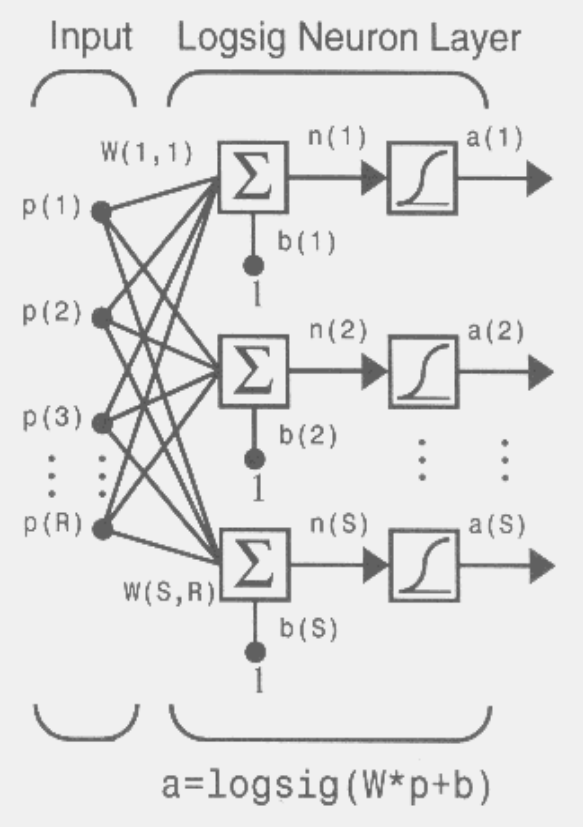

Abb 1 Neuronenverknüpfung

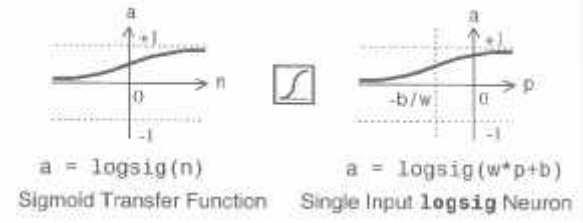

Abb 2 sigmoidale Transferfunktion

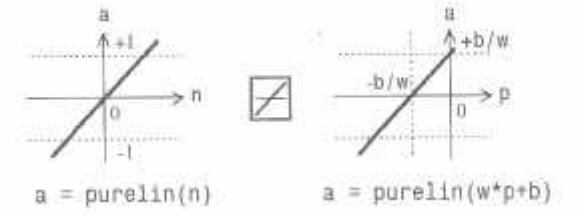

Abb 3 lineare Transferfunktion

Einführung zur Verwendung eines Neuronalen Netzwerkes

Neuronale Netze sind extrem leistungsfähige statistische Verfahren für eine Vielzahl von nicht-linearen Problemstellungen in praktischen Anwendungen. Neuronale Netzwerke haben zB im Bankwesen (Marktanalysen, Derivativgeschäfte) oder auch in der computergestützten Chemie, Biochemie und Genetik Einzug gehalten. Dieses Praktikumsbeispiel soll die Neuronalen Netzen zugrundeliegende Theorie kurz präsentieren und anhand einer „Peaktrennung“ praktisch durchgeführt werden.

Es werden die Neuronen als auch der Backpropagation Algorithmus - in für das Beispiel relevanter Weise - diskutiert. Das Demonstrationsnetz mit Levenberg-Marquardt Algorithmus wird kurz erläutert.

Ein Artificial Neural Network (ANN) bildet einen Eingangsvektor ***p (Inputmatrix***, zB Absorptionen***)*** auf einen Ausgangsvektor ***a (Zielvektor oder Outputmatrix,*** zB vorausgesagte Konzentrationen***)*** ab. Ein ANN hat Gewichtsmatrizen ***(Weights) W***, Momente ***(Bias) b*** und Funktionsvektoren ***f (Transferfunktionen)***. Das ANN lernt durch Änderung der Gewichtsmatrizen. Die Transferfunktionen sind feste Größen, die Momente können variieren. Üblicherweise gibt man den Eingangsvektor und den Zielvektor vor und trainiert anschließend das Netzwerk. ***Das heißt man bestimmt die Gewichtsmatrizen und Momente nach einem Lernalgorithmus, um das berechntete Ergebnis (=Zielvektor a) an das wahre Ergebnis t anzunähern***.

Das populärste Lernverfahren ist Backpropagation, welches auch in unserem Beispiel als Lernalgorithmus eingesetzt wird. Mit diesem Algorithmus werden zu Beginn des Trainings zufällige W und b (siehe Abb. 1) bestimmt. Nach jedem Durchlaufen des Netzes wird das Ergebnis a mit dem Zielvektor t (zB die wahren Konzentrationen) verglichen und der resultierende Fehler e zur Abänderung der W und b verwendet. Dieser Vorgang wird solange wiederholt bis das gewünschte Trainingsziel erreicht wird.

e't&a Y zusammengefaßt zur Fehlermatrix E'T&A

Da sich die ANN auf Matrizen und Vektoren aufbauen, wird durch die Dimension des Eingangsvektors p der Raum festgelegt indem agiert werden kann. Die Matrizen müssen dermaßen bestimmt werden, daß den Eingangswerten in p die richtigen Ausgangswerte zugeordnet werden können. Die Transferfunktionen in Abb. 2 und 3 dienen dazu, die mögliche Lösungsmenge an die Größe der Ausgangswerte anzupassen.
Der Trick bei Neuronalen Netzwerken liegt nun darin jeden Eingangswert mit jedem Neuron zu verbinden. Man versucht also dem Netz ein gewisses „Muster“ der Eingangs- bzw Ausgangswerte anzulernen, indem beispielsweise das Netz die abnehmenden Absorptionswerte in p mit abnehmender Konzentration in a korreliert, wobei das Netz durch die Verknüpfung aller Werte diese Tendenz in alle Neuronen und damit in alle W und b einfließen läßt.
Man kann sich die kontinuierliche “Korrektur” der Weights W und Biases b auch als einen Vorgang wie in Abbildung 4 verdeutlichen. Auf einer 3-dimensionalen Fehlerfläche ist in x und y W und b aufgetragen; in Richtung Z wird die Größe des Fehlers dargestellt. Der Lernprozeß des ANN führt nun zu einer kontinuierlichen Annäherung an das globale Fehlerminimum - hierbei ist zu beachten, daß ein “Steckenbleiben” in einem lokalen Minimum verhindert wird.

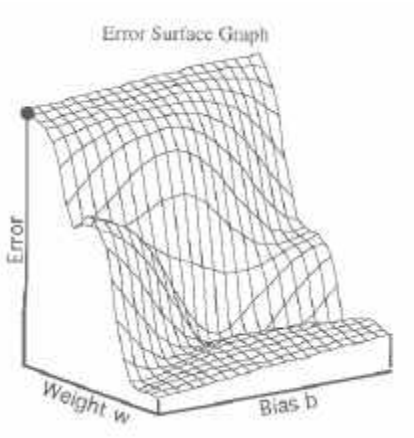

Abb 4 Fehlerfläche

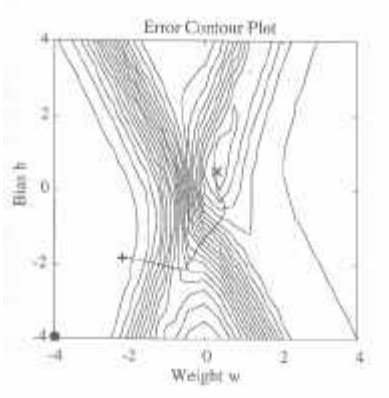

Abb 5 Höhenschichtlinien von Abb 4

Mögliche Ursachen für signifikante Diskrepanzen im trainierten Netz sind:

-**Entweder im ANN:** Eingegebene Datensatz ist nicht korrekt (daher immer eine Kontrolle ob die Trainingswerte überhaupt Sinn ergeben!); die Lernschritte sind zu groß und das Fehlerminimum wird übersprungen ...

-**Beim Backpropagation Verfahren:** Das ANN hat zB genügend "Erinnerungsmöglichkeiten" - mehr als doppelt so viele Neuronen als Eingangswerte werden verwendet. Die Transferfunktionen sind jedoch linear, deshalb schränken sie unsere Lösung nicht ein. Eine Änderung der Transferfunktionen könnte unsere Lösung besser einschränken oder die eingestellten Parameter für das Netz (Zahl der Neuronen, “Lerngeschwindigkeit”, ...) führen zu einem “übertrainieren” des ANN (Abb. 6/7).

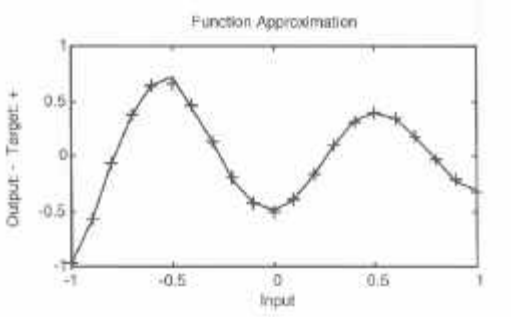

Abb 7 trainiertes Netz

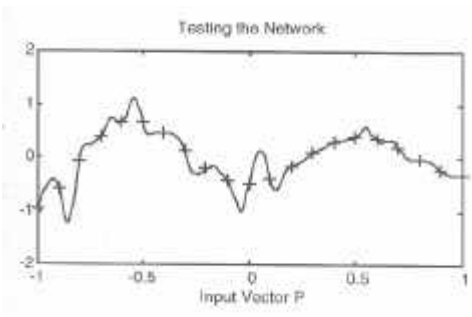

Abb 6 übertrainiertes Netz

In unserem Beispiel wird ein verbessertes Backpropagation Verfahren angewandt. Das verwendete Backpropagationverfahren ist von Levenberg-Marquardt (trainlm) und zählt zu den schnellsten Lernalgorithmen für W.

$$\ddot{A}W \text{ ' } \frac{1}{(J^T J \% \mu I)} J^T e$$

Die Schnelligkeit wird durch die Beurteilung des Fehlergradienten (Abb 4) erreicht. J ist eine Jacobi-Matrix aller Ableitungen jedes Fehlers bei jedem Weight. Der Abstieg auf der Fehlerfläche und damit die Größe der Änderung von ÄW wird durch die Größe des Skalars µ festgelegt: Bei einem großen µ erfolgt der Absteig in Richtung des Gradienten; bei kleinen Gradienten - nahe eines lokalen Minimum zB - wird µ kleiner und der Abstieg erfolgt in kleineren Schritten (genauer).

Konkret zu unserem Beispiel:

P ... ist die Inputmatrix mit den Absorptionen p_{xy} bei den vorgegebenen Wellenlängen y und bei den gemessenen Spektren x

T ... ist unsere Outputmatrix mit den bekannten Konzentrationen t_{xm} unserer Analyten Ponceau 4R und Amaranth in den jeweiligen Spektren x

Wenn wir mit unserem P das erste Mal das Netz durchlaufen bekommen wir Konzentrationen a_{xm} in A, die einen Fehler $e_{xm}=t_{xm}-a_{xm}$ gegenüber unseren wahren Werten t_{xm} aufweisen. Dieser Fehler wird nun verwendet, um unsere w_{xy} und b_{xy} nach oben beschriebenen Vorgang zu korrigieren. Mit diesen korrigierten W und B wird das Netz erneut gestartet und der - hoffentlich geringere - Fehler in den Konzentrationen erneut korrigiert und so fort.

Als Netzwerk dient uns eine einfaches Netzwerk mit:

Input Layer: Eingabe von P

Hidden Layer: Hier erfolgt die erste neuronale Verknüpfung mit w1 und b1, wobei als Transferfunktion ein ‘logsig’ verwendet wird um die Lösungsmenge einzuengen (vgl Abb. 2)

Output Layer: Eine zweite Neuronenschicht mit erneuten w2 und b2 - diesmal mit einer linearen Transferfunktion ‘purelin’; die Ergebnisse werden in A2 zusammengefaßt (vgl. Abb. 3); Zahl der Neuronen ist im Output Layer immer gleich der Anzahl der Ausgabewerte.

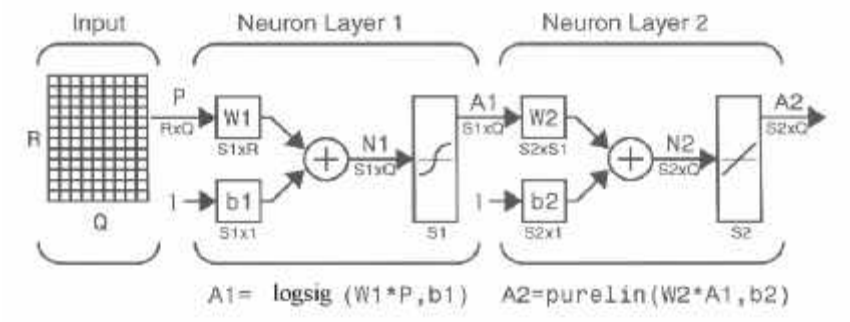

Abb 8 Unser verwendetes Netz; machen sie sich die Dimesionen der Matrizen bewusst!

Wichtigsten Punkte hinsichtlich des verwendeten Netzwerkes:

1. Verknüpfung von Information durch einfache Multiplikationen

Diese wird erreicht durch das multiplizieren der Inputmatrix (Absorption) mit den multiplikativen Gewichen (Weights). Damit werden einzelne gewichtete Absorptionswerte summiert - zu einem Skalar zusammengefaßt.

2. Nichtlinearität - nichtlineare Transferfunktionen

Durch das Einbinden von nichtlinearen Transferfunktionen (logsig) können in das Netz nichtlineare Rechengänge eingebunden werden. Die logsig Funktion beinhaltet - bildlich gesprochen - alle Steigungen die bei den Fehlerflächen auftreten (Abb 4).

3. „Lernen“ - kalibrieren und validieren

Die multiplikativen (Weights) und additiven Gewichte (Bias) sind anfänglich Zufallszahlen. Durch Vergleich der wahren Konzentration mit den errechneten wird dieser Fehler zur gezielten Abänderung der Gewichte verwendet (dem Algorithmus von Levenberg-Marquardt folgend - Gradienten der Fehlerfläche). Durch das Abbruchkriterium - welcher Vorraussagefehler soll unterschritten werden - kann man sich an das

globale Minimum der Fehlerfläche „herantasten“. Der Trainingszustand immer durch Validierung prüfen (dem Netz unbekannte Daten): schlecht, gut oder übertrainiert.

Literaturverweise

Francois E.Cellier: "Continuous System Modelling"
Hans Havlicek: "Lineare Algebra"
Howard Demuth:, Mark Beale: "Neural Network Toolbox"
Felix Breitenecker: "Unterlagen aus Regelungsmath..Modelle in der Medizin"
Möller, Popovic, Thiele: "Advances in Control Systems and Signal Processing"
...Unmengen an Homepages:
http://www.ajrhem.com/ann1.html
http://www.unet.univie.ac.at/~a9303347/nn/index.htm
http://www.neurocomputing.de/
http://web.media.mit.edu/~nitin/classes/adaptive/NNtalk/index.htm
http://www.informatik.uni-freiburg.de/~heinz/nnintro/