Backpropagation

Ausarbeitung zum Seminar Softcomputing

Lars Kaden

September 2000

Institut für Informatik

Seminarleitung: N. Weicker, Prof. Dr. V. Claus

Inhalt

INHALT 2

1 EINLEITUNG 3

2 FORMALISIERUNG 4

2.1 Durch Neuronales Netz berechnete Funktion 4

2.2 Lernen eines Neuronalen Netzes 6

2.2.1 Fehlerfunktion 6

2.2.2 Gesuchte Gewichtsbelegung 7

2.2.3 Minimierung von E^M 8

3 HERLEITUNG VON BACKPROPAGATION 9

3.1 Einfluss eines Neuroneneingangs 10

3.2 Einfluss eines Neuronenausgangs 11

3.3 Ausgangseinfluss eines Ausgabeneurons 11

3.4 Backpropagation für Neuronen der Zwischenschichten 12

3.5 Rekursionsformel 14

3.6 Berechnung der Gewichts- und Schwellwertänderungen 16

4 BEWERTUNG VON BACKPROPAGATION 18

4.1 Vorteile und Besonderheiten von Backpropagation 18

4.2 Ergänzungen zum Verfahren 19

4.3 Nachteile, Probleme von Backpropagation + Abhilfen 19

5 ZUSAMMENFASSUNG 22

LITERATUR 23

1 Einleitung

Neuronale Netze bilden einen Unterbereich des Softcomputings, eines Gebiets, das sich u. a. mit dem Berechnen von Näherungslösungen an Stelle von exakten Lösungen befasst. Die spezielle Aufgabenstellung Neuronaler Netze besteht darin, mehrdimensionale Funktionen anstatt exakt zu berechnen anzunähern. Dieser Kompromiss ist notwendig, um mit relativ geringem Aufwand komplizierte und nicht formalisierbare Funktionen modellieren zu können.
Funktionen mit diesen Eigenschaften treten vor allem bei der Simulation natürlicher, physikalischer, nicht idealisierter Weltausschnitte auf. Hierzu zählen beispielsweise Robotersteuerung in komplexen physikalischen Umgebungen, Zuordnung von Gesichtsfotografien zu Personen und Vorhersage von Börsenkurstendenzen.
Das menschliche und tierische Gehirn setzt sich in erster Linie mit nicht sinnvoll formal und sprachlich zu beschreibenden Entscheidungsfindungen auseinandersetzen. Die Bewältigung von sensorischen und motorischen Alltagsproblemen ist ein typischer Repräsentant der beschriebenen Funktionenklasse. Daher liegt es nahe, das Gehirn allgemein als Vorbild-Mechanismus für derartige Problemfelder zu betrachten.
Das Neuronale Netz ist in diesem Sinne ein stark vereinfachtes Modell der Gehirnstruktur. Nach biologischem Vorbild besteht es aus Neuronen und dazwischen liegenden Verbindungen, die Information in eine Richtung weiterleiten. Wie über reale Nervenverbindungen erhält ein Neuron über die Verbindungen von anderen Neuronen zu ihm die Ausgabewerte der Vorgängerneuronen. Sein eigener über die Ausgänge verschickter Ausgabewert wird durch eine gewichtete Summe der Eingangswerte mit Schwellwert und meist diskreter Aktivierungsfunktion bestimmt. Das Neuronale Netz besteht aus Eingabeneuronen, inneren Neuronen und Ausgabeneuronen, die biologisch interpretiert für Wahrnehmungsrezeptoren, Gehirnstruktur und Nervenenden zur motorischen Steuerung stehen können.
Bleibt man beim Gehirn als Vorbild, ergibt sich als weiterer Aspekt die Lernfähigkeit des Neuronalen Netzes, die formal als schrittweise Verbesserung der Funktionsapproximation anhand von vorgelegten Sollwerten an den Ausgangsneuronen erfolgt. Der Lernprozess kann dann Einfluss auf die vom Netz berechnete Funktion nehmen, indem er die in den Neuronen befindlichen Gewichts- und Schwellwerte anpasst. Für solches Lernen existieren verschiedene heuristische Verfahren, die eingeschränkt auf bestimmte Funktionenmengen mit hoher Wahrscheinlichkeit gegen die Gewichts- und Schwellwertbelegung des gewünschten Verhaltens konvergieren.
Eines dieser heuristischen Verfahren ist **Backpropagation**. Backpropagation ist das klassische Lernverfahren für **Feed-Forward-Netze** (Neuronale Netze ohne gerichtete Verbindungszyklen).
Bei diesem Ansatz wird zunächst die Berechnung des Neuronalen Netzes mathematisch formalisiert. Ausgehend hiervon kann dann die Güte der Berechnung definiert werden, die in Form einer Funktion über Gewichten und Schwellwerten als Differenz zwischen gewünschtem und tatsächlichem Verhalten dargestellt wird. Diese Funktion gilt es dann konsequenterweise zu minimieren, wozu sich das Gradientenverfahren aus der Numerischen Mathematik als geeignet erweisen wird. Diesen ersten Problemen widmet sich Kapitel 2. Im dritten Kapitel wird mit Hilfe analytischer Umformungen das Gradientenverfahren in eine rekursive Berechnung über die Schichtenhöhe umgewandelt, mit deren Hilfe die Gewichts- und Schwellwertänderungen pro Trainingseinheit durch einfache arithmetische Operationen und Propagieren von Zwischenwerten über die Netzverbindungen ermittelt werden können. Zuletzt werden im vierten Kapitel noch einige Aussagen über das generelle Verhalten von

Backpropagation und das in bestimmten Spezialfällen getroffen. In Fällen mit nachteiligem Verhalten werden auf Backpropagation aufbauende Alternativlösungen vorgestellt.

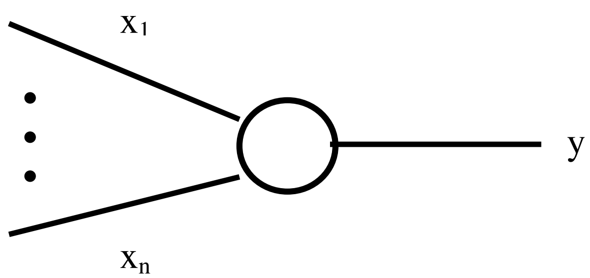
2 Formalisierung

Backpropagation besteht im Wesentlichen aus Verfahren der Numerischen Mathematik. Um diese in Zusammenhang mit Neuronalen Netzen bringen und ihre Eigenschaft als Lernverfahren aufzeigen zu können, müssen die Berechnung eines Neuronalen Netzes, der Fehler unter einer Menge an Trainingsbeispielen und das optimale Neuronale Netz formal definiert werden.

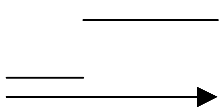
2.1 Durch Neuronales Netz berechnete Funktion

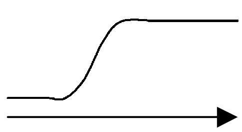
Neuronale Netze werden mathematisch vollständig durch eine Semantikfunktion beschrieben, die jedem Neuronalen Netz die durch es berechnete mehrdimensionale Funktion zuordnet.
Die Zuordnung einer Funktion ist nach dem im Folgenden beschriebenen Schema nur für Feed-Forward-Netze möglich. In Netzen mit Zyklen würde durch das rekursive Einsetzen keine endliche Funktion entstehen.

Zunächst muss die durch ein einzelnes Neuron berechnete Funktion definiert werden:



Ein Neuron mit den Gewichten $w_1, ..., w_n$ und dem Schwellwert θ berechnet die Funktion $y = f(w_1x_1 + ... + w_nx_n + \theta)$.
f ist die sogenannte Aktivierungsfunktion:





z.B.: sgn (x) $\frac{1}{1+e^{-x}}$

Neben den Eingangsgewichten, die den unterschiedlichen Einfluss der Eingänge auf den Ausgang festlegen, ermöglicht die Aktivierungsfunktion eine diskrete Fallunterscheidung. Um später das Lernverfahren mit Hilfe von Differentialrechnung zu ermöglichen, wird die Sigmoid-Aktivierungsfunktion (Bild rechts) angewandt. Sie ist eine differenzierbare Annäherung an die Signum-Funktion (Bild links).

Ein Feed-Forward-Netz berechnet dann die Funktion, die durch rekursives Einsetzen der Neuronenausgangsfunktionen in die zugehörigen Eingangsvariablen der nächsten Schicht entsteht.

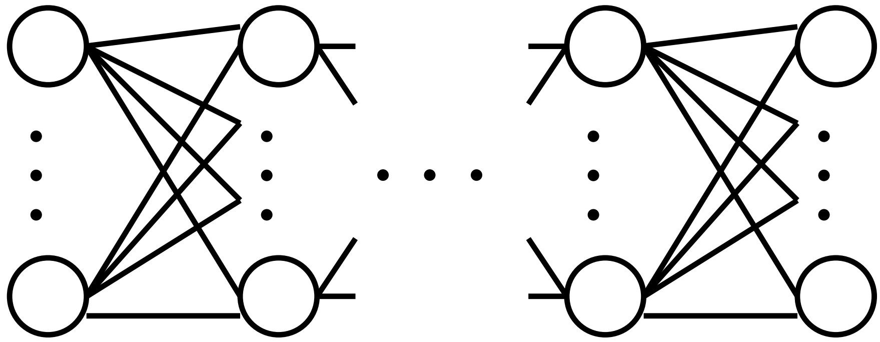
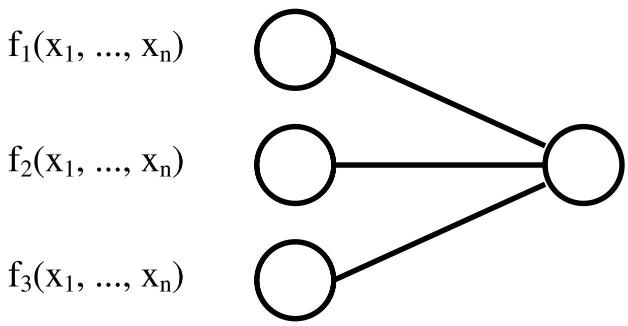


Bild: Schichtenskizze eines Feed-Forward-Netzes

D.h., ist die Funktion in Abhängigkeit der Eingangsneuronen bis zu den Ausgängen der Schicht s bekannt, kann die Funktion durch Einsetzen in die (oben beschriebenen) Einzelneuronenfunktionen der Schicht s+1 in Abhängigkeit der Eingangsneuronen bis zu den Ausgängen der Schicht s+1 dargestellt werden.

Im folgenden Beispiel wird die Funktion g des aus vier Neuronen bestehenden Feed-Forward-Netzes aus der Einzelfunktion f des Ausgangsneurons und den Einzelfunktionen f_1, f_2 und f_3 der Eingangsneuronen berechnet:



$$g(x_1, ..., x_n) = f\big(w_1 f_1(x_1, ..., x_n) + w_2 f_2(x_1, ..., x_n) + w_3 f_3(x_1, ..., x_n) + \theta \big)$$

2.2 Lernen eines Neuronalen Netzes

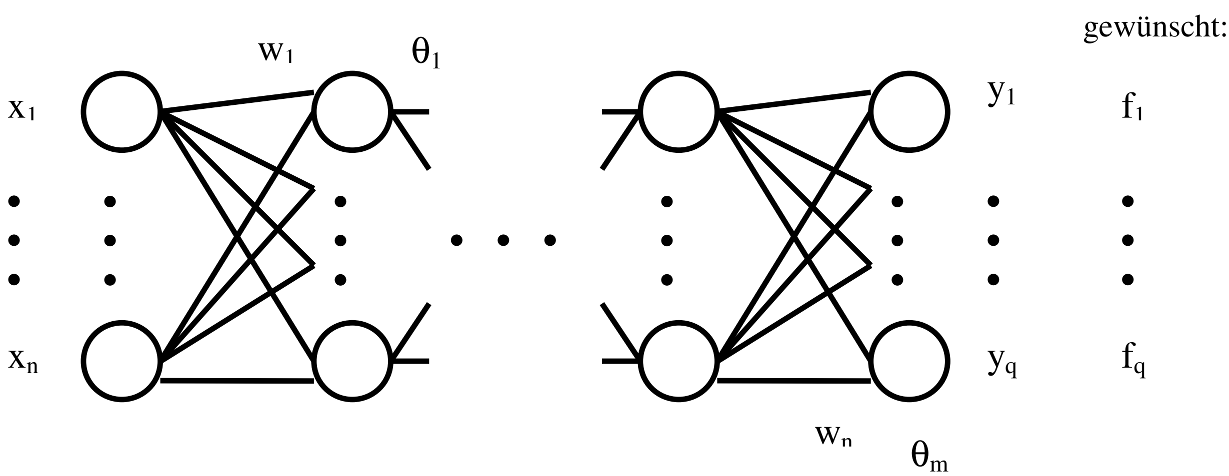
Lernen, d.h. die berechnete Funktion an die gewünschte anpassen, kann ein Neuronales Netz also durch schrittweises Anpassen der Gewichte und Schwellwerte in den einzelnen Neuronen.
Die anzunähernde Funktion wird durch endlich viele Beispiele von Ein- Ausgabepaaren (x_i, y_i) beschrieben.

2.2.1 Fehlerfunktion

Um die zur Anpassung nötigen Gewichts- und Schwellwertänderungen heuristisch berechnen zu können, wird eine Bewertung des Fehlers E^x des Neuronalen Netzes bei einer bestimmten Eingabe x und einer gewünschten Ausgabe y benötigt.
Im Fall von Backtracking wird hierzu das Quadrat der Euklidischen Norm über der Differenz von Ist (NN)- und Sollausgabewert (F) verwendet.

Fehlerfunktion:

$$E^x (w_1, ..., w_n, \theta_1, ..., \theta_m) = \| \underset{\substack{\uparrow \\ \text{NN-Ausgabe-Vektor} \\ \text{zur Eingabe x}}}{NN^{w,\,\theta} (x_1, ..., x_p)} - \underset{\substack{\uparrow \\ \text{Muster-Ausgabe-Vektor}}}{F(x_1, ..., x_p)} \|_2^2$$



$$\rightarrow E^x (w_1, ..., w_n, \theta_1, ..., \theta_m) = (f_1 - y_1)^2 + ... + (f_q - y_q)^2$$

Die Fehlerfunktion ordnet dann den Gewichten und Schwellwerten eines Neuronalen Netzes den Fehler unter einer bestimmten Eingabe x zu. NN ist die nach obigem Verfahren berechnete Funktion des Neuronalen Netzes abhängig von Gewichten und Schwellwerten.

Den Fehler der Annäherung an die gewünschte Funktion definiert man als die Summe der Fehler auf allen Ein- Ausgabebeispielen:

Fehlerfunktion für **Menge M** von **Mustern**:

$E^M(w_1, ..., w_n, \theta_1, ..., \theta_m) =$
$E^{x1}(w_1, ..., w_n, \theta_1, ..., \theta_m) + ... + E^{xn}(w_1, ..., w_n, \theta_1, ..., \theta_m)$

mit $x_1, ..., x_n$ Eingaben von Mustern aus M.

2.2.2 Gesuchte Gewichtsbelegung

Aus den Fehlerdefinitionen ergibt sich die sinnvolle Definition einer gewünschten Gewichts- und Schwellwertbelegung. Gesucht wird eine Belegung, unter der die Gesamtfehlerfunktion E^M einen möglichst kleinen Wert annimmt:

gesuchte Gewichtsbelegung:

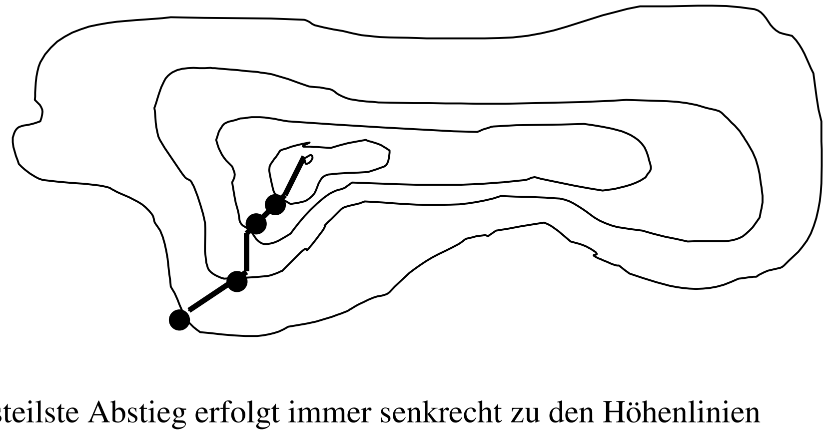
Belegung $(w_1, ..., w_n, \theta_1, ..., \theta_m)$ mit $E^M(w_1, ..., w_n, \theta_1, ..., \theta_m)$ **möglichst klein**

Mit dieser Festlegung wurde das Problem der optimalen Gewichts- und Schwellwertbestimmung auf das analytische Problem der Minimierung einer Funktion reduziert. Um die Effizienz des Lernverfahrens zu garantieren, kann dies nicht durch symbolisches Differenzieren und Lösen von Gleichungen geschehen.
Daher wird die Gewichtsanpassung numerisch gelöst. Ein klassisches heuristisches Verfahren zur Minimierung eines Funktionswerts ist die Gradientenmethode (steilste Abstieg-Methode).

steilste Abstieg-Methode:

Von jedem berechneten Punkt des Definitionsbereichs aus bewegt man sich in **Richtung des steilsten Abstiegs** um eine zum Abstieg proportionale Länge zum nächsten berechneten Punkt hin.

An folgendem graphischen Beispiel (f: $R^2 \rightarrow R$, veranschaulicht durch Höhenlinien) kann das Konvergenzprinzip veranschaulicht werden:



Der steilste Abstieg erfolgt immer senkrecht zu den Höhenlinien

Die Richtung des steilsten Anstiegs an einer bestimmten Stelle wird durch den Gradienten beschrieben, der auch eine zum Anstieg proportionale Länge besitzt. Hieraus ergibt sich der negative Gradient als Richtung des steilsten Abstiegs.
Die Konvergenz dieses Verfahrens tritt mit sehr hoher Wahrscheinlichkeit ein. Es gibt jedoch durchaus Funktionen, in denen statt Konvergenz Oszillation oder sogar Divergenz auftritt.

2.2.3 Minimierung von E^M

Auf die spezielle Fehlerfunktion E^M bezogen wird in jedem Optimierungsschritt vom Gewichts- und Schwellwertvektor der Gradient von E^M unter dieser Gewichts- und Schwellwertbelegung subtrahiert:

$$(w_1, ..., w_n, \theta_1, ..., \theta_m)_{neu} =$$
$$(w_1, ..., w_n, \theta_1, ..., \theta_m)_{alt} - \nabla E^M(w_1, ..., w_n, \theta_1, ..., \theta_m) \circ (w_1, ..., w_n, \theta_1, ..., \theta_m)_{alt}$$

Spaltet man die Vektoraddition in ihre einzelnen Komponenten auf, ergeben sich folgende Anpassungsfunktionen:

$$\Delta w_i = \frac{\partial E^M}{\partial w_i} \circ (w_1, ..., w_n, \theta_1, ..., \theta_m) \qquad w_{ineu} = w_{ialt} - \Delta w_i$$

$$\Delta \theta_i = \frac{\partial E^M}{\partial \theta_i} \circ (w_1, ..., w_n, \theta_1, ..., \theta_m) \qquad \theta_{ineu} = \theta_{ialt} - \Delta \theta_i$$

Da das Neuronale Netz die einzelnen Trainingsbeispiele nacheinander vorgelegt bekommt, müssen die partiellen Ableitungen von E^M als Summe der partiellen Ableitungen der E^{xi}, x_i Eingaben von Mustern aus M, berechnet werden. Dies ist nach der Summenregel der Differenziation möglich:

$$\Delta w_i = \frac{\partial E^M}{\partial w_i} \circ (w_1, ..., w_n, \theta_1, ..., \theta_m)$$

$$= \frac{\partial}{\partial w_i} \left(E^{x1}(w_1, ..., w_n, \theta_1, ..., \theta_m) + ... + E^{xn}(w_1, ..., w_n, \theta_1, ..., \theta_m) \right)$$

$$= \frac{\partial}{\partial w_i} E^{x1}(w_1, ..., w_n, \theta_1, ..., \theta_m) + ... + \frac{\partial}{\partial w_i} E^{xn}(w_1, ..., w_n, \theta_1, ..., \theta_m)$$

Die Berechnung der Schwellwertänderungen $\boldsymbol{\Delta\theta_i}$ für alle Trainingsbeispiele kann mit einer analogen Begründung als Aufsummierung der Schwellwertänderungen für jedes einzelne Trainingsbeispiel erfolgen.

3 Herleitung von Backpropagation

Backpropagation besteht aus einer geschickten Aufteilung der Berechnung von Gewichts- und Schwellwertänderungen nach dem Gradientenverfahren. Die Aufstellung von Rekursionsformeln macht es möglich, die Änderungen in einer Neuronenschicht mit sehr einfachen arithmetischen Operationen aus den Änderungen der um eins tieferen Schicht zu berechnen. Für die Aktualisierung eines Neurons ergibt die Aufteilung der Berechnung, dass nur die Änderungswerte der direkten Nachfogerneuronen relevant sind. Diese können jeweils kodiert in einem Wert, dem sogenannten Eingangseinfluss, über die Verbindung nach vorne übermittelt werden. Im Folgenden werden zunächst die Größen Eingangs- und Ausgangseinfluss eingeführt, die Zwischenergebnisse der rekursiven Berechnungseinteilung beschreiben. Im Anschluss daran erfolgt dann die analytische Herleitung der Rekursionsformeln ausgehend von den Ausgangsneuronen bzw. der Berechnung der Gewichts- und Schwellwertänderungen innerhalb der einzelnen Neuronen aus den über die Verbindungen propagierten Zwischenergebnissen.

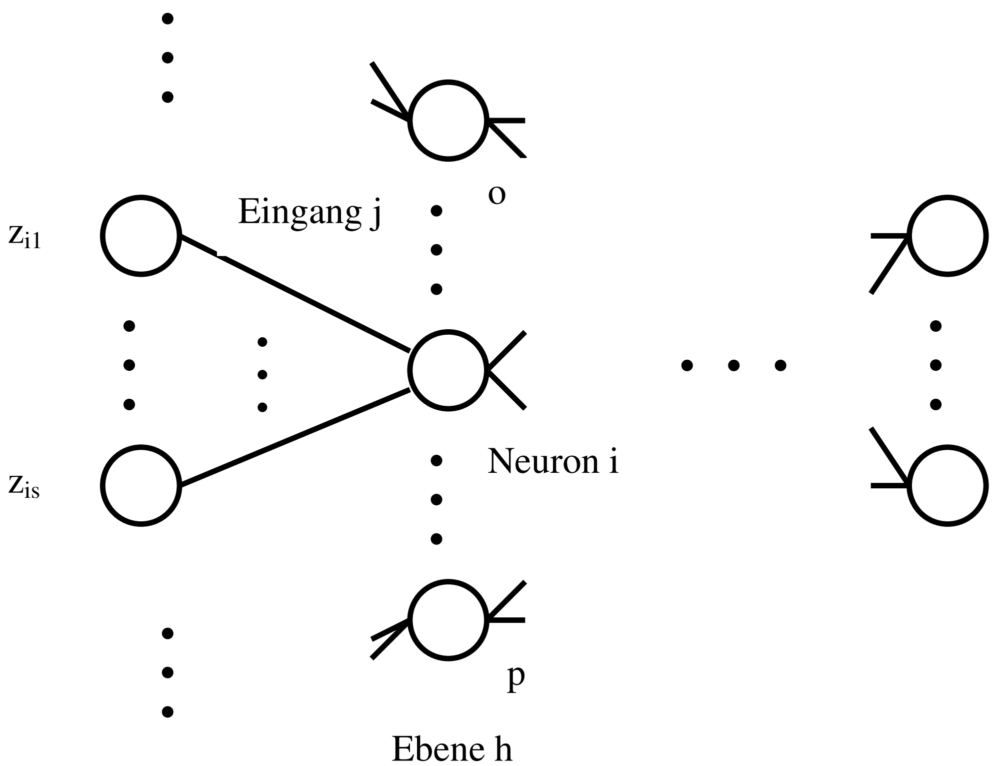
3.1 Einfluss eines Neuroneneingangs

Der Einfluss eines Neuroneneingangs ist die partielle Ableitung der Fehlerfunktion nach diesem Eingang. Dies kann als Maß dafür gedeutet werden, in welchem Verhältnis die Änderung der Fehlerfunktion zur Änderung des Neuroneneingangswerts liegt.

$$e_{ij} = \frac{\partial E_h}{\partial z_{ij}} (z_{o1}, \ldots, z_{pr}) \circ (w_1, \ldots, w_n, \theta_1, \ldots, \theta_m, x_1, \ldots, x_w)$$

(Einfluss des j-ten Eingangs des i-ten Neurons auf den Fehler bei Trainingseingabe x)

E_h ist hierbei die Fehlerfunktion in Abhängigkeit der Eingänge $z_{o1}, \ldots, z_{pr}$ der Neuronen der h-ten Schicht, in der sich das Neuron i befindet. Für $z_{o1}, \ldots, z_{pr}$ kann man sich in die Ableitung dann die Werte eingesetzt denken, die sich beim Anliegen der Eingabe x an den Neuroneneingängen der h-ten Schicht befinden, wobei diese durch Differenzieren eigentlich entfallen.



Ebene h wird als **erstes** hidden layer betrachtet,
alle Eingänge erhalten ‚gedachte' Eingangsneuronen $z_{o1}, \ldots, z_{pr}$,
die ‚gedachten' Eingangsneuronen erhalten Werte, die sie bei komplettem Netz hätten.
➜ E_h entsteht nach beschriebenem Rekursionsverfahren

3.2 Einfluss eines Neuronenausgangs

Der Einfluss eines Neuronenausgangs wird analog zum Eingangseinfluss als partielle Ableitung der Fehlerfunktion nach diesem Ausgangswert definiert:

$$a_i = \frac{\partial E_h}{\partial y_i} (y_p, ..., y_q) \circ (w_1, ..., w_n, \theta_1, ..., \theta_m, x_1, ..., x_w)$$

(Einfluss des Ausgangs von Neuron i auf den Fehler bei Trainingseingabe x)

Hierbei wird der Fehler durch E_h in Abhängigkeit von Neuronenausgängen der Schicht h dargestellt.

3.3 Ausgangseinfluss eines Ausgabeneurons

Im Gegensatz zu Neuronen in den Zwischenschichten kann der Ausgangseinfluss eines Ausgabeneurons direkt berechnet werden:

$$a_i = \frac{\partial E_{out}}{\partial y_i} (y_1, ..., y_n) \circ x$$

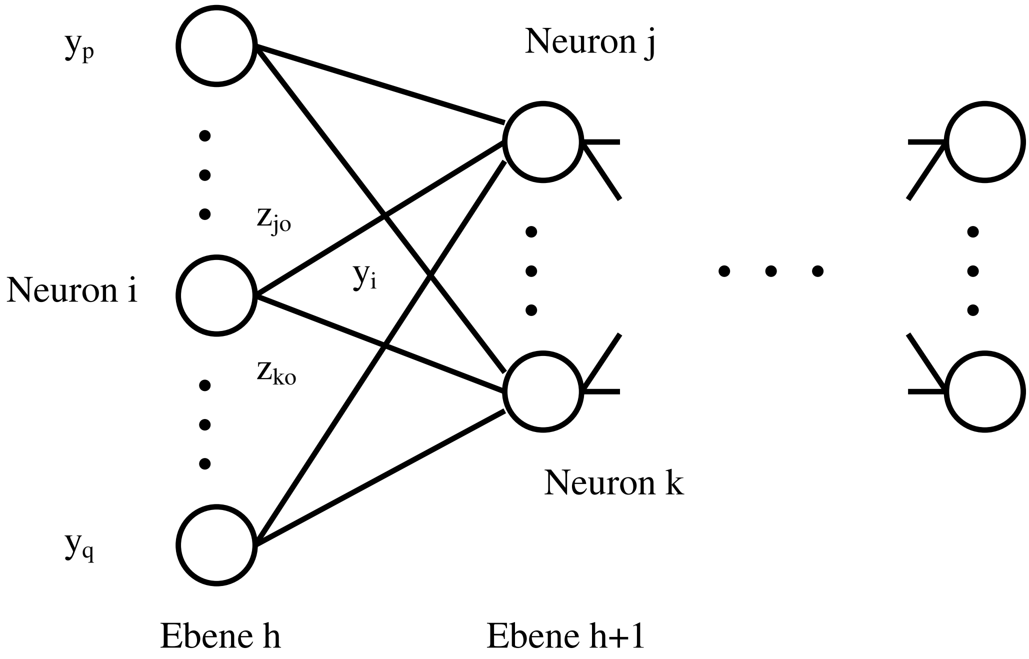
$$= \frac{\partial}{\partial y_i} ((f_1 - y_1)^2 + ... + (f_n - y_n)^2) \circ x$$

Muster-Ausgabe

$$= 2\,(y_i - f_i)$$

Hierbei ist E_{out} die Fehlerfunktion bei Eingabe x in Abhängigkeit von den Ausgaben $y_1, ..., y_n$ der letzten Neuronenschicht. Der einzige Summand, der nach Ableiten $\neq 0$ bleibt, ist der für Neuron i.

3.4 Backpropagation für Neuronen der Zwischenschichten



Wie aus dem Bild ersichtlich, können die Ausgangseinflüsse der Ebene h aus den Eingangseinflüssen der Ebene h+1 berechnet werden:

$$a_i = \frac{\partial E_h}{\partial y_i}(y_p, ..., y_q) \circ (w_1, ..., w_n, \theta_1, ..., \theta_m, x_1, ..., x_w)$$

$$= \frac{\partial E_{h+1}}{\partial y_i}(z_{j1}, ..., y_i, ..., z_{jr}, \ . \ . \ .$$

$$\downarrow$$

$$z_{jo}$$

$$z_{ko}$$

$$\uparrow$$

$$z_{k1}, ..., y_i, ..., z_{kr}) \circ (w_1, ..., w_n, \theta_1, ..., \theta_m, x_1, ..., x_w)$$

$$= \frac{\partial E_{h+1}}{\partial z_{jo}} (z_{j1}, \ldots, z_{jo}, \ldots, z_{jr}) \circ (w_1, \ldots, w_n, \theta_1, \ldots, \theta_m, x_1, \ldots, x_w) +$$

$$\ldots +$$

$$\frac{\partial E_{h+1}}{\partial z_{ko}} (z_{j1}, \ldots, z_{ko}, \ldots, z_{jr}) \circ (w_1, \ldots, w_n, \theta_1, \ldots, \theta_m, x_1, \ldots, x_w)$$

$$= e_{jo} + \ldots + e_{ko}$$

Anschaulich ergibt sich die Berechnung des Ausgangseinflusses als Summe der Eingangseinflüsse der Eingänge, mit denen dieser Ausgang verbunden ist, aus der Überlegung, dass bei einer Änderung des Ausgangswerts sich „parallel“ alle entsprechenden Eingangswerte der nächsten Schicht um den gleichen Betrag ändern und somit sich die Fehleränderungen dieser Eingänge „überlagern“.

3.5 Rekursionsformel

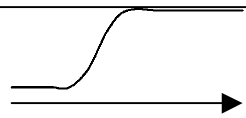
Um sämtliche Neuroneneingangs- und -ausgangseinflüsse rekursiv aus der Netzstruktur berechnen zu können, wird noch eine Formel benötigt, mit der man aus dem Einfluss eines Neuronenausgangs die Einflüsse seiner Eingänge berechnen kann. Diese ergibt sich aus folgenden Differenzialumformungen:

$$e_{ij} = \underbrace{\frac{\partial E_h}{\partial z_{ij}}(z_{o1}, \ldots, z_{pr})}_{\underbrace{\frac{\partial E_h}{\partial y_i}(y_o, \ldots, y_p)}_{a_i} \; \underbrace{\frac{\partial y_i}{\partial z_{ij}}(z_{i1}, \ldots, z_{ir})}_{\frac{\partial}{\partial z_{ij}} f(\underbrace{w_{i1}z_{i1} + \ldots + w_{ir}z_{ir} + \theta_i}_{s_i}) \quad (*)}} \circ (w_1, \ldots, w_n, \theta_1, \ldots, \theta_m, x_1, \ldots, x_w)$$

$$= w_{ij}\, f'(y_i)$$

$$= a_i\, w_{ij}\, f'(y_i)$$

für $f(s_i) = \frac{1}{1+e^{-si}}$



$$f'(s_i) = \frac{1}{1+e^{-si}}\left(1 - \frac{1}{1+e^{-si}}\right) = y_i(1 - y_i)$$

$$= a_i\, w_{ij}\, y_i\, (1 - y_i)$$

Die hier sehr kompakt aufgeführte Umformungsfolge wird im Folgenden noch etwas ausführlicher begründet:

Der Schluss vom Neuronenausgangseinfluss auf den Eingangseinfluss erfolgt durch die Aufspaltung von

$E_h(z_{o1}, \ldots, z_{pr})$ in die Verkettung $E_h(y_o, \ldots, y_p) \circ (y_o(\underbrace{z_{o1}, \ldots, z_{or}}), \ldots, y_p(\underbrace{z_{p1}, \ldots, z_{pr}}))$.

↑ Fehler in Abhängigkeit von Eingängen der Schicht h

↑ Fehler in Abhängigkeit von Ausgängen der Schicht h

jeweilige Neuroneneingänge

Nach der mehrdimensionalen Kettenregel ergibt sich dann

$$\frac{\partial E_h}{\partial z_{ij}}(z_{o1}, \ldots, z_{pr}) = \frac{\partial E_h}{\partial y_o}(y_o, \ldots, y_p)\ \frac{\partial y_o}{\partial z_{ij}}(z_{o1}, \ldots, z_{or})$$

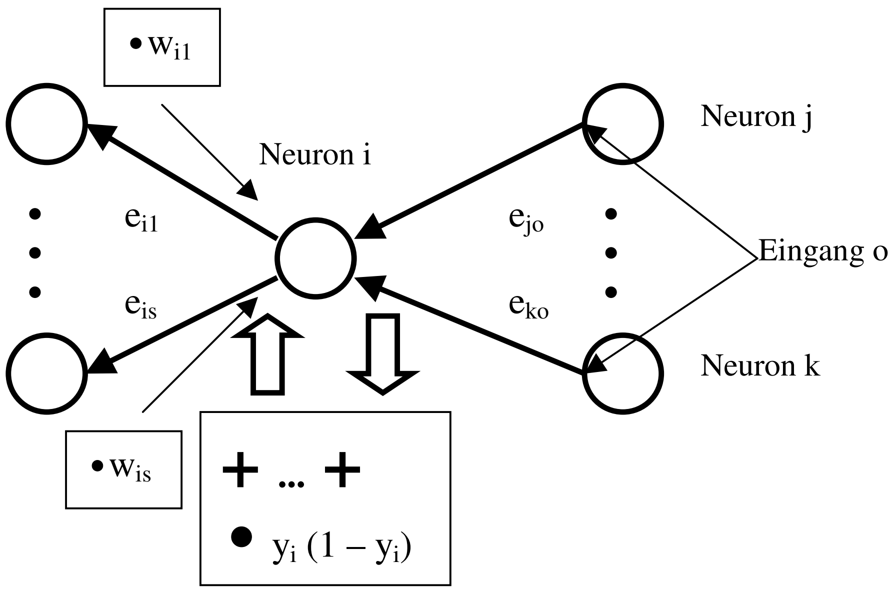
$$+ \ldots + \frac{\partial E_h}{\partial y_p}(y_o, \ldots, y_p)\ \frac{\partial y_p}{\partial z_{ij}}(z_{p1}, \ldots, z_{pr})$$

Da der einzige Summand, für den die y-Funktion z_{ij} enthält, der für Neuron i ist, sind die rechten Faktoren (Ableitungen) der übrigen Summanden und damit auch die Summanden = 0. Somit gilt:

$$\frac{\partial E_h}{\partial z_{ij}}(z_{o1}, \ldots, z_{pr}) = \frac{\partial E_h}{\partial y_i}(y_o, \ldots, y_p)\ \frac{\partial y_i}{\partial z_{ij}}(z_{i1}, \ldots, z_{ir})$$

Die übrige Berechnung ergibt sich dann, wenn man als Aktivierungsfunktion die Sigmoid-Funktion wählt.

Hiermit ist nun das Zurück-Propagieren der Eingangseinflüsse über alle Schichten vollständig über die Netzverbindungen möglich.
Das Propagieren kann wie im folgenden Bild für ein Neuron gezeigt veranschaulicht werden:



Geht man davon aus, dass die Eingangseinflüsse der Neuronen in der Schicht nach Neuron i bereits bekannt sind, können diese über die Verbindungen, deren Einflüsse sie beschreiben, rückwärts an Neuron i geschickt werden. Dort werden sie aufsummiert, woraus der Ausgangseinfluss a_i des Neurons i entsteht. a_i wird nun mit $y_i(1 - y_i)$, wobei y_i die Neuronenausgabe der letzten Berechnung ist, multipliziert. Die Eingangseinflüsse von Neuron i erhält man nun durch die jeweilige Multiplikation ihrer Gewichte mit diesem Wert. Bei rekursiver Anwendung über die Schichthöhe erhalten somit alle Neuronen die Werte ihrer Eingangs- und Ausgangseinflüsse.

3.6 Berechnung der Gewichts- und Schwellwertänderungen

Mit Hilfe analytischer Umformungen ist nun klar wie aus einfachen rekursiven Berechnungen bestimmte Zwischenergebnisse rückwärts durch das Netz propagiert werden können. Das eigentliche Ziel von Backpropagation besteht jedoch letztendlich darin, die Gewichte und Schwellwerte neu zu bestimmen. Dies ist nun in jedem Neuron für sich auf einfache Weise mit Hilfe seines eigenen Ausgangseinflusses möglich:

Die Einflüsse der Gewichte und Schwellwerte lassen sich einfach berechnen durch eine Aufspaltung der Fehlerfunktion in eine Verkettung aus Fehlerfunktion in Abhängigkeit der Neuronenausgänge einer Schicht und die Neuronenausgangsfunktionen in Abhängigkeit ihrer Gewichte und Schwellwerte:

Δ w_{ij} für **x** (Eingabe) aus **a_i** (Einfluss von Neuron i) berechnen:

$$\Delta w_{ij} = \frac{\partial}{\partial w_i} E^x(w_1, ..., w_n, \theta_1, ..., \theta_m)$$

$$= \frac{\partial E_h}{\partial y_i}(z_{o1}, ..., z_{pr}) \circ (w_1, ..., w_n, \theta_1, ..., \theta_m, x_1, ..., x_w)$$

$$\bullet \frac{\partial y_i}{\partial w_i}(w_{i1}, ..., w_{ir}) \circ (w_1, ..., w_n, \theta_1, ..., \theta_m, x_1, ..., x_w)$$

$$= a_i \, z_{ij} \, y_i \, (1 - y_i) \qquad (*)$$

Die einzelnen Umformungsschritte erfolgen analog zur Berechnung des Eingangseinflusses aus dem Ausgangseinfluss.

Analog für die Schwellwerte:

$$\Delta \theta_i = \frac{\partial}{\partial \theta_i} E^x(w_1, ..., w_n, \theta_1, ..., \theta_m)$$

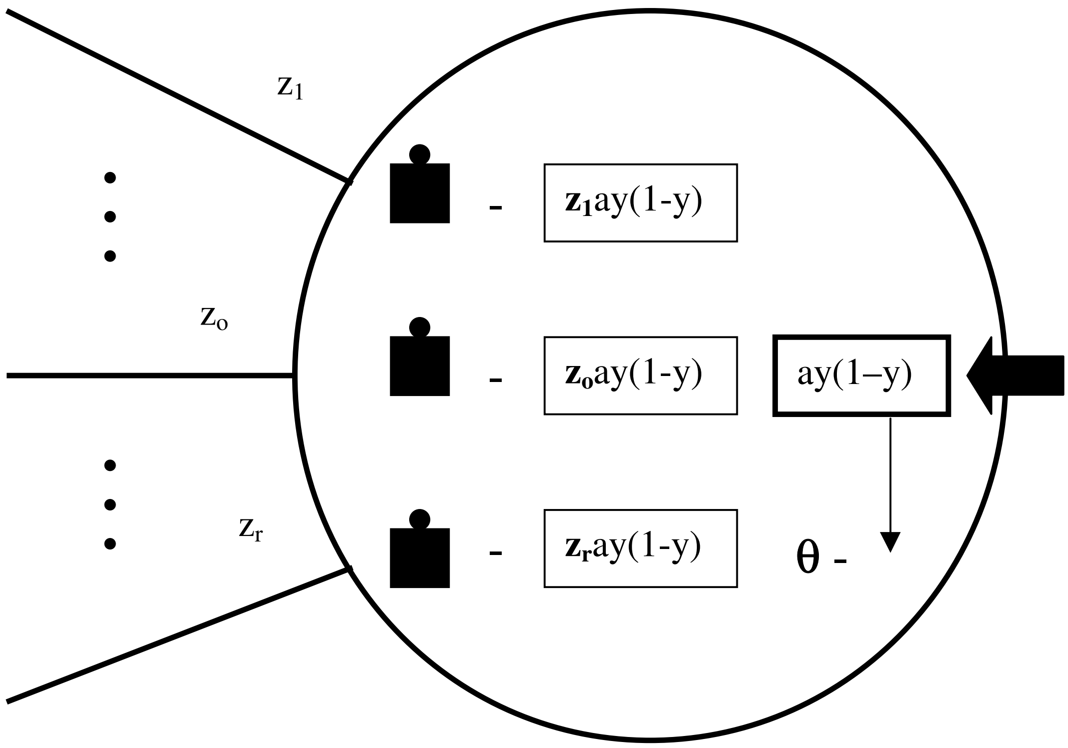
$$= \frac{\partial E_h}{\partial y_i} (z_{o1}, ..., z_{pr}) \circ (w_1, ..., w_n, \theta_1, ..., \theta_m, x_1, ..., x_w)$$

$$\bullet \frac{\partial y_i}{\partial \theta_i} (w_{i1}, ..., w_{ir}) \circ (w_1, ..., w_n, \theta_1, ..., \theta_m, x_1, ..., x_w)$$

$$= a_i\, y_i\, (1 - y_i) \qquad (*)$$

Die Schwellwertänderung ergibt sich auch aus der Überlegung, dass ein Schwellwert als Gewicht betrachtet werden kann, an dessen Eingang bei jeder Berechnung des Netzes der Wert 1 anliegt. Dann erhält man, indem man für den Eingangswert z_{ij} 1 einsetzt aus der Formel für die Gewichtsänderung die für die Schwellwertänderung.

Anhand folgender Grafik kann die Berechnung der neuen Gewichts- und Schwellwerte aus Neuronenausgabe, Ausgangseinfluss und den alten Werten veranschaulicht werden:



Den Ausgangseinfluss a erhält jedes Neuron durch den Backpropagation-Prozess. Dieser wird nach den Berechnungsvorschriften zunächst mit der Ableitung der Aktivierungsfunktion y(1-y) multipliziert. Der Ausgabewert y ist dem Neuron durch die letzte Berechnung bekannt. Dieses Produkt kann dann im Fall des Schwellwerts direkt subtrahiert werden, um den angepassten Schwellwert zu erhalten. Im Fall der Gewichte wird der Änderungswert zuvor noch mit dem Eingabewert am jeweiligen Eingang multipliziert.

4 Bewertung von Backpropagation

Im Folgenden werden noch einige Vorteile von Backpropagation als Grundverfahren für das Lernen Neuronaler Feed-Forward-Netze beschrieben. In zwei Anwendungsfällen sollen anschließend Schwachpunkte von Backpropagation aufgedeckt werden. Solche und andere Beispiele machen den Einsatz weiterentwickelter, auf Backpropagation aufbauender Verfahren sinnvoll, von denen ein klassisches vorgestellt wird.

4.1 Vorteile und Besonderheiten von Backpropagation

- Plausibilität:
  Die Konvergenz der Gewichts- und Schwellwertanpassung ist durch Anwendung der Gradientensuche plausibel. In der Numerik wurde dieses Verfahren intensiv untersucht und es liegen zahlreiche Ergebnisse vor, mit deren Hilfe Aussagen über den Erfolg von Backpropagation bei der Approximation bestimmter Funktionenmengen getroffen werden können. Ist es möglich die zu approximierenden Funktionen in bestimmter Weise einzuschränken, kann die Konvergenz oder das Finden eines absoluten Minimums eventuell sogar bewiesen werden. In jedem Fall ist die Gradientensuche als erfolgreiche Heuristik zur Suche kleiner Werte im allgemeinen Fall anerkannt.

- Ausgleichsfunktion:
  Backpropagation approximiert eine Funktion nicht durch Interpolation, sondern durch Bestimmung einer Ausgleichsfunktion, die unter dem Kriterium einer bestimmten Norm (euklidische Norm) die gewünschte Funktion am besten approximiert. Dies eignet sich besonders für Funktionen, bei denen man in den Trainingsbeispielen mit Verrauschung rechnen muss.
  Weiter ermöglicht dies eine stärkere Gewichtung von Trainingsbeispielen, die zusammengenommen Regelmäßigkeiten aufweisen, und eine Abschwächung von Beispielen, die eine Regelmäßigkeit stören. Hierdurch wird die Fähigkeit eines Neuronalen Netzes zur Generalisierung, die u.a. auf der Filterung von Ausnahmen basiert, ermöglicht.
  Außerdem kann das Neuronale Netz durch Bestimmung einer Ausgleichsfunktion bei fortlaufendem Training einen früheren Lernfehler wieder ausgleichen, indem dieser nicht mehr vorgelegt wird und durch zahlreiche „Trainingsgegenbeispiele“ überlagert und dominiert wird. Bei einer klasischen Interpolation wäre dieser Trainingsfehler nicht mehr rückgängig zu machen, da die Approximationsfunktion hier auf diesen Funktionswert festgelegt wäre. Zusätzlich kann das Neuronale Netz dann auch in Fällen eingesetzt werden, bei denen sich die Umgebungsänderungen ständig ändern.
  Die Bestimmung einer Ausgleichsfunktion ist eine besondere Leistung von Backpropagation, da dies im exakt berechneten Fall schon für lediglich eine Ausgleichsgerade im R^2 sehr aufwendig ist. Backpropagation ermöglicht diese

Berechnung näherungsweise für deutlich kompliziertere höherdimensionale Funktionenfamilien auf effiziente Weise.

- Anpassung durch eigene Struktur:
  Wie durch die Differenzialumformungen gezeigt, kann die Gewichts- und Schwellwertänderung durch ein einfaches Zurückpropagieren der Einflusswerte entlang der Netzverbindungen berechnet werden. Dies ermöglicht also eine Anpassung des Netzes nach biologischem Vorbild aus seiner eigenen Struktur ohne äußeren Kontrollmechanismus. Daher kann ein Neuronales Netz auch als Hardwarestruktur allein mit Neuronen als elementaren Bausteinen aufgebaut werden.

4.2 Ergänzungen zum Verfahren

- Lernrate:
  Bevor bei einem Lernschritt die Gewichtsanpassung ausgeführt wird, können bei Backpropagation die Gewichtsänderungen noch mit einem zusätzlichen konstanten Faktor (Lernrate) multipliziert werden. Hierdurch kann man die „Anpassungsgechwindigkeit“ des Netzes beeinflussen. Es empfiehlt sich, bei stark zerklüfteten Funktionen (viele Sprünge bzw. Vorzeichenwechsel der ersten Ableitung) eine kleine und bei flachen Funktionen eine große Lernrate zu verwenden. Sinnvoll ist es auch am Anfang des Anpassungsprozesses die Lernrate groß einzustellen und über das Trainingsverfahren die Lernrate zu verkleinern.

- Kontinuierliche Gewichtsanpassung:
  Die Gewichtsanpassung erfolgt in den meisten Anwendungsfällen nicht wie bisher beschrieben erst, nachdem alle Beispiele einmal angelegt wurden, sondern sofort nach jedem vorgelegten Beispiel. Hierdurch wird allerdings die Plausibilität der Konvergenz von Backpropagation deutlich eingeschränkt, da nicht mehr in jedem Schritt die Gesamtfehlerfunktion E^M, sondern je eine der Fehlerfunktionen E^x (x Eingabe aus M), minimiert wird. Im Allgemeinen befinden sich die Minima der Summanden in anderen Richtungen als das der Summenfunktion.

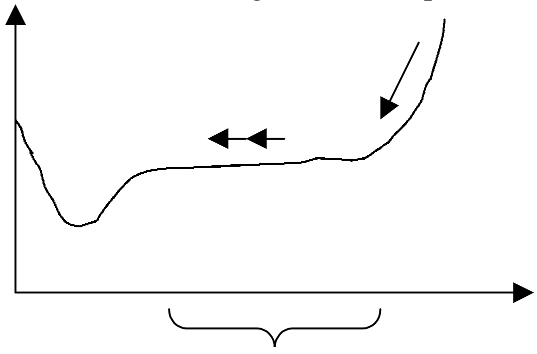
4.3 Nachteile, Probleme von Backpropagation + Abhilfen

Flache Plateaus:

Befindet sich die Gewichts- und Schwellwertbelegung des Neuronalen Netzes in einem Gebiet über dem die Fehlerfunktion nur sehr kleine gleichgerichtete Gradienten besitzt, sich in einiger Entfernung jedoch ein deutlich kleinerer Funktionswert befindet, konvergiert Backpropagation sehr langsam.

An folgender Funktion (R → R) kann dieser Vorgang veranschaulicht werden:

Problem: langsames Überqueren von flachen Plateaus



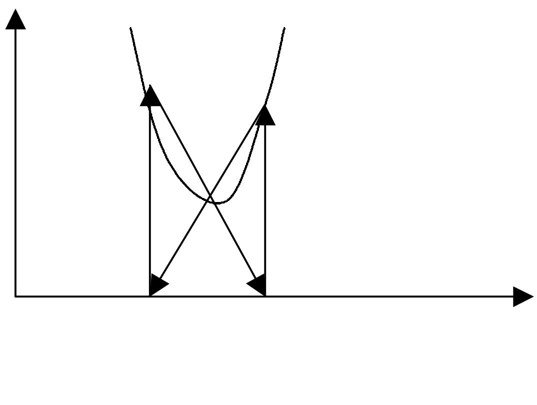
Wegen der kleinen Gradientennormen sind die Sprungweiten im flachen Plateau klein.

Schluchten:

Das umgekehrte Problem gegenüber dem in flachen Plateaus ergibt sich in Schluchten. Befindet sich die momentane Gewichts- und Schwellwertbelegung in einer Schlucht der Fehlerfunktion, d.h. in einem Gebiet, über dem die Fehlerfunktion große Gradienten mit entgegengesetzten Richtungen besitzt, kann es dazu kommen, dass die Optimierungssprünge zu groß sind, und, anstelle sich dem Minimum zu nähern, das Verfahren um das Minimum oszilliert.

Anschaulich wird dies an folgender Beispielfunktion (R → R):

Problem: Oszillieren in Schluchten



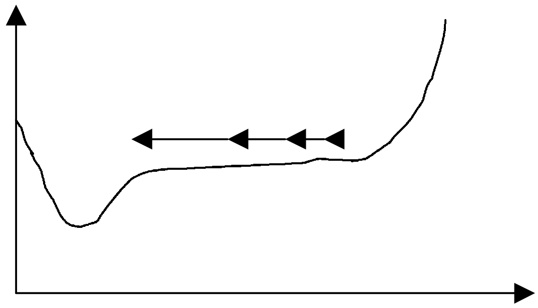
Abhilfe für diese Probleme bringt die Einführung des **Momentum-Term**:

$$\Delta' w_i (t+1) = \alpha\, \Delta\, w_i (t+1) + \beta\, \Delta' w_i (t)$$

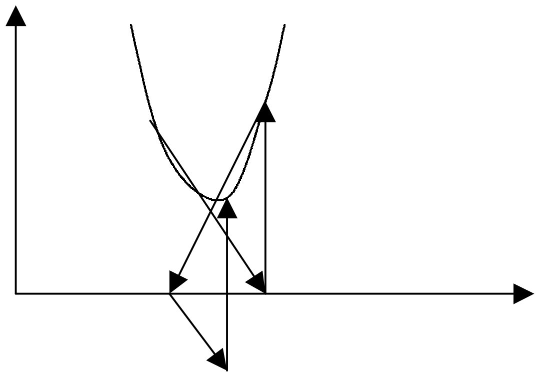
Hier wird die Gewichtsänderung auf modifizierte Weise berechnet. Wie beim herkömmlichen Backpropagation wird die Änderung Δw_i berechnet. Zum Lernschritt t+1 wird allerdings die

Gewichtsänderung $\Delta' w_i(t+1)$ verwendet, die sich aus einer Gewichtung (α, β) der herkömmlichen Änderung $\Delta\, w_i\,(t+1)$ und der Änderung im letzten Schritt $\Delta' w_i\,(t)$ berechnet.
Das hat zur Folge, dass bei gleicher Anpassungsrichtung über mehrere Lernschritte hinweg sich der Betrag des Änderungsvektors stets vergrößert (flache Plateaus), und bei Richtungsänderung in aufeinanderfolgenden Schritten sich die Schrittgröße verkleinert (Schluchten). Dies verhindert beide beschriebenen Konvergenzeinschränkungen.
In folgenden Grafiken kann das Verhalten von Backpropagation mit Momentum-Term veranschaulicht werden:

Flache Plateaus (schnelleres Überqueren des Plateaus durch Momentum-Term):



Schluchten (ausbleibende Oszillation durch Momentum-Term):



Lokale Minima:

Hierin liegt der wesentliche Nachteil von Backpropagation. Backpropagation ist ein lokales Optimierungsverfahren, das zwar mit sehr hoher Wahrscheinlichkeit gegen ein Minimum konvergiert, jedoch gegen ein beliebiges. Dieses ist wiederum mit hoher Wahrscheinlichkeit kein absolutes Minimum, der Gradient jedoch besitzt hier den Wert 0 und es erfolgt keine weitere Suche nach kleineren Werten.
Abhilfe hiervon kann nur ein äußerer Mechanismus schaffen, der durch Neuinitialisierungen in neue Gebiete des Suchraums vordringt und nach zahlreichen getrennten Lernphasen den Suchraum wieder auf die Teilgebiete mit den besseren Werten einschränkt.

5 Zusammenfassung

Backpropagation ist das klassische Lernverfahren für mehrschichtige neuronale Feed-Forward-Netze. Alle Lernverfahren für diesen Netztyp sind Varianten dieses Ansatzes. Es beruht auf einer rekursiven Berechnung der Gewichts- und Schwellwertänderungen entlang den Neuronenverbindungen. Die Berechnung beginnt an den Ausgangsneuronen und wird durch ‚rückwärts propagierte' Zwischenwerte, die sogenannten Eingangseinflüsse, in den Zwischenneuronen bis zur Eingangsschicht fortgesetzt. Die Anpassungswerte der Gewichts- und Schwellwerte werden durch das numerische Gradientenverfahren bestimmt, mit dem die Fehlerfunktion, die die Abweichung von Ist- und Soll-Verhalten misst, minimiert wird. Die Gradientenfunktion lässt sich durch partielle Differenzialrechnung über die Neuronenverbindungen rekursiv aufteilen. Mit dem Momentum-Term u.a. Erweiterungen können Nachteile wie langsame Konvergenz in flachen Plateaus, Oszillation in Schluchten und rein lokale Konvergenz beseitigt werden.

Literatur

Kröse, van der Smagt, An Introduction to Neural Networks (1996), S. 15-90

Andreas Zell, Simulation Neuronaler Netze (1994), Addison-Wesley, S. 110-116