeichnet:

- Ungetestete Beta-Version. Ist der Algorithmus richtig umgesetzt?
- Ein Speichern der erlernten Wichtungen ist nicht möglich.
- Die Ansicht kann nicht verlangsamt werden, etwa um den Rechengang Schritt für Schritt nachzuvollziehen.