GROSSER BELEG

Indirekte Gewichtskorrekturverfahren für neuronale Netze

von
Lars Beikirch
geboren am 22. Juni 1973 in Dohna

Eingereicht am 09. April 1997 an der

Technischen Universität Dresden
Fakultät Informatik
Institut für Künstliche Intelligenz
Fachgruppe Wissensverarbeitung

TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Informatik

AUFGABENSTELLUNG FÜR DEN GROSSEN BELEG

Name, Vorname des Studenten:	Beikirch, Lars
Immatrikulationsnummer:	1129054

Thema: Indirekte Gewichtskorrekturverfahren für neuronale Netze

Zielstellung:

Zur Verringerung des Trainingsaufwandes von neuronalen Netzen kan man ein indirektes Gewichtskorrekturverfahren verwenden, bei dem in einem Zwischenschritt aus der Fehlerfunktion gewichtete Summen der Trainingsmuster gebildet werden [E. Koch: A Method for Design and Training Feedforward Neural Networks]. Stand und Anwendungsmglichkeiten dieses Verfahrens sind zu untersuchen. Außerdem ist die Möglichkeit der Integration dieses Lernverfahrens in den SNNS darzustellen.

Betreuender Hochschullehrer:	Prof. S. Hölldobler
Betreuer:	Dr. E. Koch
Institut:	Künstliche Intelligenz
Fachgebiet:	Wissensverarbeitung

Inhaltsverzeichnis

1 Einführung und Motivation **1**

2 Ein konstruktives Verfahren zur optischen Zeichenerkennung **6**

2.1 Einführung 6

2.2 Die Struktur eines Q-Neurons und des Netzes 7

2.3 Gewichtsberechnung 10

2.4 Das Lernverfahren 17

2.5 Implementation 19

2.6 Erste Tests 21

2.6.1 Erstes Beispiel - einfache Ziffern 21

2.6.2 Zweites Beispiel - handgeschriebene Ziffern aus Postleitzahlen 27

2.7 Vergleich mit Backpropagation 31

2.8 Zusammenfassung 32

3 Gradientenabstiegsverfahren **34**

3.1 Untersuchung der originalen Korrekturformeln 34

3.2 Gradientenabstieg 36

3.2.1 Grundprinzip von Gradientenverfahren 36

3.2.2 Herleitung der Wichtungskorrektur 38

3.3 Implementation 43

4 Differenzen-Neuron **44**

4.1 Struktur des Differenzen-Neurons 44

4.2 Gradientenabstiegsverfahren für das D-Neuron 45

4.3 Implementation 46

5 Anlernen eines Netzwerkes **47**

5.1 Einführung und Zielsetzung 47

5.2 Lösungsansatz 47

5.3 Implementation und Tests 54

5.3.1 Unterscheidung von zwei Klassen 55

5.3.2 Unterscheidung von 4 Klassen 57

5.3.3 Unterscheiden aller Ziffern 59

5.3.4 Vergleich mit Backpropagation 63

5.4 Zusammenfassung 64

6 Integration in den SNNS **68**

6.1 Einführung in den SNNS 68

6.2 Interne Datenstrukturen 69

6.3 Implementation von Q-Neuronen im SNNS 71

7 Zusammenfassung und Ausblick **72**

A Lernkurven **74**

B Erstes Beispiel - Ziffernprototypen **109**

C Zweites Beispiel - Ziffern aus Postleitzahlen **110**

Literatur **120**

1 Einführung und Motivation

Künstliche neuronale Netze[1] haben in der vergangenen Zeit in weiten Bereichen der Wissenschaft und Technik Einzug gehalten. Ihr Einsatz hilft in vielen Fällen, Probleme zu lösen, deren exakte mathematische Beschreibung und Lösung unmöglich ist oder zumindest extreme Schwierigkeiten bereitet und daher nicht sinnvoll erscheint. Neuronale Netze werden mit Hilfe von Beispieldaten angelernt, sie sollen danach in der Lage sein, das gegebene Problem zu lösen. Die dazu eingesetzten Lernverfahren und natürlich auch die Strukturen und Funktionsprinzipien der Netze variieren zum Teil erheblich. Dabei spielt die Art der Aufgabe, die das Netz erfüllen soll, eine sehr wesentliche Rolle. So legen zum Beispiel die Ein- und Ausgabedaten durch die Form ihrer Repräsentation die Anzahl der Neuronen in der Ein- und Ausgabeschicht fest. Die Struktur der Verbindungen zwischen ihnen und das Vorhandensein von eventuell mehreren Zwischenschichten sowie deren Dimensionen wird zumeist durch die Anforderungen an das Verhalten des Netzes bestimmt. Dabei spielen Erfahrungswerte eine sehr wesentliche Rolle. Es ist z.B. allgemein bekannt, daß große Netze mit vielen freien Parametern dazu neigen, sich zu stark an die Lernmenge anzupassen und somit schlecht zu generalisieren.

Ein sehr wichtiges Anwendungsgebiet der neuronalen Netze ist die Musterverarbeitung sowie eines ihrer Teilgebiete, die Schrifterkennung. Für die Verarbeitung von Schrift werden zwei Ansätze verfolgt: die wortorientierte und die zeichenorientierte Erkennung [GuSu94]. Keinem von beiden kann prinzipiell der Vorzug gegeben werden. Der wortorientierte Ansatz vermeidet das schwierige Problem der Segmentierung von geschriebener Schrift, ist jedoch nur in begrenzten Domänen sinnvoll einsetzbar. Der zeichenorientierte Ansatz wird in der Praxis wesentlich häufiger verwendet. Dies liegt nicht zuletzt an den hauptsächlichen Einsatzgebieten: Postleitzahlenerkennung, Bankscheckverarbeitung, Formularverarbeitung, Erkennen von aufgedruckten Bezeichnungen auf Bauteilen usw. [Kov95]. Diese Aufgabenstellungen bieten meist die Möglichkeit, die einzelnen Zeichen gut zu separieren, was den Einsatz von Einzelzeichenerkennern erlaubt.

Um die Leistungsfähigkeit der Erkenner zu steigern, werden die zu erkenneden Zeichen einer Vorverarbeitung unterzogen. Die Skalierung auf eine vorgegebene Größe (nicht immer unter Beibehaltung des Breite/Höhe-Verhältnisses) stellt dabei eine Art Minimumstandard dar. Darüber hinaus werden

[1] Im Folgenden seien mit neuronalen Netzen oder auch einfach Netzen immer künstliche neuronale Netze gemeint, so es nicht anders angegeben ist.

oft folgende Korrekturverfahren verwendet:

- Neigungskorrektur ([GuSu95], [LeCun90], [Holm97])
- Reparatur von unterbrochenen Linien ([GuSu95])
- Skalierung der einzelnen Bildpunkte (Grauwerte) ([LeCun90])
- Normierung der Strichstärke bzw. Skelettierung ([Holm97])

Die Klassifikatoren werden nicht immer mit den Mustern direkt angelernt. In vielen Verfahren operieren die neuronalen Netze auf Merkmalen, welche zuvor aus den gegebenen Mustern extrahiert wurden. Die Musterextraktion selbst wird manuell z.B. durch die Angabe von Filtern gesteuert. Das am häufigsten genutzte Merkmal ist die Kantenorientierung. Dazu werden spezielle Filtermasken per Hand definiert, welche auf horizontale, vertikale und diagonale Linien reagieren. Das Originalmuster wird mit diesen Filtern abgetastet. Dabei ermittelt man für jeden Punkt die Ausgabe des Filters ([Cho97]) oder faßt mehrere benachbarte Punkte zusammen ([Kov95]). Es gibt jedoch auch Netzstrukturen, welche auf Grund ihres Aufbaus die Merkmalsextraktion selbständig ausführen [LeCun90], [Rid96], [LoTs97].

Eine Alternative zur Merkmalsextraktion stellt die Verwendung von Transformationen der Muster dar. Diese Transformationen bieten zum Teil bereits durch ihre Definition eine Invarianz gegenüber verschiedenen Deformationen, wie z.B. Translation, Skalierung oder Rotation. Als Beispiele seien die Fouriertransformation oder die Houghtransformation genannt.

Für die klassifizierenden neuronalen Netze wurden verschiedene Netzstrukturen untersucht. Als sehr erfolgreich haben sich dabei insbesondere Multilayer Perzeptrons (MLP), Selbstorganisierende Karten (SOM) und Lernende Vektorquantisierung (LVQ) und Radial Basis Function (RBF) Netze erwiesen.

- **MLP:** MLP's mit vielen freien Parametern (speziell voll vernetzte) haben insbesondere bei hoher Dimension der Eingabedaten eine sehr schlechte Generalisierungsfähigkeit ([LeCun90], [ChSa95]). Einer der bekanntesten Ansätze zur Verringerung der Anzahl der freien Parameter ist das von Le Cun vorgestellte Prinzip der gemeinsam benutzten Gewichte [LeCun90]. Die Neuronen in höheren Schichten fassen die Ausgaben von kleinen Feldern der darunter liegenden Schicht in einer Merkmalskarte unter Bewahrung der topologischen Ordnung zusammen. Dabei haben die Neuronen, welchen zusammen eine solche Karte bilden, ein und den selben Gewichtsvektor. Sie stellen eine Art Merkmalsextraktor dar. Allerdings haben nach [Rid96] diese Merkmale nur wenig Gemeinsamkeiten mit den von Menschen in der manuellen Merkmalsextraktion benutzten Filtern. Das von Le Cun vorgestellte Netz (in der Literatur auch manchmal als LeNet bezeichnet) bildet in einer ersten Stufen aus dem Eingabebild von 28x28 Punkten mit einem rezeptiven Feld von 5x5 Punkten vier Merkmalskarten der Größe 24x24. Diese werden mit einem 2x2-Mittelwertbilder verkleinert. Aus diesen werden nach einem speziellen Konnektionsschema wieder mit einem rezeptiven Feld von 5x5 Punkten insgesamt 12 Karten der Größe 8x8 gebildet, welche wiederum mit einem 2x2-Mittelwertbilder verkleinert werden. Die so entstehenden 12 4x4-Karten sind mit der Ausgabschicht von 10 Neuronen voll vernetzt. Das Netz wurde mit einem Backpropagatation-Verfahren zweiter Ordnung angelernt.

- **SOM und LVQ:** Die Selbstorganisierenden Karten nach Kohonen [Koh78] operierend grundsätzlich auf dem Abstand des präsentierten Musters von Prototypen bezüglich einer gegebenen Metrik (meist euklidischer Abstand). Das Muster wird mit dem Prototypen identifiziert, welchem es am nächsten liegt. Bei LVQ-Netzen wird durch überwachtes Lernen auf die sich ausbildende Struktur Einflußgenommen. SOM's haben den Nachteil, daß sie bei steigenden Genauigkeitsanforderungen schnell so groß werden, daß ihre Berechnung sehr viel Zeit in Anspruch nimmt. Als Möglichkeit, dies zu umgehen, wurde von Wu und Yan in [WuYa95] die Verwendung von zweistufigen Karten vorgeschlagen. Die erste Stufe hat nur eine sehr grobe Auflösung und dient zur Auswahl einer Karte aus der zweiten Stufe. Diese besitzt eine wesentlich höhere Auflösung, ist jedoch, da sie einen geringeren Datenbereich abdecken muß, wesentlich kleiner. Auf diese Art wird ein erheblicher Geschwindigkeitsgewinn erreicht. Eine etwas andere Variante der auflösungsverbesserung in kritischen Bereichen wird in Cho97 angegeben und dort als adaptive SOM's bezeichnet. Hier werden SOM-Neuronen, welche auf Eingaben von verschiedenen Klassen reagieren, durch mehrere Neuronen ersetzt.

- **RBF:** RBF-Neuronen bestimmen ihre Ausgabe über eine radialsymmetrische Basisfunktion aus dem Abstand des angelegten Musters vom Gewichtsvektor [Zell94]. Nach [FoNe95] haben sie in der Mustererkennung eine ähnliche Leistungsfähigkeit wie MLP's, jedoch meist mit kürzeren Trainingszeiten.

Ein in vielen Bereichen der Wissenschaft und Technik bekanntes und benutztes Grundprinzip ist, daß mehrere kleine Teillösungen eine bessere große Lösung ergeben. Auch dieses Prinzip hat in der Mustererkennung Anwendung gefunden. Es werden mehrere Klassifikatoren eingesetzt, welche vom gleichen Typ, z.B. mehrere MLP's - [Cho97] oder von verschiedenen Typen, z.B. SOM's und MLP's sein können. Eine weitere Möglichkeit besteht darin, die einzelnen Klassifikatoren auf unterschiedlichen Eingangsdaten arbeiten zu lassen. So werden z.B. in [WuYa95] ein Klassifikator auf Pixelbasis und ein Klassifikator auf Kantenbasis oder in [Kov95] drei Klassifiaktoren auf der Basis von Paaren aus der Abstandstransformation, Richtungsinformationen und Kurvenpunkten eingesetzt. Prinzipiell kann man der Verwendung von mehreren Klassifikatoren zwei Grundvarianten unterscheiden:

- **Paralleler Einsatz:** Mehrere Klassifikatoren bearbeiten das zu klassifizierende Muster parallel. Die Ergebnisse der einzelnen Klassifikatoren werden zusammengefaßt. Dies kann z.B. durch ein (Multilayer) Perzeptron geschehen [Kov95]. Es werden auch Abstimmungstechniken eingesetzt. Besonders in diesen findet die Zuverlässigkeit der einzelnen Klassifikatoren Beachtung. Auch Fuzzy Logik wurde bereits erfolgreich eingesetzt [Cho97].

- **Hierarchischer Einsatz:** Das zu klassifizierende Muster wird zuerste einem einfachen und schnellen, dafür jedoch nicht besonders zuverlässigen Klassifikator präsentiert. Wenn dieser kein eindeutiges Ergebnis liefert, wird ein zweiter langsamerer, aber dafür sicherer Klassifikator eingesetzt, um das Ergebnis zu bestimmen [ChWuYa94].

Die Simulation von neuronalen Netzen ist eine sehr weit verbreitete und flexible Technik. Die einzelnen Neuronen stellen kleinste Verarbeitungseinheiten dar, welche jede für sich nur eine sehr geringe Leistungsfähigkeit haben. Da bei vielen Anwendungen zumindest ein Großteil der Neuronen gleichzeitig seine Ausgaben berechnen kann, gibt es sehr intensieve Bestrebungen, sie parallel zu implementieren. Die Optik bietet dafür durch die räumliche Ausbreitung des Lichtes nahezu ideale Voraussetzungen. Ein Nachteil entsteht jedoch durch die Ungenauigkeit der technischen Systeme. Trotzdem wurden bereits sehr erfolgreiche optische Anwendungen realisiert, welche sich besonders durch eine hohe Verarbeitungsgeschwindigkeit auszeichnen. In [FoNe95] wird zum Beispiel der Aufbau eines Musterklassifikators auf der Basis eines RBF-Netzes beschrieben.

Zwei sehr wesentliche Probleme beim Anlernen von Musterklassifikatoren sind die hohe Dimensionalität des Eingaberaumes und die hohe Kardinalität der Lernmenge. In vielen Arbeiten wird insbesondere das letztere Problem durch den Einsatz von Clusterungsalgorithmen, wie z.B. k-Means oder von Selbstorganisierenden Karten zur Bildung von wenigen Prototypen zu umgehen versucht [FoNe95], [ChWuYa94]. Auch ein Einsatz der von Fritzke vorgeschlagenen Growing Cell Structures [Fri93] wäre denkbar.

In [Koch96] werden das Q-Neuronen (Quotienten-Neuron) und das dazugehörende IWD-Lernverfahren (Indirect Weight Determination) vorgestellt, welche ebenfalls den Lernaufwand erheblich reduzieren sollen. Eine Besonderheit ist, daß die Gewichte der Q-Neuronen durch Linearkombinationen von ausgewählten Mustern der Lernmenge gebildet werden. Dies hat auch eine wesentliche Einschränkung der zu adaptierenden Parameter zur Folge.

In dieser Arbeit wird das von Koch vorgestellte Neuron und das Lernverfahren detailliert untersucht und zusätzlich ein auf dem Gradientenabstieg basierendes Lernverfahren vorgeschlagen. Des weiteren wird eine Möglichkeit beschrieben, wie auf der Basis von Q-Neuronen ein neuronales Netz angelernt werden kann, ohne durch a-priori-Wissen die genaue Struktur des Netzes konstruieren zu müssen. Auch dieses Lernverfahren wird detailliert untersucht.

2 Ein konstruktives Verfahren zur optischen Zeichenerkennung

2.1 Einführung

Das hier beschriebene Verfahren zum konstruktiven Aufbau eines neuronalen Netzes wurde von Koch aus einer technischen Anwendung, in welcher mit optischen Filtern Zeichenerkennung realisiert werden konnte [Koch81], entwickelt. Aus heutiger Sicht kann man die damals entstandene Lösung als ein künstliches neuronales Netz interpretieren. Dabei war das Problem der Filterberechnung für das optische System mit dem Problem des Lernens bei neuronalen Netzen vergleichbar.

Traditionelle neuronale Netze, wie zum Beispiel des Multilayer-Perzeptron haben einige entscheidende (und wohlbekannte) Nachteile:

- Die Netze weisen oft schlechte Konvergenz beim Training auf.
- Die Energiefunktion kann lokale Minima besitzen, so daß das globale Minimum unter Umständen nicht gefunden wird, weil das Lernen in lokalen Minima steckenbleibt.
- Bei zu kleiner Netzstruktur kann das Problem möglicherweise nicht gelernt werden.
- Eine zu großzügige Netzstruktur führt oftmals zu Overfitting und damit mangelnder Generalisierungsfähigkeit.

Ein Teil dieser Probleme konnte durch veränderte Lernverfahren und/oder Netzstrukturen verringert oder gar beseitigt werden. Hierzu seien als Beispiele die folgenden Möglichkeiten genannt:

- adaptive Lernraten
- Momentumterm
- Einbeziehung höherer Ableitungen (z.B. konjugierter Gradient)
- Hinzufügen bzw. Entfernen von Neuronen, Konstruktion eines Netzes

Die zuletzt genannten konstruktiven Verfahren lösen die Aufgabe meist schrittweise, indem Teilnetze mit Teilaufgaben trainiert werden, wobei die einfachen Aufgaben zuerst gelöst werden. In dem hier vorgestellten Verfahren soll dies jedoch etwas anders gehandhabt werden.

2.2 Die Struktur eines Q-Neurons und des Netzes

In einem Perzeptron wird aus der am Neuron anliegenden Eingabe $\vec{x}$ mit dem Gewichtsvektor $\vec{w}$ des Neurons dessen Aktivierung $net(\vec{x}) = \vec{x} \cdot \vec{w}$ berechnet. Aus ihr ergibt sich über die Ausgabefunktion die Ausgabe des Neurons als $y = f_{out}(net(\vec{x}))$. Als Ausgabefunktion werden meist die Sigmoide $f_{out}(x) = \frac{1}{1+e^{-x}}$ oder der Tangens Hyperbolicus $f_{out}(x) = \tanh(x)$ verwendet.

Ziel ist es, mit dem Neuron eine Klassifikation der Eingabe durchzuführen. In dem hier betrachteten Verfahren soll ein einzelnes Neuron das gerade angelegte Eingabemuster in eine von zwei Mustermengen einordnen. Diese Mengen können aus mehreren Klassen bestehen und sind für jedes Neuron verschieden. Das Modell des Neurons wird dazu um ein zweites Gewicht erweitert. Somit können nun zwei Aktivierungen $net^{(1)}(\vec{x}) = \vec{x} \cdot \vec{w}^{(1)}$ und $net^{(2)}(\vec{x}) = \vec{x} \cdot \vec{w}^{(2)}$ berechnet werden. Der Quotient der Aktivierungen dient dann als Argument für die Ausgabefunktion. In [Koch81] wurde die Bildung des Quotienten der Bildung der Differenz in der Lernphase vorgezogen[2], da so der Intensitätsunterschied von Mustern (z.B. ein Großbuchstabe und ein Dezimalpunkt) besser ausgeglichen werden konnte. Während des Lernens wird der Quotient jedoch in Abhängigkeit von der gewünschte Mustermenge gebildet. Ist das Muster $\vec{x_i}$ Element der ersten Menge, so ergebe sich der Quotient als [3]

$$Q_i = \frac{net_i^{(2)}(\vec{x_i})}{net_i^{(1)}(\vec{x_i})} \quad , \tag{1}$$

anderenfalls sei der Quotient für $\vec{x_j}$ bestimmt durch

$$Q_j = \frac{net_j^{(1)}(\vec{x_j})}{net_j^{(2)}(\vec{x_j})} \quad . \tag{2}$$

Diese Definition ermöglicht es, das Lernverfahren unabhängig von der Mustermenge des angelegten Musters zu beschreiben. Ziel des Lernverfahrens ist

[2] In der Anwendung wurde mit der Differenz gearbeitet, da sich diese optisch wesentlich einfacher handhaben ließ.

[3] Es wird folgende Konvention eingeführt :

- ein Index i bezeichne einen entsprechenden Wert, wenn das zugehörige $\vec{x_i}$ Element der ersten Mustermenge ist
- ein Index j bezeichne einen entsprechenden Wert, wenn das zugehörige $\vec{x_j}$ Element der zweiten Mustermenge ist
- ein Index k bezeichne einen entsprechenden Wert, wobei das zugehörige $\vec{x_k}$ Element der ersten oder der zweiten Mustermenge sein kann

es, die Gewichte so einzustellen, daß die Quotienten kleiner oder nahe einem vorgegebenen Wert $Q^{(S)}$ sind:

$$Q_k \lessapprox Q^{(S)} \tag{3}$$

Die Unterscheidung zwischen Mustern der ersten und der zweiten Menge ist jedoch nur während der Lernphase sinnvoll, außerhalb dieser sei der Quotient definiert als

$$Q = \frac{net^{(2)}(\vec{x})}{net^{(1)}(\vec{x})}. \tag{4}$$

An die Aktivierungsfunktion werden keine besonderen Ansprüche gestellt. Die Ausgabe des Neurons berechnet sich als $y = f_{out}(Q)$. Es sei angemerkt, daß bei Verwendung einer binären Schwellwertfunktion der Quotient auch durch die Differenz ersetzt werden kann.

Wie an der Gleichung (4) zu erkennen ist, kann ein solches Neuron nicht in der Lage sein, zwei Muster $\vec{x}$ und $c \cdot \vec{x}$ zu unterscheiden, da sich der Faktor bei der Quotientenbildung aufhebt. Um eine solche Situation auszuschließen, werden an die anzulernenden Muster normiert, so daß sie der Forderung

$$\sum_n x_n^2 = 1 \tag{5}$$

genügen. Hier erweist es sich als Nachteil, daß der Quotientenbilden der Vorzug vor der Differenzbildung gegeben wurde, denn ein Neuron, welches die Differenz bildet, könnte diese beiden Muster unterscheiden.

An dieser Stelle seien noch zwei grundsätzliche Bemerkungen angebracht:

Einordnung in ein formales Modell eines Neurons

Unter Verwendung einer völlig unüblichen Netzeingabe- oder auch Propagierungsfunktion kann das Q-Neuron auch in ein formales Modell von neuronalen Netzen, wie es z.B. in [NaKlKr94] vorgestellt wird, eingeordnet werden. Diese Netzeingabefunktion wäre dann zu definieren als:

$$net = \frac{\vec{w}^{(2)} \cdot \vec{x}}{\vec{w}^{(1)} \cdot \vec{x}} \tag{6}$$

was von der üblichen gewichteten Summe der Eingänge erheblich abweicht. Als Aktivierungs- und Ausgabefunktion ist die Linearität zu verwenden.

Klassifikationsfähigkeit

Von einfachen Perzeptrons ist bekannt, daß sie ihren n-dimensionalen Eingaberaum durch eine (n-1)-dimensionale Hyperebene in zwei Teile zerlegen (lineare Separabilität, [NaKlKr94] u.a.). Diese Eigenschaft trifft auch auf das Q-Neuron zu. Betrachtet man als Ausgabefunktion eine Schwellwertfunktion mit dem Schwellwert θ, so ergibt sich z.B. die Klassifikation als Klasse 1 genau dann, wenn:

$$\theta \leq \frac{\vec{w}^{(2)} \cdot \vec{x}}{\vec{w}^{(1)} \cdot \vec{x}} \tag{7}$$

und somit

$$0 \leq \left(\vec{w}^{(2)} - \theta \vec{w}^{(1)}\right) \cdot \vec{x} \quad , \tag{8}$$

was als lineare Separierung zu verstehen ist. Es gibt dabei zwei Besonderheiten: 1. Die trennende Hyperebene geht durch den Nullpunkt. 2. Das Abstandsmaßvon der Trennebene unterscheidet sich von dem des Perzeptrons durch seine Nichtlinearität und Asymmetrie bezüglich des euklidischen Abstands.

Auf der Basis von Q-Neuronen kann man ein neuronales Netz konstruieren, welches aus eine verdeckten Schicht von Q-Neuronen und einer Ausgabeschicht von Schwellwerteinheiten besteht. Für jedes Q-Neuron müssen zu Beginn die beiden zu unterscheidenden Mustermengen festgelegt werden. Dabei ist es nicht notwendig, für jedes Q-Neuron festzulegen, wie es ein bestimmtes Muster klassifizieren soll. Es muß jedoch sichergestellt werden, daß die Aufgabe insgesamt lösbar ist. Sind alle Q-Neuronen angelernt, so werden die Gewichte zu den Neuronen der Ausgabeschicht bestimmt. Das Prinzip soll an dem folgenden Beispiel verdeutlicht werden:

In einem neuronalen Netz, welches Muster als Ziffern klassifizieren soll, trennt das verdeckte Neuron H_1 die Muster der Klassen 1, 4 und 7 von den Muster der Klassen 0, 3 und 8. Das Neuron H_2 unterscheidet die Muster der Klassen 0, 6 und 8 von den Mustern der Klassen 3 und 9. Für alle anderen Neuronen der verdeckten Schicht sei die Ausgabe bei einem angelegten Muster der Klasse 3 unbestimmt. Die Q-Neuronen liefern genau dann eine Ausgabe von -1, wenn das angelegte Muster als zur ersten Menge zugehörig erkannt wurde, sonst liefern sie eine 1. Dann sind die Gewichte des Ausgabeneurons O_3, welches genau dann eine Ausgabe von 1 liefern soll, wenn ein Muster der Klasse 3 angelegt wurde, so zu wählen, daß sein Schwellwert genau dann überschritten wird, wenn die Neuronen H_1 und H_2 beiden eine Ausgabe größer 1 liefern. Die Abbildung 1 verdeutlicht dieses noch einmal.

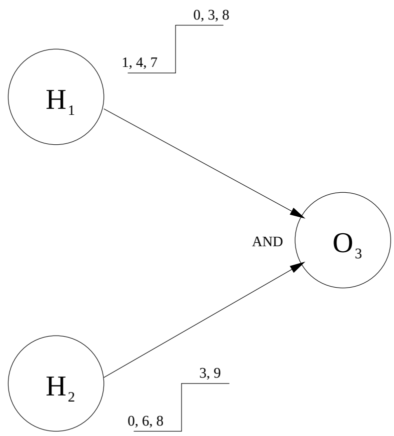

Abbildung 1: *Fragment aus einem konstruierten Netzes mit Q-Neuronen*

2.3 Gewichtsberechnung

Die Gewichtsberechnung ist ein iteratives Verfahren, bei dem die Gewichte über direkte Gleichungen aus (neu einzuführenden) Hilfsparametern berechnet werden. Dazu werden aus den eigentlichen Trainingsmustern zwei neue Eingaben gebildet, die jeweils einen gewichteten Mittelwert aller Muster der jeweiligen Mustermenge darstellen :

$$\vec{x}^{(1)} = \frac{1}{K_1} \sum_i g_i \vec{x}_i, \tag{9}$$

$$\vec{x}^{(2)} = \frac{1}{K_2} \sum_j g_j \vec{x}_j, \tag{10}$$

wobei K_1 und K_2 die Anzahl der Muster in der ersten bzw. zweiten Mustermenge seien. Die g_k dienen als Hilfsgewichte, aus denen sich die Gewichte direkt ableiten lassen. Speziell bei einem hochdimensionalem Eingaberaum wird dadurch eine erhebliche Dimensionsreduktion des Suchproblems erreicht, wenn die Anzahl der Trainingsmuster wesentlich kleiner als die Dimension der Eingabevektoren ist (und das ist eine durchaus realistische Annahme).

Aus den beiden Hilfseingabevektoren wird ein Quotientenvektor $\vec{q}$ gebildet, wobei seine Elemente q_n definiert sind als :

$$q_n = \frac{x_n^{(2)}}{x_n^{(1)}}. \tag{11}$$

Die Elemente q_n haben kleine Werte, wenn die entsprechenden Elemente der Eingabevektoren der ersten Mustermenge groß und die der zweiten klein sind, und umgekehrt. Um große Aktivierungen $net_i^{(1)}$ und kleine $net_j^{(1)}$ zu erhalten, sind die Gewichtskomponenten $w_n^{(1)}$ jeweils als monoton fallende Funktion der entsprechenden Elemente des Quotientenvektors $\vec{q}$ zu wählen. Dementsprechend müssen die $w_n^{(2)}$ als monoton fallende Funktion gewählt werden, um kleine Aktivierungen $net_i^{(2)}$ und große $net_j^{(2)}$ zu erhalten. Mit einem weiteren Parameter Θ können die eben genannten Beziehungen ausgedrückt werden als :

$$w_n^{(1)} = W(\Theta - q_n), \tag{12}$$

$$w_n^{(2)} = W(\Theta - \frac{1}{q_n}), \tag{13}$$

wobei W eine monoton wachsende Funktion darstellt.

Von Weymaere und Martens wird in [WeMa94] eine Vorgehensweise vorgeschlagen, welche der soeben dargestellten ähnelt. Sie verwenden einen Clusterungsalgorithmus (modifiziertes *k-means*) um im Eingaberaum Clusterzentren zu erzeugen, welche Prototypen von Mustern entsprechen. Diese Clusterzentren sind die Schwerpunkte einer Menge von Mustern, Abbildung 2 verdeutlicht dies. Aus diesen Clusterzentren wählen sie unter bestimmten Gesichtspunkten Paare aus, welche dann einen Gewichtsvektor für ein Perzeptron bilden. Die Trennebene, die ein solches Perzeptron in den Raum legt, entspricht der Ebene, welche senkrecht zu der Geraden durch die beiden Clusterzentren genau in der Mitte zwischen den selben liegt. Wie man in der Abbildung am Muster 2-4 sieht, kann es passieren, daß ein solches Perzeptron ein Muster der falschen Klasse zuordnet. Deshalb verwenden Weymaere und Martens dieses Perzeptron als Grundlage für ein weiteres Lernen mit Backpropagation.

In dem hier betrachteten Verfahren können durch die Wichtungen für jedes Muster nach den Gleichungen (9) und (10) gewichtete Schwerpunkte gebildet werden, so daß ein analog konstruiertes Q-Neuron (oder auch ein Perzeptron) die gegebenen Muster korrekt klassifiziert. Man vergleiche hierzu Abbildung 3.

Die Gewichtsberechnung erfolgt also über eine feste Funktion aus den g_k und dem Θ. Das Lernverfahren kann auf diesen Zusammenhang aufgebaut werden. Es wird nicht mehr direkt dazu eingesetzt, die Gewichte zu berechnen, sondern um die g_k und Θ zu korrigieren.

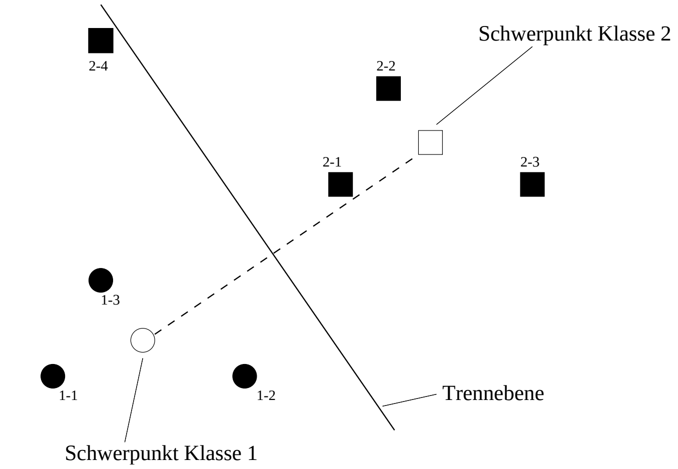

Abbildung 2: *Schwerpunkte nach k-means-Clusterung und die daraus resultierende Trennebene*

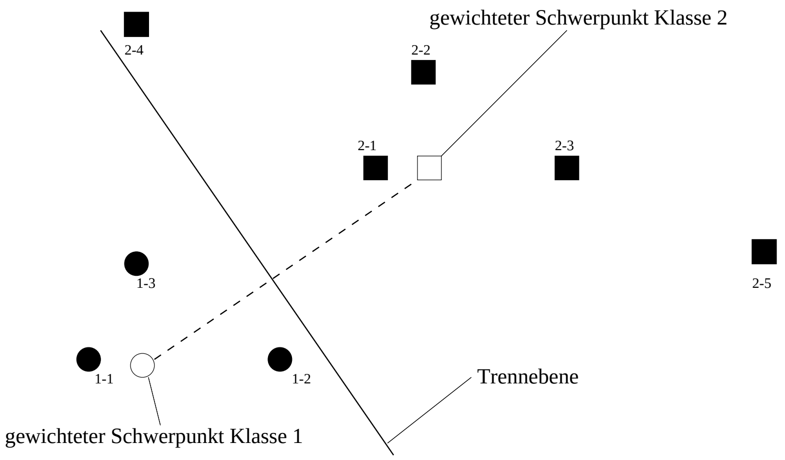

Abbildung 3: *Gewichtete Schwerpunkte und die daraus resultierende Trennebene*

In [Koch96] wurde als Grundlage für die Korrekturformeln angegeben, daß der Mittelwert der Quotienten Q_k am stärksten von Θ und die einzelnen Quotienten Q_k am stärksten von den zu diesem Muster gehörenden Wichtungen g_k abhängen würden. Diese Aussage erweist sich jedoch bei Betrachtung der komplexen Zusammenhänge bei der Gewichtsberechnung als nicht allgemeingültig. Nach Rücksprache mit Koch wurde als Begründung für die Parameteränderung folgende Schlußfolgerungskette angegeben:

Annahme : Der Wert für Q_i ist größer, als erwünscht (man beachte, daß hier das Muster Element der ersten Musterklasse ist). Wie kann eine Verringerung von Q_k erreicht werden? Bei Untersuchung der Gleichung (1) stellt man fest, daß dies durch eine Vergrößerung von $\vec{x_k} \cdot \vec{w}^{(1)}$ und Verkleinerung von $\vec{x_k} \cdot \vec{w}^{(2)}$, also durch eine Vergrößerung von $\vec{w}^{(1)}$ und eine Verkleinerung von $\vec{w}^{(2)}$ erreicht werden kann. Unter Beachtung der Abhängigkeiten in (12) und (13) erkennt man, daß dafür eine Verringerung der zugehörigen q_n notwendig ist. Dies ist nach (11) offensichtlich durch Erhöhen der Werte $x_n^{(1)}$ erreichbar. Da alle Elemente in $\vec{x_k}$ nicht negativ sind, kann man genau dieses Ziel durch Vergößern des zugehörigen Hilfsgewichtes g_k realisieren.

Gleiches gilt für ein zu großes Q_j zu einem Muster aus der zweiten Mustermenge, da die Definition des Quotienten hier reziprok ist. Auf Grund analoger Überlegungen erkennt man, daß bei einem zu kleinem Q_k das dazugehörige g_k zu verkleinern ist.

Für den Einfluß des Parameters Θ wurde nach ähnlichen Gesichtspunkten angenommen, daß eine Erhöhung dieses Wertes eine Verringerung der Quotienten und eine Verkleinerung von Θ eine Vergrößerung der Quotienten zur Folge hat.[4]

In [Koch96] werden auf der Grundlage dieser Zusammenhänge schrittweise Korrekturformeln entwickelt. Die veränderten Parameter nach einem Lernschritt ergeben sich durch

$$\begin{aligned} \Theta(t) &= \Theta(t-1) + \Delta\Theta(\bar{Q}, t), &(14)\\ g_k(t) &= g_k(t-1) + \Delta g_k(Q_k, t), &(15) \end{aligned}$$

wobei

$$\bar{Q} = \frac{1}{K}\sum_k Q_k. \tag{16}$$

[4] Wenn der aufmerksame Leser einwenden möchte, daß dies nicht korrekt sei, so sei er in seiner Vermutung bestätigt und auf das Kapitel 3.1 verwiesen.

Die Abweichungen der Quotienten der einzelnen Muster vom gewünschten Quotienten $Q^{(S)}$ werden in die Werte $\Delta\bar{Q}$ und ΔQ_k umgesetzt, welche definiert sind als:

$$
\begin{aligned}
\Delta\bar{Q} &= Q^{(S)} - \bar{Q} && (17)\\
\Delta Q_k &= Q_k - \bar{Q}. && (18)
\end{aligned}
$$

Dies bedeutet, daß die Korrekturformeln versuchen, den Mittelwert $\bar{Q}$ der Quotienten Q_k gegen $Q^{(s)}$ und die einzelen Quotienten gegen den Mittelwert konvergieren zu lassen.

Als eine erste recht einfache Form der Änderungsangabe wird

$$
\begin{aligned}
\Delta\Theta^{(1)} &= \tau_1^{(1)} \cdot \arctan(\tau_2^{(1)} \Delta\bar{Q}(t-1)), && (19)\\
\Delta g_k^{(1)} &= \gamma^{(1)} \cdot \Delta Q_k(t-1), && (20)
\end{aligned}
$$

eingeführt. Dabei sollen $\tau_1^{(\nu)}, \tau_2^{(\nu)}, \gamma^{(\nu)}$ sowie $\Delta\Theta^{(\nu)}$ und $\Delta g_k^{(\nu)}$ Werte repräsentieren, die zu Formeln der Stufe ν gehören. Die Formeln (19) und (20) weisen eine entfernte Ähnlichkeit mit der Deltaregel auf, da der auftretende Fehler ($\Delta\bar{Q}$ bzw. ΔQ_k), als Grundlage für die Korrektur verwendet wird.

Jedoch ist auf ihrer Grundlage keine sehr gute Konvergenz zu erwarten, ja es ist nicht einmal sicher, daß das Verfahren überhaupt konvergiert. In [Koch96] wird als ein Ansatz zur Behebung dieses Problems vorgeschlagen, die Fehler und Änderungen aus dem vorherigen Korrekturschritt in die Korrekturformel mit einzubeziehen:

$$
\begin{aligned}
\Delta\Theta^{(2)} &= (\Theta(t-1)-\Theta(t-2))\cdot\Big\{\tau^{(2)}\cdot\arctan\Big(\frac{\Delta\bar{Q}(t-1)}{\tau^{(2)}\cdot\big(\Delta\bar{Q}(t-1)-\Delta\bar{Q}(t-2)\big)}\Big)\\
&\quad +\tfrac{\pi}{2}\tau^{(2)}\cdot\Big(1-\mathrm{sgn}\Big((\Theta(t-1)-\Theta(t-2))\cdot\Big(\Delta\bar{Q}(t-1)-\Delta\bar{Q}(t-2)\Big)\Big)\Big)\Big\}, && (21)\\
\Delta g_k^{(2)} &= (g_k(t-1)-g_k(t-2))\cdot\Big\{\gamma^{(2)}\cdot\arctan\Big(\frac{\Delta Q_k(t-1)}{\gamma^{(2)}\cdot\big(\Delta Q_k(t-2)-\Delta Q_k(t-1)\big)}\Big)\\
&\quad +\tfrac{\pi}{2}\gamma^{(2)}\cdot(1-\mathrm{sgn}((g_k(t-1)-g_k(t-2))\cdot(\Delta Q_k(t-2)-\Delta Q_k(t-1))))\Big\}. && (22)
\end{aligned}
$$

Diese Formel weist recht starke Analogien zu dem Quickprop-Verfahren auf [Zell94], welches mit dem letzten Korrekturschritt die Fehlerfunktion als Parabel approximiert. Der Term $(1 - sgn(..))$ ist 0, wenn sich die Werte $\Delta\bar{Q}$ und ΔQ_k in der gewünschten Richtung verändert haben. Durch den arctan-Term wird eine Begrenzung von zu großen Quotienten vorgenommen, welche bei Quickprop durch eine Fallunterscheidung realisiert wird. Insgesamt wird durch diese Formeln eine zu schwache Korrektur im letzten Lernschritt verstärkt und andererseits eine zu starke Korrektur abgeschwächt. Trotzdem treten sehr häufig wie auch in der ersten Stufe zu große Korrekturterme auf und bewirken eine Oszillation. Ein weiterer Nachteil besteht darin, daß ein kleiner Korrekturterm oft bewirkt, daß erst nach mehreren Schritten wieder Korrekturwerte erreicht werden, die das Ergebnis wesentlich beeinflussen. Dieses Problem entfällt in den Formeln der ersten Stufe. Deshalb wird in [Koch96] vorgeschlagen, die Formeln der ersten Stufe weiterzuentwickeln und dann mit denen der zweiten Stufe zu kombinieren.

Um die Oszillation nach den Formeln (19) und (20) zu dämpfen, werden zwei Hilfsterme eingeführt :

$$
\begin{aligned}
d_\Theta(t) &= \left\{ d_\Theta(t-1) - \tau_1^{(3)} \mathrm{sgn}\left(\Delta\bar{Q}(t-1)\Delta\bar{Q}(t-2) \right) \right\} \tau_2^{(3)}, && (23)\\
d_k(t) &= \left\{ d_k(t-1) - \gamma_1^{(3)} \mathrm{sgn}\left(\Delta Q_k(t-1)\Delta Q_k(t-2) \right) \right\} \gamma_2^{(3)}. && (24)
\end{aligned}
$$

Diese Terme wachsen, wenn die letzte Korrektur zu stark war, und werden kleiner, wenn sie zu schwach war. Sie werden benutzt, um die Korrekturterme nach (19) und (20) zu dämpfen :

$$
\begin{aligned}
\Delta\Theta^{(3)} &= \Delta\Theta^{(1)} \left\{ 1 - \tau_3^{(3)} \left[d_\Theta(t) - 1 \right] \left(1 + \mathrm{sgn}\left(d_\Theta(t) \right) \right) \right\}, && (25)\\
\Delta g_k^{(3)} &= \Delta g_k^{(1)} \left\{ 1 - \gamma_3^{(3)} \left[d_k(t) - 1 \right] \left(1 + \mathrm{sgn}\left(d_k(t) \right) \right) \right\}, && (26)
\end{aligned}
$$

wobei $[x]$ den ganzzahligen Anteil von x darstellt. Eine gleichzeitige Verstärkung von zu schwachen Korrekturen kann durch

$$
\begin{aligned}
\Delta\Theta^{(3)} &= \Delta\Theta^{(1)} (1 + |d_\Theta(t)|)^{-\mathrm{sgn}(d_\Theta(t))}, && (27)\\
\Delta g_k^{(3)} &= \Delta g_k^{(1)} (1 + |d_k(t)|)^{-\mathrm{sgn}(d_k(t))}. && (28)
\end{aligned}
$$

erreicht werden.

In [Koch96] wird angenommen, daß bei einem großen Unterschied zwischen $\Theta^{(2)}$ und $\Theta^{(3)}$ der Wert von $\Theta^{(2)}$ keine sinnvolle Korrektur ergibt. Dies ist analog zu den Problemen, welche vom Quickprop-Verfahren bereits bekannt sind. Daher sollte der tatsächliche Korrekturwert nahe bei $\Theta^{(3)}$ liegen. Ist die Differenz jedoch nicht übermäßig groß, so sollte er näher an $\Theta^{(2)}$ liegen. Analoge Überlegungen werden auch für $\Delta g_k^{(2)}$ und $\Delta g_k^{(3)}$ angestellt. Daraus resultieren die folgenden Korrekturformeln :

$$
\Delta\Theta^{(4)} = \frac{\Delta\Theta^{(2)} + f_\Theta \Delta\Theta^{(3)}}{1 + f_\Theta} \qquad (29)
$$

mit

$$f_\Theta = \exp\left\{\frac{\tau_1^{(4)}\left|\left|\Delta\Theta^{(2)}\right| - \left|\Delta\Theta^{(3)}\right|\right|}{|\Delta\Theta^{(2)}| + |\Delta\Theta^{(3)}|} - \tau_2^{(4)}\right\} \tag{30}$$

und

$$\Delta g_k = \frac{\Delta g_k^{(2)} + f_k \Delta g_k^{(3)}}{1 + f_k} \tag{31}$$

mit

$$f_k = \exp\left\{\frac{\gamma^{(4)}\left|\left|\Delta g_k^{(2)}\right| - \left|\Delta g_k^{(3)}\right|\right|}{\left|\Delta g_k^{(2)}\right| + \left|\Delta g_k^{(3)}\right|} - \gamma_2^{(4)}\right\}. \tag{32}$$

In [Koch96] werden nun die Korrekturterme nochmals begrenzt, um z.B. den Einfluß von ungünstigen Initialisierungen zu begrenzen :

$$\Delta\Theta^{(5)} = \tau^{(5)}\Theta(t-1)\arctan\left(\frac{\Delta\Theta^{(4)}}{\tau^{(5)}\Theta(t-1)}\right), \tag{33}$$

$$\Delta g_k^{(5)} = \gamma^{(5)}\arctan\left(\frac{\Delta g_k^{(4)}}{\gamma^{(5)}}\right). \tag{34}$$

Weiterhin wird festgestellt, daß eine Oszillation des Lernverfahrens entstehen kann, wenn die Quotienten der einen Mustermenge vom Mittelwert $\bar{Q}$ aus hauptsächlich in die eine Richtung streuen, und die Quotienten der anderen Mustermenge vorwiegend in die andere. Ein Maß für diese Asymmetrie erhält man mit

$$\delta = \frac{1}{K}\left|\sum_i \Delta Q_i - \sum_j \Delta Q_j\right|. \tag{35}$$

Daraus kann ein Korrekturfaktor

$$d_\delta = \gamma_1^{(6)} + \frac{2}{\pi}\arctan\left(\frac{(K+2)\gamma_2^{(6)}}{K\delta}\right) \tag{36}$$

bestimmt werden, welcher einfach als zusätzlicher Faktor auf $\Delta\Theta^{(5)}$ und $\Delta g_k^{(5)}$ angewendet wird :

$$\Delta\Theta = d_\delta \cdot \Delta\Theta^{(5)}, \tag{37}$$

$$\Delta g_k^{(6)} = d_\delta \cdot \Delta g_k^{(5)}. \tag{38}$$

Damit ist die Korrektur der Θ-Werte abgeschlossen. Nach den Gleichungen (15) und (38) kann es jedoch vorkommen, daß einzelne g_k negative Werte annehmen. Dies ist allerdings nicht sehr sinnvoll und wird durch

$$g_k^{(7)} = \left\{ \begin{array}{ccc} g_k(t-1) + \Delta g_k^{(6)} & \text{falls} & g_k^{(6)} \geq 0 \\ 0 & \text{falls} & g_k^{(6)} < 0 \end{array} \right\} \tag{39}$$

ausgeschlossen.

Abschließend werden alle g_k skaliert, so daß ihre Summe K ergibt :

$$g_k(t) = \frac{K}{\sum_{k'} g_{k'}^{(7)}} \cdot g_k^{(7)}. \tag{40}$$

Mit diesen Formeln wird die Gewichtskorrektur ausgeführt. Das Verfahren ist nach [Koch96] sehr stabil und zeigt gute Konvergenzeigenschaften.

2.4 Das Lernverfahren

Auf der Basis dieser Korrekturterme wird in [Koch96] ein Verfahren beschrieben, wie ein einzelnes Neuron angelernt werden kann.

Es soll noch einmal hervorgehoben werden, daß die Quotienten für Muster der ersten Mustermenge im angelernten Zustand kleiner als $Q^{(S)}$ sein sollen und für Muster der zweiten Mustermenge größer als $Q^{(S)}$. Während des Lernvorganges wird jedoch die Definition der Quotienten für Muster der zweiten Menge geändert, so daß alle Quotienten kleiner als $Q^{(S)}$ sein sollen.

Das Lernverfahren besteht im Kern aus zwei eingebetteten Schleifen. In der äußeren Schleife (Schritte 2. bis 6.) wird für jede Mustermenge eine Teilmenge von ihr als aktive Lernmenge definiert. In der inneren Schleife (Schritte 3. bis 5.) werden die Gewichte nach den oben hergeleiteten Formeln so lange korrigiert, bis alle Muster der beiden aktiven Lernmengen korrekt klassifiziert werden, das heißt alle ihre Quotienten kleiner als $Q^{(S)}$ sind.

1. **Initialisierung:** Die zu lernenden Muster bestehen aus zwei disjunkten Teilmengen, welche unterschieden werden sollten. Für jedes Muster gibt es ein Hilfsgewicht g_k. $Q^{(S)}$ ist der gewünschte Quotient, er muß kleiner als 1 sein. Die beiden aktiven Lernmengen sind leer.

2. **Anpassung der aktiven Lernmengen:** Für alle Muster werden die Quotienten Q_k berechnet (dazu müssen zuvor natürlich aus Θ und den g_k die Gewichte bestimmt worden sein). Für jede Mustermenge wird das Muster mit dem größten Q_k in die jeweilige aktive Lernmenge aufgenommen, wenn der Quotient größer als $Q^{(S)}$ ist. Seine Wichtung g_k wird mit 1 initialisiert, auch wenn es sich um eine wiederholte Aufnahme handelt.

3. **Gewichtskorrektur:** Nach den oben angegebenen Formeln werden $\Delta\Theta$ und alle diejenigen Δg_k, für die das Muster k in einer aktiven Lernmenge enthalten ist, berechnet, die Änderungen ausgeführt und die g_k skaliert.

4. **Entfernen überflüssiger Muster:** Alle Muster, deren Wichtung g_k in diesem Lernschritt 0 geworden ist, werden aus der aktiven Lernmenge entfernt.

5. **Aktive Lernmenge gelernt?:** Aus Θ und den g_k werden die Gewichte berechnet, danach werden für alle Muster der aktiven Lernmengen die Quotienten berechnet. Wenn nicht alle Quotienten kleiner als $Q^{(S)}$ sind, wird mit Schritt 3. fortgesetzt.

6. **Alle Muster gelernt?:** Es werden alle Quotienten gebildet. Wenn mindestens einer größer als $Q^{(S)}$ ist, wird mit Schritt 2. fortgesetzt.

7. **Ende**

Nach Aussage von Koch werden die in den verschiedenen Lernschritten zu berechnenden Gewichte nur aus den Mustern, die in der aktiven Lernmenge enthalten sind, (und den dazugehörigen g_k und Θ) bestimmt. Die endgültigen Gewichte des Neurons ergeben sich also aus den Mustern, welche in der letzten aktiven Lernmenge enthalten waren, als alle Muster korrekt klassifiziert wurden, welche nicht notwendigerweise mit der vollständigen Mustermenge identisch sein muß.

Zum 4. Lernschritt sei noch angemerkt, daß das Entfernen eines Musters k aus der aktiven Lernmenge keine absolut sichere Aussage über die Beziehung des Quotienten dieses Musters zu $Q^{(S)}$ zuläßt. Es wird entfernt, wenn seine Wichtung g_k im letzten Korrekturschritt kleiner als 0 geworden ist. Daraus kann man lediglich schlußfolgern, daß der letzte Korrekturterm für dieses g_k negativ war, was nach den oben angegebenen Grundzusammenhängen bedeutet, daß der Quotient Q_k kleiner als der "gewünschte Quotient" war.

Dieser ist aber nicht $Q^{(S)}$ (wie man eventuell vermuten könnte), sondern nach der Formel (18) der Mittelwert aller Quotienten $\bar{Q}$. Da man für diesen nicht mit Sicherheit abgeschätzen kann, in welcher Relation er zu $Q^{(S)}$ steht, ist auch eine zuverlässige Aussage für die Relation von Q_k zu $\bar{Q}$ nicht möglich.

2.5 Implementation

Bei der Implementation des oben vorgestellten Verfahrens ergeben sich einige kleinere Probleme mathematischer Natur. In [Koch96] werden verschiedene nicht gerade unerhebliche Details offengelassen, für deren Umsetzung Festlegungen getroffen werden mußten.

1. **Die Funktion $W(x)$:** Die einzige Angabe, die zu dieser Funktion gemacht wurde, ist die Aussage, daß es sich um eine monoton wachsende Funktion handeln muß. Bei der Implementation wurde dafür die Sigmoide

$$W(x) = \frac{1}{1 + e^{-x}} \tag{41}$$

verwendet.

2. **$\Theta(t-2), g_k(t-2), \Delta\bar{Q}(t-2), \Delta Q_k(t-2)$:** In den Formeln der Stufen 2 und 3 (Gleichungen (21) bis (26)) werden diese Werte aus dem vorherigen Zeitschritten verwendet. Wird jedoch die innere Schleife zum ersten mal durchlaufen, so sind diese Werte undefiniert. Bei der Implementation des Verfahrens wurden deshalb im ersten Schleifendurchlauf die Werte $\Delta\Theta^{(2)}$ und $\Delta\Theta^{(3)}$ durch $\Delta\Theta^{(1)}$ und die Werte $\Delta g_k^{(2)}$ und $\Delta g_k^{(3)}$ durch $\Delta g_k^{(1)}$ ersetzt.

3. **d_Θ und d_k:** Auch bei der Berechnung diese Werte werden in den Gleichungen (23) und (24) ihre Werte aus dem vorherigen Zeitschritt benutzt. Da diese Gleichungen wegen dem Punkt 2 bereits umgangen werden, ist nur noch eine Initialiserung dieser Werte für ihre Verwendung im zweiten Schritt nötig, dazu wird der Wert 0 verwendet.

4. **Interpretation von $\lessapprox$:** Die Bedeutung dieses selbstdefinierten Vergleichsoperators ist "Höchstens ein wenig größer". Um diese vage Aussage rechnerisch umsetzen zu können, wurde sie als $x <= 1.1 \cdot y$ interpretiert. Im allgemeinen birgt diese Vorgehensweise einen potentiellen

Fehler in sich, nämlich genau dann, wenn durch diese zehnprozentige Erhöhung des Schwellwertes ein qualitativ anderes Verhalten erzeugt wird.

5. **Parameter:** Die Parameter $\tau_i^{(j)}$ und $\gamma_i^{(j)}$ haben einen sehr wesentlichen Einflußauf die Ergebnisse. Da ihre Anzahl recht groß und ihr Zusammenspiel recht komplex ist, ist es sehr aufwendig, durch Experimentieren optimale Parameter zu finden. Es wurde deshalb auf die in [Koch96] angegebenen Werte zurückgegriffen, sie sind in Tabelle 1 aufgeführt.

Stufe	1	2	3	4	5	6
Parameter						
τ		2			$\frac{3}{10\pi}$	
τ_1	$\frac{1}{10\pi}$		$\frac{1}{2}$	7		
τ_2	$\frac{5\pi}{2}$		$\frac{9}{10}$	6		
τ_3			$\frac{1}{6}$			
γ	$\frac{1}{5}$	$\frac{4}{\pi}$			$\frac{4}{5\pi}$	
γ_1			1	7		$\frac{1}{5}$
γ_2			$\frac{4}{5}$	6		$\frac{1}{5}$
γ_3			$\frac{1}{4}$			

Tabelle 1: *Lernparameter*

6. **Gewichtsberechnung:** Die Gewichte berechnen sich aus den Mustern, welche in den aktiven Lernmengen enthalten sind. Zu Beginn des Verfahrens sind diese leer, andererseits ist jedoch eine Gewichtsberechnung notwendig, um zu entscheiden, welche Muster in die aktiven Lernmengen aufgenommen werden sollen. Würde man formal vorgehen, ergäben sich die Vektoren $\vec{x}^{(1)}$ und $\vec{x}^{(2)}$ als Nullvektoren und somit alle Gewichte (bei $\Theta = 1$) als 0.5. Dies wiederum hat zur Folge, daß alle Quotienten 1 sind. Eine sinnvolle Auswahl der Startlernmengen ist also nicht möglich. Um diese Situation zu umgehen, wurden für die allererste Gewichtsberechnung (wenn also die aktiven Lernmengen noch leer sind) die gesamte Lernmenge herangezogen.

7. **Quotienten in den Korrekturformeln:** In den Korrekturformeln tauchen an verschiedenen Stellen Brüche auf, deren Nenner sehr leicht den Wert 0 annehmen können. Wenn der Bruch im Argument einer arctan-Funktion steht (dies sind alle relevanten Fälle), so wird das

Ergebnis dieses arctan unter Beachtung des Vorzeichens des Zählers als $\frac{\pi}{2}$ bzw. $-\frac{\pi}{2}$ angegeben.

8. **Quotienten bei der Gewichtsberechnung:** Bei der Berechnung des Quotientenvektors $\vec{q}$ als elementweisen Quotienten aus den gewichteten Mustermittelwerten $\vec{x}^{(1)}$ und $\vec{x}^{(2)}$ tritt ebenfalls sehr häufig eine 0 im Nenner auf, insbesondere wenn nur recht wenige Muster in $\vec{x}^{(1)}$ eingehen bzw. die Muster einen relativ hohen Anteil an 0-Werten haben (was bei Zeichenerkennung meistens der Fall ist). Um diese Probleme zu eliminieren, werden alle Elemente der Vektoren $\vec{x}^{(1)}$ und $\vec{x}^{(2)}$ vor der Quotientenbildung auf den Wert 0.0001 gesetzt, wenn sie kleiner als dieser waren. Da der Quotient bzw. sein Reziprokes letzendlich als ein Argument für die Sigmoide dienen, verfälscht diese Vorgehensweise das Ergebnis nur unwesentlich.

9. **Exponentenüberlauf:** Um bei der Berechnung der Ergebnisse von $W(x)$ einen Exponentenüberlauf zu verhindern, wird der Definitionsbereich der Funktion so begrenzt, daß ihre Ergebnisse im Bereich der rechnerintern darstellbaren Zahlen liegt. Argumente außerhalb dieses Definitionsbereiches werden auf dessen Rand verschoben.

Damit sind alle für das Verfahren relevanten Randbedingungen abgesteckt, und der detailierten Untersuchung steht nichts mehr im Wege.

2.6 Erste Tests

Für die ersten Untersuchungen des Lernverfahrens wurden zwei Beispiele erzeugt. Das erste Beispiel ist ein recht triviales. Mit einem selbstgeschriebenen Mustereditor wurden für die Ziffern 0 bis 9 je ein Prototyp in einem 8x8-Feld erzeugt, wobei jedes Rasterelement nur den Wert 0 oder 1 annehmen konnte. Als zweites Beispiel wurden einige Muster aus einer Datenbasis, welche handgeschriebene Ziffern aus US-amerikanischen Postleitzahlen enthält, ausgewählt. Die Muster liegen in einem 16x16-Raster als 256-Graustufenbilder vor. Die verwendeten Muster sind im Anhang dieser Arbeit abgebildet.

2.6.1 Erstes Beispiel - einfache Ziffern

Überraschenderweise gab es mit den vermeintlich einfachsten Beispielen erhebliche Probleme. Die anzulernende Lernmenge bestand jeweils aus zwei

Mustern verschiedener Klassen. Dieser Fall mag zwar zu trivial erscheinen, ist jedoch sinnvoll, da die ersten aktiven Lernmengen grundsätzlich aus je einem einzigen Muster bestehen. Dies ist insofern ein interessanter Sonderfall, als daß sich der Quotientenvektor $\vec{q}$ aus dem elementweisen Quotienten der beiden (eventuell gewichteten) Mustervektoren berechnet. Als anzustrebender Zielquotient $Q^{(S)}$ wurde 0.6 eingesetzt.

Egal welche Musterpaare ausgewählt wurden, es gab qualitativ nur zwei verschiedene Fälle:

- die Muster wurden von Anfang an korrekt unterschieden, es war kein Lernen nötig
- die Muster wurden anfangs nicht korrekt unterschieden, es war jedoch auch nicht möglich, sie anzulernen. (Hätte man $Q^{(S)}$ erhöht, so wäre ein Lernen möglich gewesen, aber den Lernerfolg auf diese Art zu erreichen wäre ganz gewiß nicht das Ziel gewesen - zumal die Forderung $Q^{(S)} = 0.6$ bei weitem nicht überzogen ist.)

Der erste Fall trat bei Kombinationen auf, bei denen nur sehr wenige Punkte in allen beiden Mustern gesetzt sind, also die "Und-Verknüpfung" der beiden Muster fast weiß erscheint. Da die Anzahl der Stellen, an denen in genau einem der beiden Muster der entsprechende Punkt gesetzt ist, gegenüber denen, an denen in beiden Mustern der entsprechende Punkt gesetzt ist, recht groß ist, und an diesen Stellen die entstehenden Gewichte $\vec{w}^{(1)}$ bzw. $\vec{w}^{(2)}$ einen großen Wert annehmen (siehe Abbildung 4), unterscheiden sich, wie in Abbildung 5 zu erkennen, die Aktivierungen $net^{(1)}(\vec{x}_k) = \vec{x}_k \cdot \vec{w}^{(1)}$ und $net^{(2)}(\vec{x}_k) = \vec{x}_k \cdot \vec{w}^{(2)}$ sehr stark. Infolge dessen liegt der Quotient Q_k meist weit unter dem geforderten Wert, und ein Lernen ist nicht nötig.

Im zweiten Fall zeigten sämtliche Lernversuche (auch bei Verwendung von Korrekturtermen der verschiedenen Stufen) prinzipiell das in den Bildern 10 bis 26 am Beispiel der Kombination 3 und 9 und den Korrekturstufen 1, 4 und 6 dargestellte qualitative Verhalten.

Untersuchung der Korrektur der g_k

Die Quotienten hatten am Anfang zu große Werte, diese näherten sich relativ schnell aneinander an, und nahmen dann einen gemeinsamen Verlauf. Diese Tatsache ist nicht verwunderlich, sondern liegt daran, daß nach

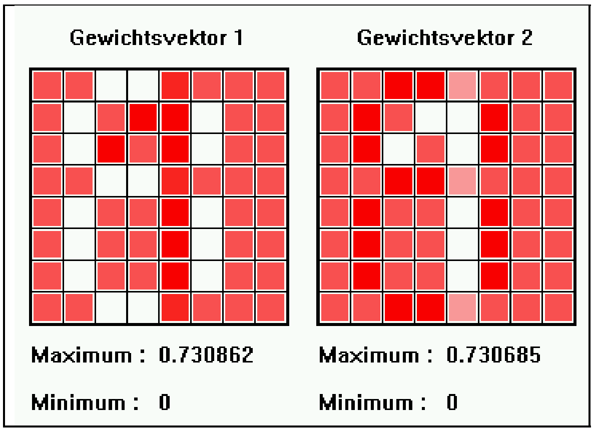

Abbildung 4: *Die Gewichtsvektoren für ein Neuron, welches die Muster 1 und 8 unterscheidet.*

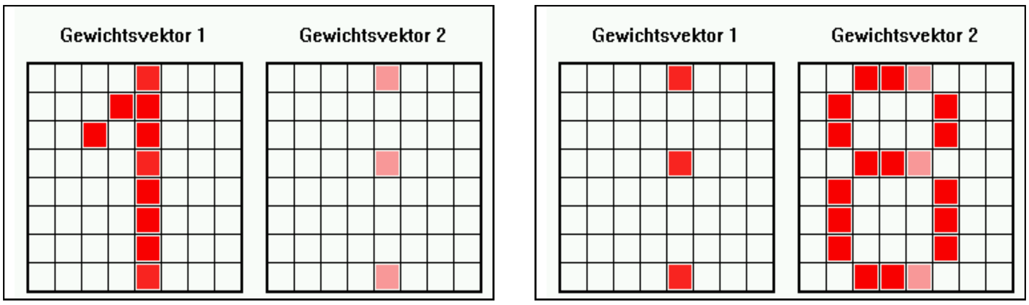

Abbildung 5: *Ergebnis des elementweisen Produkts der Muster 1 und 8 mit den Gewichtsvektoren. (Die Summe über die einzelnen Elemente des Vektors ergibt das Skalarprodukt.)*

der Formel (18) das direkte Ziel der Korrektur der Wichtungen g_k nicht der gewünschte Quotient $Q^{(S)}$ ist, sondern der Mittelwert $\bar{Q}$, welcher bei einer zweielementigen Lernmenge dem Mittel der Quotienten dieser beiden Muster entspricht. Die beiden Quotienten konvergieren also sehr rasch gegen ihr arithmetisches Mittel. Sind sie erst einmal nahe beieinander, so sind die Werte ΔQ_k und somit Δg_k sehr klein und bewirken fast keine Veränderung mehr. Dieser Verlauf war im Prinzip von der verwendeten Korrekturstufe unabhängig, die verschiedenen Stufe hatten lediglich einen Einfluß auf die Beträge der Korrektur. In den Stufen 4 und 6 wurde stärker korrigiert, so daß sich die Quotienten der Muster stärker veränderten und somit die Δg_k unregelmäßiger als in der Stufe 1 waren, in der Stufe 6 sogar leicht oszillierten. — Grundsätzlich war also ab einem gewissen Punkt mit Korrekturen der g_k keine Veränderung zu erreichen, der Erfolg des Lernens hing einzig und allein von den Korrekturen für Θ ab.

Untersuchung der Korrektur von Θ

Betrachtet man den Verlauf des Wertes von Θ über die gesamte Lernphase, so lassen sich zwei Qualitäten unterscheiden:

1. Θ läuft mehr oder weniger ungebremst gegen $-\infty$ (Korrekturstufen 1 und 4), oder
2. Θ konvergiert asymptotisch gegen 0 (Korrekturstufe 6).

Im ersten Fall (Korrekturstufen 1 und 4) sind während der gesamten Lernphase die einzelnen $\Delta\Theta$ negativ. Dadurch ergeben sich alle Gewichte unabhängig von den jeweils dazugehörenden Elementen q_n des Quotientenvektors ab einem bestimmten Zeitpunkt als Ergebnis der Funktion $W(x)$ von einem negativen Argument mit großem Betrag. Ist dieser Betrag groß genug, werden alle Elemente des Gewichtsvektors fast 0. In Folge dessen sind die Aktivierungen $net_i(\vec{x}_k)$ und $net_j(\vec{x}_k)$ fast 0 und insbesondere fast gleich, so daß sämtliche Quotienten 1 werden. Da dieser Wert immer noch größer als der gewünschte Quotient $Q^{(S)}$ ist, wird Θ weiter verkleinert. Die Korrektur der g_k bleibt weiterhin 0, da die einzelnen Quotienten Q_k mit dem Mittelwert $\bar{Q}$ zusammenfallen. Aus diesem Teufelskreis gibt es kein Entrinnen.

Im zweiten Fall (Korrekturstufe 6) konvergiert Θ recht schnell gegen 0. Dieses Verhalten ist vor allem die Folge der Skalierung mit dem Wert von Θ selbst bei der Berechnung von $\Delta\Theta^{(5)}$ (Gleichung (28)). Da $\bar{Q}$ zu groß ist, wird Θ verkleinert, es läuft also von seinem Startwert 1 gegen den Nullpunkt. Je kleiner Θ jedoch wird, um so kleiner werden auch seine Korrekturen. Dadurch ist es jedoch nicht in der Lage, negative Werte anzunehmen. Dieser Umstand, und die Tatsache, daß die Wichtungsänderungen g_k ebenfalls gegen 0 konvergieren, führt auch in diesem Fall das Lernverfahren in eine Sackgasse. Der Unterschied zum ersten Fall besteht einzig und allein darin, daß hier die Quotienten nicht auf 1 springen, sondern ihren Wert beibehalten.

Diese Tatsachen mögen etwas niederschmetternd erscheinen. Jedoch in Anbetracht dessen, daß das Verfahren bereits erfolgreich eingesetzt worden ist, entsteht die Vermutung, daß die Ursache für das Versagen in diesem Fall eher in der verwendeten Beispielmenge zu suchen ist.

Untersuchung der Muster

Die beiden in diesem Beispiel verwendeten Muster haben an den gesetzten Punkten nach der Normierung den Wert $\frac{1}{\sqrt{15}}$ (Muster 3) bzw. $\frac{1}{\sqrt{18}}$ (Muster 9). Somit ergeben sich für die Elemente des Quotientenvektors $\vec{q}$ folgende mögliche Werte :

1. Beide Muster besitzen in dem entsprechenden Punkt den Wert 0. Dann ist q_n 1, da die Werte $\vec{x}_n^{(1)}$ und $\vec{x}_n^{(2)}$ jeweils auf 0.0001 gesetzt werden. (Genaugenommen spielt dieser Wert aber keine Rolle, da er sowieso nicht mit in das Skalarprodukt eingeht.)
2. Der Punkt in Muster 3 ist gesetzt, der korrespondierende in Muster 9 jedoch nicht. Entsprechend den Festlegungen ergibt sich $q_n = \frac{10000}{\sqrt{15}} \cdot g_3$.
3. Der Punkt in Muster 9 ist gesetzt, der korrespondierende in Muster 3 jedoch nicht. Entsprechend den Festlegungen ergibt sich in diesem Falle $q_n = \frac{\sqrt{18}}{10000 \cdot g_9}$.
4. Beide Punkte sind gesetzt. Dann ist $q_n = \frac{\sqrt{18}}{\sqrt{15}} \cdot \frac{g_3}{g_9}$.

Betrachtet man den stationären Zustand, der sich nach einer gewissen Zeit einstellt, so stabilisieren sich die Werte für g_3 bei etwa 0.96 und für g_9 bei etwa 1.04. Damit ergeben sich für die jeweils angegebene Anzahl von Quotienten die möglichen Werte:

1. 46 mal : 1.0
2. 0 mal : 2478.7

3. 3 mal : 0.000408
4. 15 mal : 1.011

Da die Werte der einzelnen Musterelemente fast gleich sind, können die 15 Gewichte, die zu den in beiden Mustern gesetzten Punkten gehören, keine quantitativ ausreichende Unterscheidung herbeiführen. Der entscheidende Einfluß müßte also durch die 3 Gewichte entstehen, für die deren zugehörigen Musterelemente einen Unterschied aufweisen. Da jedoch die Gewichte nur Werte zwischen 0 und 1 annehmen können, und andererseits die Elemente der Muster selbst nur einen Wert von 0.246 haben, ist deren Einfluß gegenüber dem der 15 gleichen Musterelemente ebenfalls nicht ausreichend. Ein Blick auf Abbildung 6 verdeutlicht dies.

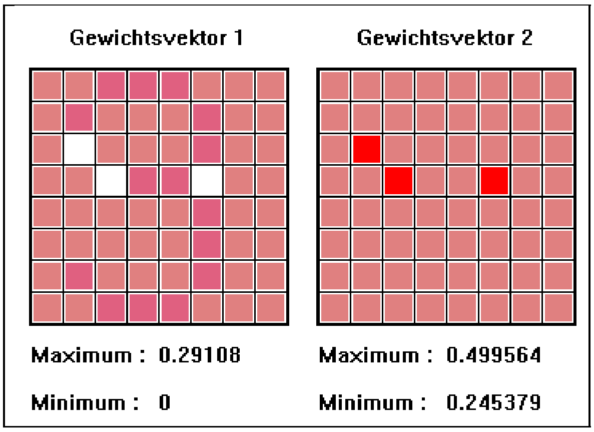

Abbildung 6: *Die Gewichtsvektoren für ein Neuron, welches lernen soll, die Muster 3 und 9 zu unterscheiden. Ein Zustand mitten im Lernprozeß.*

Fazit

Das Lernverfahren ist ohne Modifikationen offensichtlich nicht in der Lage, sich nur wenig unterscheidende Binärmuster anzulernen. Nicht nur das Verfahren, sondern auch das auf diese Weise konstruierte Neuron selbst, sind für derartige Problemstellungen nicht geeignet. Es wäre dann sinnvoll, z.B. nicht die Muster selbst zu verwenden, sondern deren Fouriertransformierten. Dann schlagen sich die soeben beschriebenen minimale Unterschiede zwischen Mustern in wesentlich auffälligeren Unterschieden im Fourier-Spektrum nieder.

2.6.2 Zweites Beispiel - handgeschriebene Ziffern aus Postleitzahlen

Die in diesem Beispiel verwendeten handgeschriebenen Ziffern stammen aus US-amerikanischen Postleitzahlen. Sie liegen als 256-Grauwert-Bilder in einem 16x16 Raster vor. Durch Vorverarbeitung wurden die Ziffern bereits an die Größe des Rasters optimal angepaßt und zentriert. Dies ist besonders wichtig, da das vorgestellte Verfahren nicht skalierungs-, translations- oder rotationsinvariant arbeitet.

Die ersten Versuche mit den originalen Mustern waren wiederum nicht erfolgreich. Aber auch hier gibt es eine einfache Erklärung. In allen Mustern war der Hintergrund mit dem Wert 64 belegt, was ein Viertel des maximalen Wertes ist. Da in vielen Mustern solche Hintergrundpunkte korrespondierten, stellte sich für sie ein Gewicht ein, welches deutlich größer 0 war. Da natürlich die Hintergrundpunkte in einer wesentlich höheren Anzahl auftraten, als die eigentlich relevanten Punkte, haben sie auf den Quotienten einen extrem großen Einfluß gehabt, welcher durch das Lernverfahren nicht kompensiert werden konnte.

Damit ergibt sich eine weitere Forderung an die Muster, die mit Hilfe des vorgestellten Verfahrens angelernt werden sollen : Die Anzahl der Punkte, die für die Unterscheidung der Muster relevant sind, darf gegenüber der Anzahl der von 0 verschiedenen Punkte, die in allen Mustern den gleichen Wert haben, nicht deutlich geringer sein. (Anderer Ausweg : Fouriertransformation, siehe oben)

Nachdem in allen Mustern der Hintergrund durch den Wert 0 ersetzt wurde, konnte das Lernverfahren wesentlich bessere Ergebnisse erzielen. So war es mit Formeln der Korrekturstufe 1 möglich, jede untersuchte Kombination von einzelnen Klassen (z.B. alle Muster der Klasse 0 und der Klasse 6) zu unterscheiden. Bei Verwendung von Korrekturformeln der Stufe 6 war dies jedoch in der Regel nicht mehr möglich. Ebenfalls gelang es nicht, beliebige Kombinationen von Mustern zu unterscheiden (z.B. alle Muster der Klassen 0 und 4 von allen Mustern der Klassen 6 und 7), wenn sie nicht eine gewisse Ähnlichleit miteinander hatten oder ihre Anzahl nicht besonders klein war. Der Grund dafür liegt wieder in dem Einfluß, welchen insbesondere Translationen oder Skalierungen von Mustern auf die jeweiligen Quotienten haben.

Der Verlauf des Lernprozesses war von der konkreten Aufgabe relativ unabhängig. Im folgenden werden typische Charakteristika an dem Beispiel der Unterscheidung aller Muster der Klasse 0 (Nummern 1 bis 11) von allen Mustern der Klasse 6 (Nummern 77 bis 92) erläutert.

Korrekturen nach Stufe 1

Die Verläufe der Quotienten, Wichtungen und deren Änderungen für die Muster Nr. 5 und 80 und die Verläufe von Θ und $\Delta\Theta$ sind in den Bildern 27 bis 34 dargestellt. Die Muster Nr. 5 (stellt eine 0 dar) und 80 (stellt eine 6 dar) wurden als Beispiel ausgewählt, weil einerseits der Verlauf ihrer Werte charakteristisch für alle Muster ist und sie andererseits immer in den aktiven Lernmengen enthalten waren. Die Zugehörigkeit der einzelnen Mustern zu den aktiven Lernmengen ist in der Tabelle 2 dargestellt. Die beiden Gewichtsvektoren des Neurons haben sich so eingestellt, wie es zu erwarten war, Abbildung 7 verdeutlicht dies. Der eine hat besonders an den Stellen, an denen die Muster der Klasse 0 große und die Muster der Klasse 6 kleine Werte haben, große Werte, der andere hat sich genau auf die umgekehrten Fälle eingestellt. Die Elemente, welche in beiden Klassen ähnlich häufig bzw. stark gesetzt sind, weisen relativ geringe Gewichte auf.

Äußere Schleife	Innere Schleifen	Aktive Lernmenge der Menge 1	Aktive Lernmenge der Menge 2
1	0	5	80
2	3	5, 11	79, 80
3	19	2, 5, 11	78, 79, 80
4	76	2, 5, 6, 11	78, 79, 80, 87
5	45	2, 3, 5, 6, 11	77, 78, 79, 80, 87
6	38	2, 3, 5, 6, 9, 11	77, 78, 79, 80, 82, 87
7	51	2, 3, 5, 6, **7**, 9, **11**	77, 78, 79, 80, 82, 87, 89
8	50	2, 3, 5, 6, **7**, 9	77, 78, 79, 80, 82, 85, 87, 89
9	47	2, 3, 5, 6, **7**, 9	77, 78, 79, 80, 82, 84, 85, 87, 89
10	23	2, 3, 5, 6, 7, 9	77, 78, 79, 80, 82, 83, 84, 85, 87, 89
11	38	2, 3, 5, 6, **7**, 8, 9	77, 78, 79, 80, 82, 83, 84, 85, 87, 89
12	25	2, 3, 5, 6, 7, 8, 9	77, 78, 79, 80, 81, **82**, 83, 84, 85, 87, 89
13	12	2, 3, 5, 6, 7, 8, 9	77, 78, 79, 80, 81, 83, 84, 85, 86, 87, 89

Tabelle 2: *Zugehörigkeit zu den aktiven Lernmengen. Fett geschriebene Muster wurden in der jeweiligen äußeren Schleife aus den aktiven Lernmengen entfernt.*

Generell waren insbesondere in den ersten Lernschritten recht hektische Korrekturen zu erkennen, welche teilweise leicht oszillierten. Gegen Ende der inneren Schleife nahme diese Osziallation jedoch ab. Das Entfernen eines Musters aus der Lernmenge sorgte jedoch oft für eine kurze Unruhephase. Der Grund dafür ist, daß in einer solchen Situation plötzlich ein Summand bei der Bildung von $\bar{Q}$ entfällt und somit dieser Wert oft einen größeren Sprung

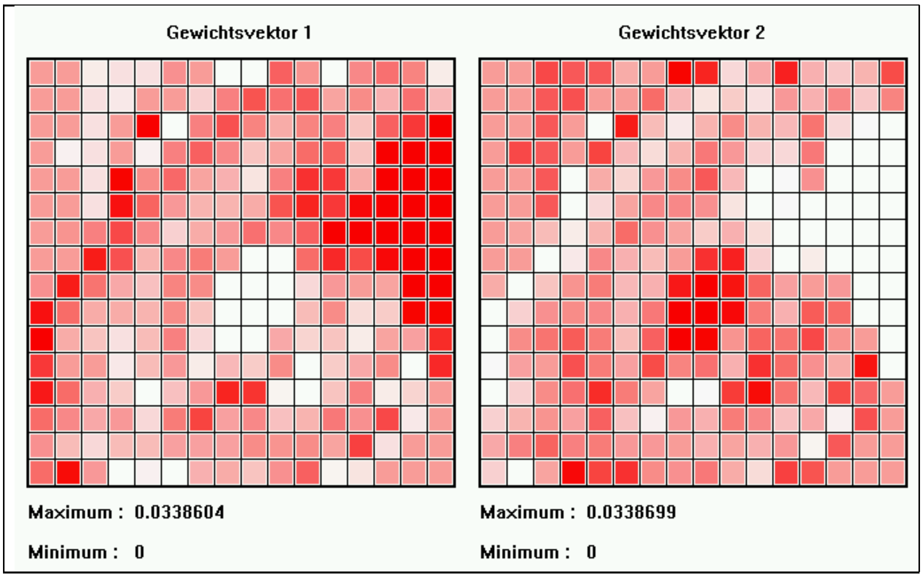

Abbildung 7: *Gewichtsvektoren eines Neurons, das die Klassen 0 und 6 unterscheidet*

macht, als dies bei normalen Gewichtskorrekturen der Fall ist. Der Verlauf der einzelnen Kurven spricht dafür, daß das Korrekturverfahren sehr schnell die richtige Korrekturen ausführt, andererseits jedoch unnötige Korrekturen schnell kompensiert und unterläßt.

Das Hinzunehmen eines Musters hat, wie auch das Entfernen, eine kurze Unruhephase zur Folge. Dies ist regelmäßig zu Beginn einer neuen inneren Schleife zu beobachten.

Der Wert von Θ sinkt mit der Zeit immer weiter. Dies hat seine Ursache in der Korrekturvorschrift. Ist der Mittelwert aller Quotienten zu groß, was wie in Abbildung 35 zu erkennen in diesem Beispiel immer der Fall war, wird nach Gleichung (19) Θ immer weiter verringert.

Das Lernen erfolgte hier mit einem erwünschten Quotienten von $Q^{(S)} = 0.6$. In [Koch96] wurde vorgeschlagen, für $Q^{(S)}$ mit einem kleineren Wert anzufangen, als dem eigentlich zu erreichenden. Diese Heuristik ist noch relativ gut beherrschbar, wenn man nach jedem Lernschritt eingreifen will bzw. kann. In einem automatischen Verfahren ist dies natürlich nicht erwünscht. Zu Testzwecken wurde mit $Q^{(S)} = 0.5$ gestartet und nach jeder inneren Schleife der Wert um die Hälfte seines Abstandes zu 0.6 erhöht. Damit ergab sich eine Folge von $Q^{(S)} = 0.5, 0.55, 0.575, \ldots$. Mit dieser konvergierte das Lernverfahren noch etwas schneller, da die ersten Muster "genauer" gelernt wurden, als dies bei konstanten $Q^{(S)}$ der Fall war.

Bei der genaueren Betrachtung der Tabelle 2 fällt auf, daß das Muster 7 häufig aus der aktuellen Lernmenge entfernt wurde (weil seine Wichtungen g_k 0 wurde), aber im nächsten Schritt wieder eingefügt werden mußten, da sein Quotient zu schlecht war. Dies geht mit der bei der Beschreibung des Lernverfahrens getroffenen Aussage über den Zusammenhang von Wichtung und Quotient einher: Wichtung gleich 0 bedeutet nicht unbedingt, daß der Quotient sehr klein wäre. Auf Grund dieser Beobachtung sollte man darüber nachdenken, Muster mit Wichtung 0 nicht aus der Lernmenge zu entfernen, damit ihre Quotienten nicht sofort wieder zu groß werden. Damit würde man zwei Fliegen mit einer Klappe schlagen: die Unruhephase der Wichtungen nach dem Entfernen eines Musters durch den Sprung des Mittelwertes $\bar{Q}$ würde ebenfalls nicht mehr auftreten. Außerdem wäre dann die Anzahl der zum Lernen maximal benötigten äußeren Schleifen begrenzt. Das Lernen startet mit je einem Muster in den aktiven Lernmengen. Nach jeder äußeren Schleife wird mindestens ein Muster hinzugenommen, oder das Lernen ist bereits erfolgreich beendet. Somit werden maximal $K - 1$ äußere Schleifen für das Lernen benötigt. (Natürlich schließt dies noch nicht den Fall aus, daß in einer inneren Schleife die aktiven Lernmengen nicht angelernt werden können.)

Korrekturen nach Stufe 6

Auch bei dem Test mit den Korrekturformeln der Stufe 6 seien wieder die Muster 5 und 80 als charekteristische Beispiele betrachtet. Die Verläufe der Quotienten, Wichtungen und deren Änderungen der Muster sowie die Verläufe von Θ und $\Delta\Theta$ sind in den Bildern 37 bis 44 dargestellt. Des weiteren sind in der Tabelle 3 die Zugehörigkeiten zu den aktiven Lernmengen angegeben.

Auch beim Lernen der Ziffern aus den Postleitzahlen tritt bei der Verwendung der Korrekturstufe 6 offensichtlich das gleiche Problem zutage, wie im ersten Beispiel. Der Wert von Θ nähert sich sehr schnell an 0 an, und kann dann diese nicht überspringen. Das wiederum wäre, wie am Erfolg der Korrektur nach Stufe 1 zu erkennen ist, jedoch vermutlich notwendig, um das Problem zu lösen.

Unter Betrachtung der Gleichung 33 wäre es eine Möglichkeit, diese Schwelle durch Ändern des Parameters $\tau^{(5)}$ zu überwinden. Allein der prak-

Äußere Schleife	Innere Schleifen	Aktive Lernmenge der Menge 1	Aktive Lernmenge der Menge 2
1	0	5	80
2	2	5, 11	79, 80
3	9	5, 6, 11	78, 79, 80
4	∞	3, 5, 6, 11	77, 78, 79, 80

Tabelle 3: *Zugehörigkeit zu den aktiven Lernmengen.*

tische Versuch war nicht von Erfolg gekrönt. Bis zu einem Wert von ca. 0.36 verändert sich das qualitative Verhalten nicht, bei geringfügig größeren Werten konvergiert Θ sehr schnell gegen $-\infty$, was natürlich zur Folge hat, daß alle Quotienten 1 werden und das Verfahren ebenfalls nicht erfolgreich ist.

2.7 Vergleich mit Backpropagation

Um einen Vergleich bezüglich der Effizienz des IWD-Lernverfahrens zu erhalten, wurde ein einfaches Perzeptron mit dem Lernverfahren Backpropagation angelernt, die Klassen 0 und 6 zu unterscheiden. Der Verlauf des Fehlers in der Lernphase ist in Abbildung 46 dargestellt. Wie in Abbildung zu sehen ist, weisen die positiven und die negativen Elemente des Gewichtsvektors eine ähnliche Struktur wie die zwei Gewichtsvektoren des Q-Neurons (vergleiche Abbildung 8) auf.

Die Lernrate η wurde auf 1.0 gesetzt. Nach etwa 150 Lernschritten hatte das Perzeptron einen Gesamtfehler von weniger als 0.1 erreicht. Die Anzahl der benötigten Lernschritte hing sehr wesentlich von der gewählten Lernrate ab, auch ein Erhöhen derselben auf 10.0 war noch möglich, ohne daß beim Lernen das Ziel übersprungen wurde (was genau die Gefahr bei zu großer Lernrate ist).

Bei einem Vergleich der Effizienz ist offensichtlich, daß Backpropagation bei diesem Beispiel mit 150 Lernschritten wesentlich schneller zum Erfolg kommt, als das IWD-Verfahren mit ca. 420 Korrekturen. Wenn man außerdem noch bedenkt, daß ein IWD-Korrekturschritt deutlich aufwendiger ist, als ein BP-Lernschritt, so wird der Unterschied noch deutlicher.

Für weitere Betrachtungen sei auf den Abschnitt 5.4 verwiesen.

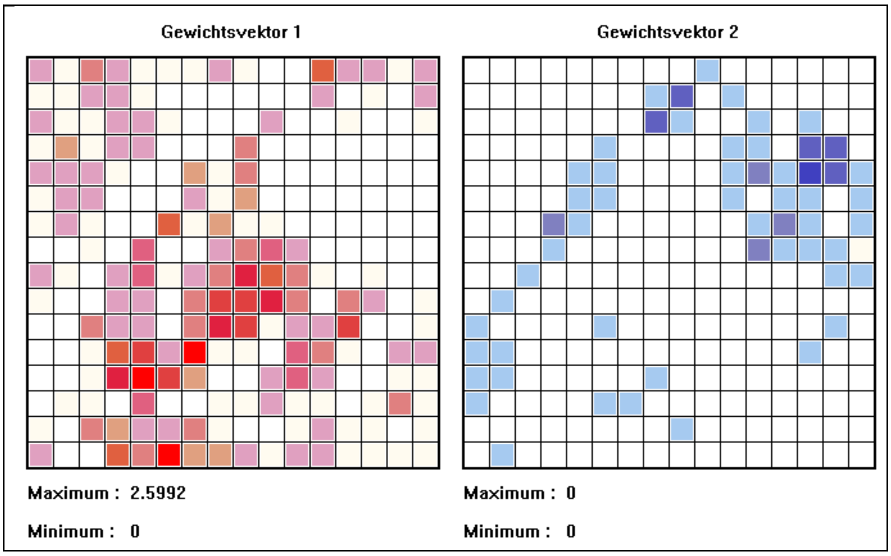

Abbildung 8: *Gewichtsvektoren eines Perzeptrons, das die Klassen 0 und 6 unterscheidet. Links sind die positive Elemente des Gewichtsvektors dargestellt, rechts die negativen.*

2.8 Zusammenfassung

Das in diesem Abschnitt vorgestellte Neuron berechnet im Gegensatz zu den meisten bekannten Neuronen seine Ausgabe über zwei Gewichtsvektoren. Es ist in der Lage, die ihm präsentierten Muster in zwei Mengen einzuteilen. Die Gewichtsvektoren werden in einem Lernprozeß unter Zuhilfenahme der gewichteten Summen der zu trennenden Muster bestimmt.

Die Muster, welche dem Neuron präsentiert werden, sollten bei Verwendung dieses Verfahrens auf den Betrag 1 normiert sein. Einer der Gründe dafür ist, daß Muster, welche ein elementweises Vielfaches eines anderen sind, nicht von diesem unterschieden werden können. Muster, welche verschieden klassifiziert werden sollen, dürfen nur einen relativ geringen Anteil an übereinstimmenden Elementen, welche verschieden von 0 sind, haben. Es hat sich ebenfalls als ungünstig erwiesen, wenn alle von 0 verschiedenen Elemente eines Musters den gleichen Wert besitzten, insbesondere, wenn dieses für jedes Muster gilt.

Arbeitet man nicht mit den Mustern selbst, sondern mit deren Fouriertransformierten, dann umgeht man auf diese Weise praktisch den Großteil der eben erwähnten Probleme.

Das vorgestellte Lernverfahren zeigt, wenn es in der Lage ist, das Neuron anzulernen, gute Konvergenzeigenschaften. Es zeigt jedoch speziell bei der Verwendung der schneller konvergierenden Korrekturvorschriften die Tendenz, den Wert Θ nicht sinnvoll einzustellen.

3 Gradientenabstiegsverfahren

3.1 Untersuchung der originalen Korrekturformeln

Wie bereits im Abschnitt 2.6.2 gezeigt wurde, arbeitet das Lernverfahren mit den Korrekturformeln nach Stufe 6 (und genaugenommen auch 5) nicht wie gewünscht. Da das vorgestellte Beispiel mit Korrekturen nach Stufe 1 angelernt werden konnte, sind offensichtlich die Korrekturformeln der Stufen 5 und 6 nicht gut an die auftretenden Probleme angepaßt. Wie bereits ausgeführt, ist die Skalierung des Korrekturwertes $\Delta\Theta^{(4)}$ mit dem Wert von Θ selbst die Ursache für die Konvergenz von Θ gegen 0. Diese Skalierung wurde in [Koch81] auch tatsächlich eingeführt, um negative Θ-Werte zu verhindern. Dort war es auch sinnvoll, weil für $W(x)$ eine Schwellwertfunktion mit Schwellwert 0 verwendet wurde. Bei Verwendung einer Sigmoiden für $W(x)$ ist diese Notwendigkeit nicht mehr gegeben, sie ist sogar eine hemmende Einschränkung.

Funktion des Parameters Θ

Die Funktion des Parameters Θ wird in den Formeln zur Gewichtsberechnung deutlich:

$$w_n = W(\Theta - q_n), \tag{42}$$

$$w_n = W(\Theta - \frac{1}{q_n}) \tag{43}$$

Stellt man sich die Funktion $W(x)$ als eine Zuordnungsfunktion vor, so ergeben sich die Gewichte praktisch aus den $W(x)$ mit den Argumenten $-q_n$ bzw. $-\frac{1}{q_n}$, wobei Θ einen "Arbeitspunkt" einstellt, was man als eine Verschiebung der Funktion entlang der Abszisse deuten kann. Das Prinzip kann am einfachsten verdeutlicht werden, wenn man für $W(x)$ eine Schwellwertfunktion verwendet :

$$W(x) = \begin{cases} 0 & \text{falls} \quad x < 0 \\ 1 & \text{falls} \quad x \geq 0 \end{cases} \tag{44}$$

Dann bestimmt Θ einen Schwellwert. Ist das Element q_n des Quotientenvektors $\vec{q}$ größer als Θ, so wird das entsprechende Element des Gewichtsvektors $w_n^{(1)}$ 0, sonst ist es 1. Es ist jedoch keine konkrete Aussage für $w_n^{(2)}$ möglich, da aus $q_n > \Theta$ nicht grundsätzlich abgeleitet werden kann, ob $\frac{1}{q_n} > \Theta$ ist, denn dies ist von den Werten selbst abhängig. Betrachtet man den in Abschnitt

2.3 postulierten Zusammenhang, daß eine Erhöhung von Θ eine Verringerung der Q_k zur Folge hat, so zeigt sich, daß dies nicht notwendigerweise der Fall sein muß[5]:

Es sei :

$$q_{i_1} \leq q_{i_2} \leq \ldots \leq q_{i_{n-1}} \leq q_{i_n}. \tag{45}$$

Der Wert Θ sei durch vorherige Lernschritte so eingestellt, daß gilt:

$$q_{i_j} \leq \Theta < q_{i_{j+1}}. \tag{46}$$

Daraus folgt, daß es ein k gibt, so daß:

$$\frac{1}{q_{i_k}} > \Theta \geq \frac{1}{q_{i_{k+1}}} \tag{47}$$

Damit sind j und k die Anzahl der Elemente der Gewichtsvektoren $\vec{w}^{(1)}$ bzw. $\vec{w}^{(2)}$, die den Wert 1 haben. Der Lernschritt modifiziere Θ so, daß gilt :

$$q_{i_{j+1}} \leq \Theta < q_{i_{j+2}} \qquad \text{und damit} \tag{48}$$

$$\frac{1}{q_{i_{k+l}}} > \Theta \geq \frac{1}{q_{i_{k+l+1}}} \tag{49}$$

Somit haben jetzt $j+1$ Elemente des Gewichtsvektors $\vec{w}^{(1)}$ den Wert 1. Es ist jetzt jedoch andere Aussage über l möglich, als daß sein Wert nicht negativ ist. Sein Wert ist einzig und allein durch die Werte der q_n bestimmt. Da in diesem konstruierten Fall nicht einmal eine Aussage über die Änderung des Gewichtsvektors $\vec{w}^{(2)}$ möglich ist, kann auch keine Aussage über das Verhältnis der Veränderung der Skalarprodukte der einzelnen Muster zueinander getroffen werden. (Dies wäre auch nicht möglich, wenn man für l einen konkreten positiven Wert angeben könnte, da es dann noch von dem Muster selbst abhängig wäre). Als Konsequenz dessen ist natürlich auch keine zuverlässige Aussage über die Änderung der Quotienten Q_k oder gar deren Mittelwert $\bar{Q}$ zu finden.

Wird für die Gewichtsfunktion $W(x)$ nicht wie eben beschrieben eine Schwellwertfunktion verwendet, so wird das Problem noch wesentlich schwieriger zu handhaben. Dann hat nämlich eine Änderung von Θ eine Änderung

[5] Dies ist nur für den eingschwungenen Zustand $Q_k = \bar{Q} \forall k$ und somit einer bestimmten Kombination von g_k und unter Annahme der kontinuierlichen Verteilung der Werte q_n beweisbar.

jedes Elements der beiden Gewichtsvektoren zur Folge. Somit ist auch der Abstand der einzelnen q_n untereinenander und der Betrag der Änderung von Θ entscheidend dafür, ob eine Gewichtsänderung $\Delta w_n^{(1)}$ größer oder kleiner als die damit verbundene Gewichtsänderung $\Delta w_n^{(2)}$ ist.

Das Lernverfahren baut in Gleichung (19) auf dem oben dargestellten Zusammenhang, welcher soeben als falsch bewiesen wurde, auf. Die Effekte, die durch dieses Problem auftraten, sollten mit Hilfe der Formeln (35) und (36) eliminiert werden. d_δ dient als Maß für die Verteilung der einzelnen q_n. Es ist nicht auszuschließen, daß diese Heuristik in vielen Fällen recht gut funktioniert, sie ist allerdings nicht allgemeingültig.

Ein weiteres Problem des Lernverfahrens besteht darin, daß die Anzahl der Parameter sehr groß und die Wirkung eines jeden einzelnen nur sehr schwer einzuschätzen ist. Ein sinnvolles Experimentieren erscheint kaum möglich zu sein. Wie am Beispiel des Parameters $\tau^{(5)}$ in Abschnitt 2.6.2 gezeigt wurde, können schon minimale Änderungen eines Parameters extreme Auswirkungen haben.

Im folgenden Abschnitt soll versucht werden, die Änderung der Werte von Θ und g_k auf ein Gradientenabstiegsverfahren zurückzuführen.

3.2 Gradientenabstieg

3.2.1 Grundprinzip von Gradientenverfahren

Prinzipiell berechnet ein Neuron anhand seiner Gewichte eine Ausgabe. Diese Ausgabe stimmt in der Regel nicht völlig mit der gewünschten Ausgabe überein. Die Abweichung, auch als Fehler des Neurons bezeichnet, ist natürlich über die Ausgabe von den Gewichten des Neurons abhängig. Somit kann man den Fehler bei festen Beispielen als eine Funktion der Gewichte auffassen:

$$E(\vec{w}) = E(w_1, w_2, \ldots, w_n) \tag{50}$$

Jedes Lernverfahren für neuronale Netze stellt eine Suche in diesem mehrdimensionalen Raum, welcher durch die Komponenten des Gewichtsvektors aufgespannt wird, dar. Ziel ist es, den Gewichtsvektor so zu bestimmen, daß der zugehörige Fehler minimal wird.

Eine einfache Methode ist es, den Gewichtsvektor jeweils ein wenig in Richtung des negativen Gradienten zu verändern:

$$\Delta\vec{w} = -\eta \nabla E(\vec{w}), \tag{51}$$

woraus sich für ein einzelnes Element des Gewichtsvektors die Änderung

$$\Delta w_i = -\eta \frac{\partial E(\vec{w})}{\partial w_i} \tag{52}$$

ergibt. Dies entspricht einem ständigen Absteigen auf der Oberfläche der Fehlerfunktion.

Als Fehlerfunktion wird meist der quadratische Abstand zwischen der gewünschten und der tatsächlichen Ausgabe verwendet. Der Gesamtfehler ergibt sich aus der Summe der Fehler über alle Muster:

$$E(\vec{w}) = \sum_p E_p(\vec{w}) \quad , \tag{53}$$

$$\text{mit} \quad E_p(\vec{w}) = \frac{1}{2}(t_p - o_p(\vec{w}))^2. \tag{54}$$

Dabei sei $o_p(\vec{w})$ die Ausgabe, die das Neuron mit dem Gewichtsvektor $\vec{w}$ bei angelegtme Muster $\vec{x}_p$ erzeugt, und t_p die eigentlich erwünschte Ausgabe.

Die Lernverfahren kann man prinzipiell in zwei Klassen einteilen:

- **Batch-Lernverfahren:** Die Verfahren arbeiten genau so, wie oben beschrieben. Die Gewichtsänderungen werden nach Formel (51) bestimmt.
- **Online-Lernverfahren:** Statt nach Formel (51) die Gewichtsänderung zu berechnen, wird für ein einzelnes Muster eine Gewichtsänderung nach

  $$\Delta\vec{w} = -\eta \nabla E_p(\vec{w}). \tag{55}$$

  bestimmt. Es werden Muster nacheinander (in einer zufälligen oder festen Reihenfolge) präsentiert und die entsprechenden Gewichtsänderungen sofort ausgeführt. Damit findet natürlich nicht exakt ein Gradientenabstieg auf $E(\vec{w})$ statt. Aus diesem Grunde werden diese Online-Lernverfahren auch als stochastische Gradientenverfahren bezeichnet. Sie haben den Vorteil, daß sie effizienter zu implementieren sind und oft mit einem geringeren Trainingsaufwand das Neuron anlernen können.

3.2.2 Herleitung der Wichtungskorrektur

Analog dem allgemeinen Prinzip wird für des vorgestellte Neuron die Fehlerfunktion definiert als :

$$E_k = \frac{1}{2}(Q^{(S)} - Q_k)^2, \tag{56}$$

wobei die Quotienten für die Muster der zweiten Menge wie in Gleichung (2) als Reziproke des normalen Quotienten definiert sind. An die Stelle des Gewichtsvektors für normale Neuronen tritt ein Vektor, welcher aus Θ und den Musterwichtungen g_k besteht:

$$\vec{w} = \begin{pmatrix} \Theta \\ g_1 \\ g_2 \\ \vdots \\ g_K \end{pmatrix} \tag{57}$$

Somit ergeben sich für die Korrekturen der Komponenten die folgenden Formeln :

$$\Delta\Theta = -\eta \frac{\partial E(\vec{w})}{\partial \Theta}, \tag{58}$$

$$\Delta g_p = -\eta \frac{\partial E(\vec{w})}{\partial g_p}. \tag{59}$$

Ersetzt man den Gesamtfehler E durch die Summe der Fehler der einzelnen Muster, so erhält man:

$$\Delta\Theta = -\eta \sum_k \frac{\partial E_k(\vec{w})}{\partial \Theta}, \tag{60}$$

$$\Delta g_p = -\eta \sum_k \frac{\partial E_k(\vec{w})}{\partial g_p}. \tag{61}$$

Die Ableitungen der Einzelfehler sind mit der Kettenregel noch recht einfach zu handhaben:

$$\frac{\partial E_k(\vec{w})}{\partial \Theta} = \frac{\partial E_k(\vec{w})}{\partial Q_k} \cdot \frac{\partial Q_k}{\partial \Theta}, \tag{62}$$

$$\frac{\partial E_k(\vec{w})}{\partial g_p} = \frac{\partial E_k(\vec{w})}{\partial Q_k} \cdot \frac{\partial Q_k}{\partial g_p}. \tag{63}$$

Mit Gleichung (56) gilt :

$$\frac{\partial E_k(\vec{w})}{\partial Q_k} = -(Q^{(S)} - Q_k) \tag{64}$$

Die Ableitungen der Quotienten nach Θ bzw. g_p ist nicht mehr in einer allgemeinen Formel anzugeben, da die Definition des Quotienten davon abhängt, ob das entsprechende Muster zur ersten oder zur zweiten Menge gehört. Im folgenden bezeichne Q_{p_i} einen zu einem Muster der ersten Menge und Q_{p_j} einen zu einem Muster der zweiten Menge gehörenden Quotienten. Ihre Definitionen ergeben sich nach den Formeln (1) und (2) als:

$$Q_{p_i} = \frac{\vec{w}^{(2)} \cdot \vec{x}_{p_i}}{\vec{w}^{(1)} \cdot \vec{x}_{p_i}} \qquad \text{und} \tag{65}$$

$$Q_{p_j} = \frac{\vec{w}^{(1)} \cdot \vec{x}_{p_j}}{\vec{w}^{(2)} \cdot \vec{x}_{p_j}}. \tag{66}$$

Nach der Definition der Gewichtsvektoren (siehe Gleichungen (12) und (13)) ergibt sich:

$$Q_{p_i} = \frac{\sum\limits_n W\left(\Theta - \frac{1}{q_n}\right) x_{p_i,n}}{\sum\limits_n W\left(\Theta - q_n\right) x_{p_i,n}} \qquad \text{und} \tag{67}$$

$$Q_{p_j} = \frac{\sum\limits_n W\left(\Theta - q_n\right) x_{p_j,n}}{\sum\limits_n W\left(\Theta - \frac{1}{q_n}\right) x_{p_j,n}}. \tag{68}$$

Dabei bezeichnet $x_{p_i,n}$ das n-te Element des Mustervektors $\vec{x}_{p_i}$. Mit der Definition des elementweisen Quotienten der gewichteten Mustermittelwerte $\vec{q}$ (siehe Gleichungen (9) bis (11)) ergibt sich:

$$q_n = \frac{K_1}{K_2} \cdot \frac{\sum_j g_j x_{j,n}}{\sum_i g_i x_{i,n}} \tag{69}$$

Daraus folgt letztendlich:

$$Q_{p_i} = \frac{\sum\limits_n W\left(\Theta - \frac{K_2}{K_1} \cdot \frac{\sum_i g_i x_{i,n}}{\sum_j g_j x_{j,n}}\right) x_{p_i,n}}{\sum\limits_n W\left(\Theta - \frac{K_1}{K_2} \cdot \frac{\sum_j g_j x_{j,n}}{\sum_i g_i x_{i,n}}\right) x_{p_i,n}} \qquad \text{und} \tag{70}$$

$$Q_{p_j} = \frac{\sum\limits_n W\left(\Theta - \frac{K_1}{K_2} \cdot \frac{\sum_j g_j x_{j,n}}{\sum_i g_i x_{i,n}}\right) x_{p_j,n}}{\sum\limits_n W\left(\Theta - \frac{K_2}{K_1} \cdot \frac{\sum_i g_i x_{i,n}}{\sum_j g_j x_{j,n}}\right) x_{p_j,n}}. \tag{71}$$

Auf der Basis dieser Formeln sind nun die Ableitungen $\frac{\partial Q_k}{\partial \Theta}$ und $\frac{\partial Q_k}{\partial g_p}$ zu berechnen.

1. Ableitungen nach Θ

Nach der Quotientenregel

$$\frac{\partial \frac{f_1(x)}{f_2(x)}}{\partial x} = \frac{\frac{\partial f_1(x)}{\partial x} f_2(x) - f_1(x) \frac{\partial f_2(x)}{\partial x}}{f_2(x)^2}$$

ergibt sich mit

$$\frac{\partial \sum_n W\left(\Theta - \frac{1}{q_n}\right) x_{p_k,n}}{\partial \Theta} = \sum_n W'\left(\Theta - \frac{1}{q_n}\right) x_{p_k,n} \qquad \text{und} \tag{72}$$

$$\frac{\partial \sum_n W\left(\Theta - q_n\right) x_{p_k,n}}{\partial \Theta} = \sum_n W'\left(\Theta - q_n\right) x_{p_k,n} \tag{73}$$

nach Gleichung (67) die Ableitung als:

$$\frac{\partial Q_{p_i}}{\partial \Theta} = \frac{\sum_n W'\left(\Theta - \frac{1}{q_n}\right) x_{p_i,n} \cdot \sum_n W(\Theta - q_n) x_{p_i,n} - \sum_n W\left(\Theta - \frac{1}{q_n}\right) x_{p_i,n} \cdot \sum_n W'(\Theta - q_n) x_{p_i,n}}{\left(\sum_n W(\Theta - q_n) x_{p_i,n}\right)^2}, \tag{74}$$

oder kurz :

$$\frac{\partial Q_{p_i}}{\partial \Theta} = \frac{\sum_n W'\left(\Theta - \frac{1}{q_n}\right) x_{p_i,n} \cdot \vec{w}^{(1)} \cdot \vec{x}_{p_i} - \cdot \vec{w}^{(2)} \cdot \vec{x}_{p_i} \cdot \sum_n W'(\Theta - q_n) x_{p_i,n}}{\left(\vec{w}^{(1)} \cdot \vec{x}_{p_i}\right)^2}, \tag{75}$$

Für Muster der zweite Mustermenge ergibt sich nach (68) unter Verwendung von (72) und (73) ganz analog:

$$\frac{\partial Q_{p_j}}{\partial \Theta} = \frac{\sum_n W'(\Theta - q_n) x_{p_j,n} \cdot \sum_n W\left(\Theta - \frac{1}{q_n}\right) x_{p_j,n} - \sum_n W(\Theta - q_n) x_{p_j,n} \cdot \sum_n W'\left(\Theta - \frac{1}{q_n}\right) x_{p_j,n}}{\left(\sum_n W\left(\Theta - \frac{1}{q_n}\right) x_{p_j,n}\right)^2}, \tag{76}$$

oder kurz :

$$\frac{\partial Q_{p_j}}{\partial \Theta} = \frac{\sum_n W'(\Theta - q_n) x_{p_j,n} \cdot \vec{w}^{(2)} \cdot \vec{x}_{p_j} - \cdot \vec{w}^{(1)} \cdot \vec{x}_{p_j} \cdot \sum_n W'\left(\Theta - \frac{1}{q_n}\right) x_{p_j,n}}{\left(\vec{w}^{(2)} \cdot \vec{x}_{p_j}\right)^2}, \tag{77}$$

2. Ableitungen nach g_n

Auch hier wird die Ableitung der Skalarprodukte nach g_p benötigt.

$$\frac{\partial \sum\limits_n W\left(\Theta - \frac{1}{q_n}\right) x_{i,n}}{\partial g_p} = -\sum_n W'\left(\Theta - \frac{1}{q_n}\right) \frac{\partial \frac{1}{q_n}}{\partial g_p} x_{i,n} \qquad \text{und} \tag{78}$$

$$\frac{\partial \sum\limits_n W\left(\Theta - q_n\right) x_{i,n}}{\partial g_p} = -\sum_n W'\left(\Theta - q_n\right) \frac{\partial q_n}{\partial g_p} x_{i,n} \tag{79}$$

Mit der Definition der q_n (siehe Gleichung (69)) ergibt sich für die g_p, deren zugehöriges Muster zu der ersten Klasse gehört:

$$\begin{aligned}
\frac{\partial \frac{1}{q_n}}{\partial g_{p_i}} &= \frac{\partial \frac{K_2}{K_1} \cdot \frac{\sum_i g_i x_{i,n}}{\sum_j g_j x_{j,n}}}{\partial g_{p_i}} && (80)\\
&= \frac{K_2}{K_1} \cdot \frac{\partial \frac{g_{p_i} x_{p_i,n}}{\sum_j g_j x_{j,n}}}{\partial g_{p_i}} && (81)\\
&= \frac{K_2}{K_1} \cdot \frac{x_{p_i,n}}{\sum_j g_j x_{j,n}} \qquad \text{, kurz :} && (82)\\
&= \frac{K_2}{K_1} \cdot \frac{x_{p_i,n}}{x_n^{(2)}} \qquad \text{und} && (83)\\
\frac{\partial q_n}{\partial g_{p_i}} &= \frac{\partial \frac{K_1}{K_2} \cdot \frac{\sum_j g_j x_{j,n}}{\sum_i g_i x_{i,n}}}{\partial g_{p_i}} && (84)\\
&= \frac{K_1}{K_2} \cdot \frac{\frac{\partial \sum_j g_j x_{j,n}}{\partial g_{p_i}} \sum_i g_i x_{i,n} - \sum_j g_j x_{j,n} \frac{\partial \sum_i g_i x_{i,n}}{\partial g_{p_i}}}{\left(\sum_i g_i x_{i,n}\right)^2} && (85)\\
&= -\frac{K_1}{K_2} \cdot \frac{x_{p_i,n} \sum_j g_j x_{j,n}}{\left(\sum_i g_i x_{i,n}\right)^2} \qquad \text{, kurz :} && (86)\\
&= -\frac{K_1}{K_2} \cdot \frac{x_{p_i,n} x_n^{(2)}}{\left(x_n^{(1)}\right)^2}. && (87)
\end{aligned}$$

Für die g_p, deren zugehörige Muster der anderen Mustermenge angehören, ergibt sich:

$$\frac{\partial \frac{1}{q_n}}{\partial g_{p_j}} = \frac{\partial \frac{K_2}{K_1} \cdot \frac{\sum_i g_i x_{i,n}}{\sum_j g_j x_{j,n}}}{\partial g_{p_j}} \tag{88}$$

$$= \frac{K_2}{K_1} \cdot \frac{\frac{\partial \sum_i g_i x_{i,n}}{\partial g_{p_j}} \sum_j g_j x_{j,n} - \sum_i g_i x_{i,n} \frac{\partial \sum_j g_j x_{j,n}}{\partial g_{p_j}}}{\left(\sum_j g_j x_{j,n}\right)^2} \tag{89}$$

$$= -\frac{K_2}{K_1} \cdot \frac{x_{p_j,n} \sum_i g_i x_{i,n}}{\left(\sum_j g_j x_{j,n}\right)^2} \quad \text{, kurz :} \tag{90}$$

$$= -\frac{K_2}{K_1} \cdot \frac{x_{p_j,n} x_n^{(1)}}{\left(x_n^{(2)}\right)^2}. \quad \text{und} \tag{91}$$

$$\frac{\partial q_n}{\partial g_{p_j}} = \frac{\partial \frac{K_1}{K_2} \cdot \frac{\sum_j g_j x_{j,n}}{\sum_i g_i x_{i,n}}}{\partial g_{p_j}} \tag{92}$$

$$= \frac{K_1}{K_2} \cdot \frac{\partial \frac{g_{p_j} x_{p_j,n}}{\sum_i g_i x_{i,n}}}{\partial g_{p_j}} \tag{93}$$

$$= \frac{K_1}{K_2} \cdot \frac{x_{p_j,n}}{\sum_i g_i x_{i,n}} \quad \text{, kurz :} \tag{94}$$

$$= \frac{K_1}{K_2} \cdot \frac{x_{p_j,n}}{x_n^{(1)}} \tag{95}$$

Für die Ableitung der Quotienten für Muster der ersten Menge ergibt sich:

$$\frac{\partial Q_{p_i}}{\partial g_p} = \frac{\sum_n W'\left(\Theta - \frac{1}{q_n}\right) \cdot \frac{\partial \frac{1}{q_n}}{\partial g_p} x_{p_i,n} \cdot \sum_n W(\Theta - q_n) x_{p_i,n} - \sum_n W\left(\Theta - \frac{1}{q_n}\right) x_{p_i,n} \cdot \sum_n W'(\Theta - q_n) \cdot \frac{\partial q_n}{\partial g_p} x_{p_i,n}}{\left(\sum_n W(\Theta - q_n) x_{p_i,n}\right)^2}, \tag{96}$$

oder kurz :

$$\frac{\partial Q_{p_i}}{\partial g_p} = \frac{\sum_n W'\left(\Theta - \frac{1}{q_n}\right) \cdot \frac{\partial \frac{1}{q_n}}{\partial g_p} x_{p_i,n} \cdot \vec{w}^{(1)} \cdot \vec{x}_{p_i} - \cdot \vec{w}^{(2)} \cdot \vec{x}_{p_i} \cdot \sum_n W'(\Theta - q_n) \cdot \frac{\partial \frac{1}{q_n}}{\partial g_p} x_{p_i,n}}{\left(\vec{w}^{(1)} \cdot \vec{x}_{p_i}\right)^2}, \tag{97}$$

Analog ergibt sich für Muster der zweiten Menge:

$$\frac{\partial Q_{p_j}}{\partial g_p} = \frac{\sum_n W'(\Theta - q_n) \cdot \frac{\partial q_n}{\partial g_p} x_{p_j,n} \cdot \sum_n W\left(\Theta - \frac{1}{q_n}\right) x_{p_j,n} - \sum_n W(\Theta - q_n) x_{p_j,n} \cdot \sum_n W'\left(\Theta - \frac{1}{q_n}\right) \cdot \frac{\partial \frac{1}{q_n}}{\partial g_p} x_{p_j,n}}{\left(\sum_n W\left(\Theta - \frac{1}{q_n}\right) x_{p_i,n}\right)^2}, \tag{98}$$

oder kurz :

$$\frac{\partial Q_{p_j}}{\partial g_p} = \frac{\sum_n W'(\Theta - q_n) \cdot \frac{\partial q_n}{\partial g_p} x_{p_j,n} \cdot \vec{w}^{(2)} \cdot \vec{x}_{p_j} - \cdot \vec{w}^{(1)} \cdot \vec{x}_{p_j} \cdot \sum_n W'\left(\Theta - \frac{1}{q_n}\right) \cdot \frac{\partial \frac{1}{q_n}}{\partial g_p} x_{p_j,n}}{\left(\vec{w}^{(2)} \cdot \vec{x}_{p_i}\right)^2}, \tag{99}$$

Wird als Gewichtsfunktion $W(x)$ die Sigmoide verwendet, so ergibt sich ihre Ableitung als:

$$\frac{\partial W(x)}{\partial x} = \frac{\partial \frac{1}{1+e^{-x}}}{\partial x} \tag{100}$$
$$= -\frac{e^{-x}}{(1+e^{-x})^2} \tag{101}$$

Zusammengefaßt ergeben sich die Korrekturen der Werte von Θ und g_k als:

$$\Delta\Theta = \eta \sum_k (Q^{(S)} - Q_k) \frac{\partial Q_k}{\partial \Theta} \qquad \text{und} \tag{102}$$
$$\Delta g_p = \eta \sum_k (Q^{(S)} - Q_k) \frac{\partial Q_k}{\partial g_p}. \tag{103}$$

Je nachdem, ob des Muster k zur ersten oder zweiten Menge gehört, muß die entsprechende Ableitung $\frac{\partial Q_{p_i}}{\partial \Theta}$ oder $\frac{\partial Q_{p_j}}{\partial \Theta}$ bzw. $\frac{\partial Q_{p_i}}{\partial g_p}$ oder $\frac{\partial Q_{p_j}}{\partial g_p}$ ausgewählt werden. In den Ableitungen $\frac{\partial Q_i}{\partial g_p}$ und $\frac{\partial Q_j}{\partial g_p}$ ist je nachdem, ob das Muster p zur ersten oder zweiten Menge gehört, für $\frac{\partial q_n}{\partial g_p}$ bzw. $\frac{\partial \frac{1}{q_n}}{\partial g_p}$ wiederum $\frac{\partial q_n}{\partial g_{p_i}}$ oder $\frac{\partial q_n}{\partial g_{p_j}}$ bzw. $\frac{\partial \frac{1}{q_n}}{\partial g_{p_i}}$ oder $\frac{\partial \frac{1}{q_n}}{\partial g_{p_j}}$ einzusetzen.

3.3 Implementation

Die Implementation des Gradientenabstiegsverfahrens brachte leider keine zufriedenstellenden Ergebnisse. Bei einer Lernrate von 0.1 oder größer stieg der Gesamtfehler nach einem kurzen Abfall extrem an. Bei sehr kleinen Lernraten war die Konvergenz so schlecht, daß auch nach mehreren Stunden noch kein befriedigendes Ergebnis erreicht wurde.

Die Ursachen für dieses Verhalten sind noch unklar. Da die Formeln und deren Implementation mehrfach überprüft wurden, ist ein Herleitungs- oder Programmierfehler relativ unwahrscheinlich. Eine sinnvolle Variante, dieses Problem zu untersuchen, wäre die Arbeit mit wenigen, niedrigdimensionalen Mustern.

4 Differenzen-Neuron

4.1 Struktur des Differenzen-Neurons

Wie bereits in Abschnitt 2.2 erwähnt, kann man statt dem Quotienten der beiden Aktivierungen $net^{(1)}$ und $net^{(2)}$ auch die Differenz der beiden verwenden:

$$D_k = net^{(2)}(\vec{x}_k) - net^{(1)}(\vec{x}_k) \tag{104}$$

Dann ist es jedoch sinnvoll, die Definition der Gewichtsvektoren ebenfalls zu ändern:

$$w_n^{(1)} = W(\Theta - d_n), \tag{105}$$
$$w_n^{(2)} = W(\Theta + d_n), \tag{106}$$

wobei die d_n die Elemente des Differenzvektors

$$\vec{d} = \vec{x}^{(2)} - \vec{x}^{(1)} \tag{107}$$

aus den beiden gewichteten Mustermittelwerten sind.

Es ist offensichtlich, daß hier die Unterscheidung der beiden Gewichtsvektoren nicht mehr zwingend notwendig ist. Sie können durch einen Gewichtsvektor ersetzt werden, der der Differenz aus den beiden oben beschrieben Gewichtsvektoren entspricht:

$$\vec{w} = \vec{w}^{(2)} - \vec{w}^{(1)} \quad \text{, mit} \tag{108}$$
$$w_n = W(\Theta + d_n) - W(\Theta - d_n). \tag{109}$$

Die Aktivierung des Neurons ergibt sich dann als

$$net(\vec{x}) = \vec{w} \cdot \vec{x}. \tag{110}$$

Im weiteren werden aber trotzdem die beiden Gewichtsvektor $\vec{w}^{(1)}$ und $\vec{w}^{(2)}$ verwendet, um die Parallelen zum Quotienten-Neuron zu verdeutlichen.

Für die Lernphase ist es wieder sinnvoll, die Differenz für die Muster der verschiedenen Klassen unterschiedlich zu definieren:

$$D_i = net^{(2)}(\vec{x}_i) - net^{(1)}(\vec{x}_i), \tag{111}$$
$$D_j = net^{(1)}(\vec{x}_j) - net^{(2)}(\vec{x}_j). \tag{112}$$

Analog zum Sollquotienten $Q^{(S)}$ wird eine Solldifferenz $D^{(S)}$ eingeführt.

Die in [Koch96] vorgestellten Korrekturformeln sind auf die Quotientenbildung zugeschnitten, sie sind bei dem Differenzen-Neuron ohne Anpassung nicht mehr einsetzbar. Für die Korrektur der Musterwichtungen g_k soll im folgenden Abschnitt ein Gradientenabstiegsverfahren eingesetzt werden.

4.2 Gradientenabstiegsverfahren für das D-Neuron

Der Ansatz kann hier analog dem Gradientenabstiegsverfahren für das Q-Neuron in Abschnitt 3.2 gewählt werden. In den Formeln (56) bis (64) kann $Q^{(S)}$ durch $D^{(S)}$ und Q_k durch D_k ersetzt werden.

Die Differenzen für die zu der ersten bzw. zweiten Mustermenge gehörenden Muster ergeben sich unter Verwendung aller Definitionen als:

$$\begin{aligned} D_{p_i} &= \sum_n W\left(\Theta + \frac{1}{K_2}\sum\nolimits_j g_j x_{j,n} - \frac{1}{K_1}\sum\nolimits_i g_i x_{i,n}\right) x_{p_i,n} \\ &\quad - \sum_n W\left(\Theta - \frac{1}{K_2}\sum\nolimits_j g_j x_{j,n} + \frac{1}{K_1}\sum\nolimits_i g_i x_{i,n}\right) x_{p_i,n} \text{ und} \qquad (113) \\ D_{p_j} &= \sum_n W\left(\Theta - \frac{1}{K_2}\sum\nolimits_j g_j x_{j,n} + \frac{1}{K_1}\sum\nolimits_i g_i x_{i,n}\right) x_{p_j,n} \\ &\quad - \sum_n W\left(\Theta + \frac{1}{K_2}\sum\nolimits_j g_j x_{j,n} - \frac{1}{K_1}\sum\nolimits_i g_i x_{i,n}\right) x_{p_j,n} \qquad (114) \end{aligned}$$

Die Ableitungen nach Θ ergeben sich (etwas verkürzt geschrieben) als:

$$\begin{aligned} \frac{\partial D_{p_i}}{\partial \Theta} &= \sum_n W'\left(\Theta + d_n\right) x_{p_i,n} - \sum_n W'\left(\Theta - d_n\right) x_{p_i,n} \quad \text{und} \qquad (115) \\ \frac{\partial D_{p_j}}{\partial \Theta} &= \sum_n W'\left(\Theta - d_n\right) x_{p_j,n} - \sum_n W'\left(\Theta + d_n\right) x_{p_j,n} \qquad (116) \end{aligned}$$

Bei den Ableitungen nach g_p muß wieder unterschieden werden, ob das g_p zu einem Muster der ersten oder der zweiten Mustermenge gehört. Sie ergeben sich (wiederum etwas verkürzt geschrieben) als:

$$\frac{\partial D_{p_i}}{\partial g_{m_i}} = \sum_n W'\left(\Theta + d_n\right) \frac{1}{K_1}(-x_{m_i,n}) x_{p_i,n}$$

$$- \sum_n W'(\Theta - d_n) \frac{1}{K_1} x_{m_i,n} x_{p_i,n} \quad , \tag{117}$$

$$\frac{\partial D_{p_j}}{\partial g_{m_j}} = \sum_n W'(\Theta - d_n) \frac{1}{K_1} x_{m_i,n} x_{p_j,n}$$

$$- \sum_n W'(\Theta + d_n) \frac{1}{K_1} (-x_{m_i,n}) x_{p_j,n} \quad , \tag{118}$$

$$\frac{\partial D_{p_i}}{\partial g_{m_i}} = \sum_n W'(\Theta + d_n) \frac{1}{K_2} x_{m_j,n} x_{p_i,n}$$

$$- \sum_n W'(\Theta - d_n) \frac{1}{K_2} (-x_{m_j,n}) x_{p_i,n} \quad \text{und} \tag{119}$$

$$\frac{\partial D_{p_j}}{\partial g_{m_j}} = \sum_n W'(\Theta - d_n) \frac{1}{K_2} (-x_{m_j,n}) x_{p_j,n}$$

$$- \sum_n W'(\Theta + d_n) \frac{1}{K_2} x_{m_j,n} x_{p_j,n} \quad . \tag{120}$$

Mit diesen Formeln kann wieder zusammengefaßt die Korrekturvorschrift angegeben werden als:

$$\Delta\Theta = \eta \sum_k (D^{(S)} - D_k) \frac{\partial D_k}{\partial \Theta} \quad \text{und} \tag{121}$$

$$\Delta g_p = \eta \sum_k (D^{(S)} - D_k) \frac{\partial D_k}{\partial g_p}. \tag{122}$$

Auch hier sind wieder entsprechend der Mengenzugehörigkeit der Muster die entsprechenden Ableitungen aus den Formeln (115) bis (120) einzusetzen.

4.3 Implementation

Auch für das D-Neuron konnte das Gradientenabstiegsverfahren nicht erfolgreich implementiert werden. Es traten ähnliche Probleme wie beim Q-Neuron auf. Auch hier wäre eine weitere Untersuchung angebracht.

5 Anlernen eines Netzwerkes

5.1 Einführung und Zielsetzung

In diesem Abschnitt soll untersucht werden, wie ein neuronales Netz, welches zumindest teilweise aus Q-Neuronen besteht, angelernt werden kann.

Ziel ist es, ein Netz zu erhalten, welches eine ähnliche Struktur wie das in 2.2 beschriebe Netz hat. Seine verdeckte Schicht soll aus Q-Neuronen und seine Ausgabeschicht aus Perzeptrons bestehen. Es soll jedoch nicht wie dort angegeben direkt in Abhängigkeit von der Aufgabe konstruiert werden.

Die einzigen Vorgaben, welche in die Netzstruktur eingehen, sollen die Anzahl der verdeckten und die Anzahl der Ausgabeneuronen sein. Die Neuronen seien voll vernetzt. Für jedes anzulernende Muster gibt es eine gewünschte Ausgabe. Es wird jedoch nicht festgelegt, welche verdeckten Neuronen welche Muster wie klassifizieren sollen. Dieses soll sich durch das Lernen selbst ergeben.

5.2 Lösungsansatz

Durch die Struktur und die Verhaltensweise des Q-Neurons und das für ein solches Neuron verwendete IWD-Lernverfahren ergeben sich bei der Umsetzung in ein Netzwerk eine Reihe von Problemen:

Quotient:

Durch die Bildung des Quotienten kommt es zu einer Asymmetrie im Wertebereich der Neuronenausgaben. Wurde das Neuron z.B. mit einem Wert von $Q^{(S)} = 0.6$ angelernt, so liegen seine Ausgaben für Muster der ersten Menge im Intervall $[0, 0.6]$. Für Muster der zweiten Menge ist nur sicher, daß die Ausgabe größer als $\frac{1}{0.6}$, also $1.\bar{6}$ ist. Wenn man ein Muster betrachtet, das sehr sicher erkannt wird, weil sein Quotient nach der Definition des Quotienten für die Trainingsphase z.B. etwa bei 0.2 liegt, so sieht man, daß seine Ausgabe um $0.6 - 0.2 = 0.4$ vom Sollwert abweicht, wenn es zur ersten Menge gehört. Gehört es jedoch zur zweiten Menge, so beträgt die Abweichung vom Sollwert $\frac{1}{0.2} - \frac{1}{0.6} = 3.\bar{3}$. Es ist zu vermuten, daß diese Ungleichheit die Eigenschaften des Lernverfahrens negativ beeinflußt. Sie kann eliminiert werden, wenn man als Ausgabefunktion der Q-Neuronen eine Schwellwertfunktion mit Schwellwert 1 verwendet:

$$f_{out}(Q) = \begin{cases} -1 & \text{falls} \quad Q \leq 1 \\ 1 & \text{falls} \quad Q > 1 \end{cases} \tag{123}$$

Bilden der Lernmengen:

Im Lernverfahren für ein einzelnes Q-Neuron wird davon ausgegangen, daß zwei Lernmengen definiert sind, welche das Neuron unterscheiden soll. In einem Netzwerk existieren solche Zuordnungen jedoch nicht a priori. Des weiteren ist es jedoch schwierig, eine solche Zuordnung zu schaffen.

Ein Möglichkeit bestünde darin, aus der gewünschten Ausgabe eines Ausgabeneurons und seinen Gewichten zu bestimmen, ob das jeweilige verdeckte Neuron eine 1 oder eine -1 ausgeben und somit das Muster zur ersten oder zur zweiten Lernmenge gehören sollte. Der genaue Zusammenhang ist in Tabelle 4 dargestellt.

Gewünschte Ausgabe von O_i	Gewicht w_{ij}	Gewünschte Ausgabe von H_j	Mustermenge in H_j
1	> 0	1	2
1	< 0	-1	1
0	> 0	-1	1
0	> 0	1	2

Tabelle 4: *Zuordung des Musters zur Mustermenge in H_j*

Bei dieser Vorgehensweise entstehen leider zwei wesentliche Probleme:

- Es kann zu Inkonsitsenzen kommen. Bsp.: Die gewünschte Ausgabe der Neuronen O_1 und O_2 seien 1 und 0. Das Gewicht w_{11} sei positiv und das Gewicht w_{21} negativ. Somit ergibt sich für das Neuron H_1 einerseits die Forderung, das Muster soll zur zweiten Menge gehören und andererseits die Forderung, es soll zur ersten gehören. Diese Inkonsistenzen würden zwar im Laufe des Lernens durch die Korrekturen der Gewichte zwischen der verdeckten Schicht und der Ausgabeschicht verschwinden, jedoch ist, solange sie noch existieren, ihre Auflösung nötig. Dies könnte geschehen, indem man nicht die oben angegebenen Tabelle verwendet, sondern statt dessen wie bei Backpropagation die Fehler der Ausgabeschicht rückwärts durch die Netzstruktur propagiert.

- Auf diese Art würde für jedes Neuron für jedes Muster festgelegt, wie es zu klassifizieren ist. Das bedeutet, daß die Vereinigung der beiden zu unterscheidenden Mustermengen stets der gesamten Lernmenge entspricht. Dies ist eine Aufgabe, die von einem Q-Neuron nicht sinnvoll gelöst werden kann.

Generell stellt die Bestimmung der Lernmengen für die verdeckten Neuronen das wahrscheinlich schwierigste Problem dar. Es ist zu erwarten, daß man ohne vorherige Einschränkung der Netzstruktur kein vernüftiges Verfahren angeben kann, welches für ein Q-Neuron zwei genügend kleine Lernmengen bestimmt.

Außerdem hätte das grundsätzliche Vorgehen :

1. Gewichte w_{ij} (zwischen verdeckter Schicht und Ausgabeschicht) festlegen
2. Lernmengen bestimmen
3. Q-Neuronen anlernen
4. Gesamte Lernmenge erweitern
5. Wieder von vorn

den Nachteil, daß die Initialisierung der Gewichte w_{ij} entscheidende Bedeutung hätte. Eine sinnvolle Korrektur derselben ist nicht möglich. Denn nachdem die Q-Neuronen angelernt wurden, liefern alle Ausgabeneuronen prinzipiell die gewünschten Ausgaben. Ein Lernen der Verbindungsgewichte w_{ij} bringt keine qualitativen Veränderungen.

Um in beiden Ebenen gleichermaßen sinnvolles Lernen durchführen zu können, müßten sich die einzelnen Lernschritte für Ausgabeneuronen und verdeckte Neuronen immer abwechseln. Dann ist jedoch das IWD-Lernverfahren mit seiner schrittweisen Erweiterung der aktiven Lernmengen nicht mehr direkt einsetzbar.

Das letztere stellt kein echtes Problem dar. In Experimenten hat sich herausgestellt, daß das IWD-Lernverfahren in der Regel in der Lage ist, auch ohne die schrittweise Erweiterung der aktiven Lernmengen zu arbeiten. Dazu wurden die aktiven Lernmengen von Beginn an mit den gesamten Lernmengen gleichgesetzt. Um ein vollständiges Lernen der anfangs gesetzten aktiven Lernmengen zu erreichen, dürfen Muster, deren Wichtung g_k im Verlaufe des Lernprozesses 0 wird, nicht aus der aktiven Lernmenge entfernt werden. Das Lernen auf diese Weise benötigte etwas mehr innere Schleifen, als das Lernen mit schrittweiser Erweiterung insgesamt in allen äußeren Schleifen. Beim Lernen des zu Tabelle 2 gehörenden Beispiels wurden so 541 statt vorher insgesamt 426 innere Schleifen benötigt. Der Grund dafür ist offensichtlich, daß, in den einzelnen inneren Schleifen des erweiternden Verfahrens meist mit einer relativ guten Ausgangsposition gestartet werden kann.

Als nächstes soll die Gewichtsberechnung für ein Q-Neuron etwas genauer betrachtet werden. Die Wichtungen für die einzelnen Muster im Korrekturschritt T ergeben sich als

$$g_k(T) = g_k(0) + \sum_{t=1}^{T} \Delta g_k(t). \tag{124}$$

Somit kann man die Hilfsmuster $\vec{x}^{(1)}$ und $\vec{x}^{(2)}$ ausdrücken als

$$\vec{x}^{(1)} = \sum_i \left(g_i(0) + \sum_{t=1}^{T} \Delta g_i(t) \right) \cdot \vec{x}_i \quad \text{und} \tag{125}$$

$$\vec{x}^{(2)} = \sum_j \left(g_j(0) + \sum_{t=1}^{T} \Delta g_j(t) \right) \cdot \vec{x}_j. \tag{126}$$

Mit $g_k(0) = 1$ und etwas Umstellung ergibt sich :

$$\vec{x}^{(1)} = \sum_{t=1}^{T} \sum_i (\Delta g_i(t) \cdot \vec{x}_i) + \sum_i \vec{x}_i \quad \text{und} \tag{127}$$

$$\vec{x}^{(2)} = \sum_{t=1}^{T} \sum_j (\Delta g_j(t) \cdot \vec{x}_j) + \sum_j \vec{x}_j. \tag{128}$$

Mann kann also die Vektoren $\vec{x}^{(1)}$ und $\vec{x}^{(2)}$ mit $\sum_i \vec{x}_i$ bzw. $\sum_j \vec{x}_j$ initialisieren und dann die sich in den Lernschritten ergebenden Änderungen $\Delta g_k(t) \cdot \vec{x}_k$ aufsummieren.

Mit Hilfe dieser Erkenntnis kann man nun versuchen, die Änderungen Δg_k so zu gestalten, daß sie 0 sind, wenn das entsprechende Muster $\vec{x}_k$ nicht in einer der beiden aktiven Lernmengen des Q-Neurons enthalten ist. Dann verhält sich (abgesehen von der Initialisierung) das Verfahren weiterhin wie das originale. Die Initialisierung für das Neuron kann jedoch nicht sinnvoll bestimmt werden, denn um dies zu tun, müßte bereits am Anfang bekannt sein, welches Q-Neuron welche Unterscheidungsleistung erbringen soll. Eine Initialisierung mit $\vec{0}$ ist nicht möglich. Dann wären sämtliche Quotienten 1 und somit alle $\Delta Q_k = 0$, was zu $\Delta g_k = 0$ führt und somit $\vec{x}^{(1)}$ und $\vec{x}^{(2)}$ nicht verändert. Es bleiben eigentlich zwei Möglichkeiten: Initialisierung mit (kleinen) zufälligen Werten, welche nach einer gewissen Zeit nicht mehr ins Gewicht fallen, oder man wählt sich für die Berechnung von $\vec{x}^{(1)}$ und $\vec{x}^{(2)}$ eine beliebige Verteilung der Muster in die Lernmengen. Die zweite Variante ist jedoch von der ersten nicht weit entfernt.

Für das Anlernen der Perzeptrons in der Ausgabeschicht kann ein normales Backpropagation-Verfahren, wie es z.B. in [Zell94] dargestellt ist, verwendet werden. Es ergibt sich für die Änderung des Gewichts w_{ij} zwischen verdecktem Neuron H_j und Ausgabeneuron O_j:

$$\Delta w_{ij} = \eta \sum_p o_{pj} f'_{act}(net_{pi})(t_{pi} - o_{pi}), \tag{129}$$

wobei o_{pj} und o_{pi} die tatsächliche Ausgaben der Neuronen H_j und O_i und t_{pi} die erwartete Ausgabe von O_i bei angelegtem Muster p sind. f_{act} ist die Aktivierungsfunktion des Perzeptrons, net_{pi} seine Netzeingabe. Analog zum Backpropagation-Verfahren kann man für die Q-Neuronen ein abgewandeltes Fehlersignal berechnen:

$$\begin{aligned} \delta_{pj} &= \sum_i w_{ij}\delta_{pi} \\ &= \sum_i w_{ij} f'_{act}(net_{pi})(t_{pi} - o_{pi}) \end{aligned} \tag{130}$$

Wie sind diese Werte zu interpretieren? $(t_{pi} - o_{pi})$ ist die Abweichung der Ausgabe von O_i vom Sollwert. Durch $f'_{act}(net_{pi})$ wird dem Verlauf der Aktivierungsfunktion Rechnung getragen, o.B.d.A. soll hier der Fall, daß $f_{act}(x)$ die Sigmoide und somit ihre Ableitung immer positiv sei, betrachtet werden. w_{ij} stellt die Verbindung zwischen den Schichten her. Ist $(t_{pi} - o_{pi}) > 0$ und w_{ij} ebenfalls, so ergibt sich $\delta_{pj} > 0$, was bedeutet, daß die Ausgabe des Neurons H_j größer werden sollte, damit die Aktivierung von O_i und somit seine Ausgabe ansteigt und sich dem Sollwert nähert. Die weiteren Zusammenhänge ergeben sich analog, sie sind in Tabelle 5 dargestellt. Die Summenbildung über alle Neuronen der Ausgabeschicht stellt eine Abwägung des Korrektureffektes auf den bei diesem Muster entstehenden Fehler über alle Ausgabeneuronen dar.

In Anbetracht dessen, daß die Ausgabefunktion der Q-Neuronen als Schwellwertfunktion gewählt wurde, gibt es nicht viele Möglichkeiten, den

Abweichung	Gewicht	Fehlersignal	Korrekturziele		
$t_{pi} - o_{pi}$	w_{ij}	$\delta_p j$	o_{pj}	net_{pi}	o_{pi}
> 0	> 0	> 0	$\uparrow$	$\uparrow$	$\uparrow$
> 0	< 0	< 0	$\downarrow$	$\uparrow$	$\uparrow$
< 0	> 0	< 0	$\downarrow$	$\downarrow$	$\downarrow$
< 0	< 0	> 0	$\uparrow$	$\downarrow$	$\downarrow$

Tabelle 5: *Zusammenhang von Fehlern, Gewichten und notwendigen Korrekturen*

Ausgabewert zu korrigieren, er kann nur die Werte -1 oder 1 annehmen. Die einzige echte Veränderung ist also der Sprung über diese Grenze. Dieser erfolgt, wenn der Quotient, den das Q-Neuron bildet, den Wert 1 überspringt. Da es Ziel des Lernens für ein Q-Neuron ist, $Q_k < Q^{(S)}$ zu erreichen, wenn das Muster zur ersten Menge und $Q_k > \frac{1}{Q^{(S)}}$, wenn es zur zweiten Menge gehört, kann dieses Ziel letztendlich nur erreicht werden, wenn das Muster in der entsprechenden entahlten Mustermenge ist. Man kann die folgenden Zusammenhänge feststellen:

- Wenn $\delta_{pj} > 0$ ist, soll o_{pj} erhöht werden. Das bedeutet, daß o_{pj} den Wert 1 annehmen soll. Somit muß das Muster p für das entsprechende Neuron als Mitglied der zweiten Mustermenge betrachtet werden.
- Wenn $\delta_{pj} < 0$ ist, soll o_{pj} verringert werden. Das bedeutet, daß o_{pj} den Wert -1 annehmen soll. Somit muß das Muster p für das entsprechende Neuron als Mitglied der ersten Mustermenge betrachtet werden.

Beachtet man an dieser Stelle wieder die Möglichkeit, die Vektoren $\vec{x}^{(1)}$ und $\vec{x}^{(2)}$ als Summen der einzelnen Änderungen $\Delta g_k \cdot \vec{x}_k$ zusammenzusetzten, kann man nun ein Lernverfahren für hybride Q-Neuron-Perzeptron-Netze angeben:

1. **Initialisierung:** Die Gewichte w_{ij} zwischen der verdeckten Schicht und der Ausgabeschicht werden mit zufälligen Werten initialisiert. Die Hilfsvektoren $\vec{x}^{(1)}$ und $\vec{x}^{(2)}$ eines jeden Q-Neurons werden ebenfalls zufällig initialisiert.
2. **Vorwärtspropagation:** Es wird ein Muster p zufällig ausgewählt und angelegt. Für alle Q-Neuronen wird ihr Quotient Q_{pj} und daraus ihre Ausgabe o_{pj} bestimmt. Aus diesen berechnen sich die Ausgaben o_{pi} der Ausgabeneuronen.

3. **Änderung der Verbindungsgewichte berechnen:** Die Änderungen der Gewichte zwischen der verdeckten Schicht und der Ausgabeschicht werden analog zum Backpropagation-Verfahren nach der Formel

$$\Delta w_{pij} = \eta o_{pj} f'_{act}(net_{pi})(t_{pi} - o_{pi})$$

   berechnet.

4. **Fehlerrückpropagation:** Es werden die Fehlersignale für die Q-Neuronen in der verdeckten Schicht berechent:

$$\delta_{pj} = \sum_i w_{ij} f'_{act}(net_{pi})(t_{pi} - o_{pi}) \tag{131}$$

5. **Bestimmung der Korrekturen Δg_p und $\Delta\Theta$:** Für jedes Q-Neuron wird das Muster p anhand des Fehlersignals einer der Mustermengen zugeordnet. Damit ist sein Quotient bestimmt. Die Wichtungsänderung Δg_p und die Schwellwertänderung $\Delta\Theta$ ergiben sich aus der entsprechenden Korrekturformel.

6. **Korrektur aller Gewichte:** Die Verbindungsgewichte w_{ij} werden um die ermittelten Beträge Δw_{ij} korrigiert. Für jedes Q-Neuron werden die Hilfsvektoren $\vec{x}^{(1)}$ und $\vec{x}^{(2)}$ (in Abhängigkeit davon, zu welcher Mustermenge das entsprechende Muster in diesem Neuron zugeordnet ist) um die Beträge $\Delta g_p \cdot \vec{x}_p$ erhöht, außerdem wird die Korrektur für Θ ausgeführt. Danach werden die Gewichte der Q-Neuronen neu berechnet.

7. **Alle Muster gelernt ?:** Wenn nicht alle Muster die gewünschte Ausgabe erzeugen, wird mit Schritt 2 fortgesetzt.

8. **Ende**

Dieses Verfahren ist ein Online-Lernverfahren. Man kann es jedoch in ein Batch-Lernverfahren überführen, indem man die Schritte 2 bis 5 für alle Muster hintereinander ausführt und die einzelnen Änderungen summieret Und erst danach die Gewichtsupdates durchführt.

Als letztes sind noch die in Punkt 5. angesprochenen Korrekturformeln näher zu spezifizieren. Da zum einen nicht alle Muster in jedem Neuron einer Mustermenge fest zugeordnet und zum anderen Wechsel der zugeordneten Mengen zu erwarten sind, können Korrekturformeln, die sich auf Werte des letzten Lernschrittes beziehen, wahrscheinlich nur sehr schlecht eingesetzt werden. Somit kommen nur die Korrekturterme der ersten Stufe in Frage (siehe Abschnitt 2.3):

$$\Delta\Theta^{(1)} = \tau_1^{(1)} \cdot \arctan(\tau_2^{(1)} \Delta\bar{Q}(t-1)), \tag{132}$$

$$\Delta g_k^{(1)} = \gamma^{(1)} \cdot \Delta Q_k(t-1). \tag{133}$$

Die darin verwendeten Abweichungen $\Delta\bar{Q}$ und ΔQ_k sind definiert als

$$\Delta\bar{Q} = Q^{(S)} - \bar{Q} \tag{134}$$

$$\Delta Q_k = Q_k - \bar{Q}. \tag{135}$$

Der Wert $\Delta\bar{Q}$ stellt jedoch wieder einen solchen unliebsamen, sich aus allen Quotienten ergebenden Wert dar. Deshalb wurden die in den Gleichungen (132) und (133) enthaltenen Differenzen durch die Differenz des einzelnen Quotienten zum gewünschten Sollwert ersetzt. Da dadaurch jedoch für jedes Muster ein $\Delta\Theta$ entsteht, das in der Größenordnung des ursprünglichen Wertes für alle Muster liegt, muß dieser Wert noch entsprechend der Anzahl der Muster verringert werden. Die Korrekturformeln ergeben sich somit als:

$$\Delta\Theta_k = \frac{1}{K_1 + K_2} \cdot \tau_1^{(1)} \cdot \arctan(\tau_2^{(1)}(Q^{(S)} - Q_k)), \tag{136}$$

$$\Delta g_k = \gamma^{(1)} \cdot (Q_k - Q^{(S)}). \tag{137}$$

5.3 Implementation und Tests

Die Implementation des soeben beschriebenen Lernverfahrens bereitete keine wesentlichen Probleme.

Für die Initialisierung der Hilfsvektoren $\vec{x}^{(1)}$ und $\vec{x}^{(2)}$ wurden kleine Zufallszahlen im Intervall $[0.0001, 0.0002]$ verwendet, Θ wurde mit 1.0 initialisiert. Die Verbindungsgewichte zwischen der verdeckten Schicht und der Ausgabeschicht wurden mit Zufallszahlen aus dem Intervall $[0, 1]$ und die Schwellwerte der Ausgabeneuronen mit 0.0 vorbelegt.

Als Lernparameter für die Q-Neuronen wurden die bewährten Werte aus Tabelle 1 benutzt. Als Wert für η wurde 1.0 eingesetzt.

Der zulässige Gesamtfehler des Netztes wurde auf 0.1 festgelegt.

Für die Reihenfolge der Musterpräsentation wurden verschiedene Verfahren implementiert:

- **Random:** Das nächste dem Netz zu präsentierende Muster wurde zufällig ausgewählt. Danach wurden die Gewichte neu bestimmt.
- **Round Robin:** Die Muster wurden der Reihe nach ausgewählt. Nach jedem einzelnen Muster wurden die Gewichte neu berechnet.
- **Worst Pattern Next:** Es wurde für jeden Lernschritt das Muster ausgewählt, welches den größten Einzelfehler erzeugt. Auch hier erfolgte in jedem Schritt ein Gewichtsupdate.
- **Batchbetrieb:** Hier wurden, wie oben beschrieben, alle Muster nacheinander angelegt und ihre einzelnen Änderungen Δw_{ij}, Δg_k und $\Delta\Theta_k$ summiert. Erst nachdem die Korrekturen eines jeden Musters bestimmt waren, wurden die tatsächlichen Gewichtveränderungen ausgeführt.

Die Ergebnisse der umfangreichen Tests sowie ihre Interpretation sind in den nächsten Abschnitten zu finden.

5.3.1 Unterscheidung von zwei Klassen

In einem ersten Test wurde das Netz angelernt, zwei Klassen von Mustern zu unterscheiden. Dabei sollten in der verdeckten Schicht ein Q-Neuron und in der Ausgabeschicht zwei Perzeptrons verwendet werden. Diese kommt dem einfachen Lernen eines einzelnen Q-Neurons sehr nahe. Der wesentliche Unterschied besteht darin, daß hier die Einteilung der Muster in die Mustermengen nicht a priori vorgegeben ist, sondern durch das Netz bestimmt wird.

In dem Test wurden wieder die bereits im Abschnitt 2.6.2 verwendeten Klassen 0 und 6 der Postleitzahlen benutzt.

Im allgemeinen kam das Lernen relativ schnell zum Erfolg. Der Verlauf des Gesamtfehlers des Netztes über alle Muster wurde nach einem kurzen recht steilen Abstieg sehr flach mit einigen Sprüngen. Die Sprünge entstanden dadurch, daß das Q-Neuron plötzlich die Eingabe von -1 auf 1 oder umgekehrt wechselte. Der ansonsten sehr geringe Abfall des Fehlers entsteht durch das langsame Lernen der Perzeptrons. Die Verläufe des Gesamtfehlers sind für die einzelnen Verfahren in den Bildern 47 bis 50 dargestellt.

Zwischen den einzelnen Präsentationstrategien gab es jedoch wesentliche Unterschiede:

Round Robin: Dieses Verfahren brauchte in der Regel länger als die anderen Vefahren, um alle Muster zu lernen. Dies ist nicht verwunderlich, da das Netz auch ständig für Muster, die es bereit gut erkennt, unnötige Lernschritte ausführt.

Random: Dieses Verfahren war in der Regel auch noch sehr langsam, jedoch noch etwas schneller, als Round Robin. Der Grund dafür ist, daß bei Round Robin erst die Muster der einen Klasse und danach die Muster der anderen Klasse präsentiert wurden. Dadurch wurden insgesamt immer relativ starke Änderungen erzeugt, welche danach wieder kompensiert werden mußten. Würde man die Muster in der Reihenfolge mischen, wäre der Unterschied zwischen Random und Round Robin sicherlich nicht mehr markant.

Worst Pattern Next: Dieses Verfahren kam wesentlich schneller zum Erfolg als die anderen. Der Grund hierfür ist offensichtlich: Da immer das Muster mit dem größten Fehler im nächsten Lernschritt verwendet wird, wird der Fehler recht schnell minimiert. Auch blieben auf diese Art die sonst üblichen Sprünge im Gesamtfehler aus, denn bereits nach wenigen Lernschritten waren die Quotienten der Muster so eingestellt, wie sie bleiben sollten. Das weitere Lernen war nur noch die Anpassung der ausgebenden Perzeptrons an die geforderten Werte.

Bei der Betrachtung der einzelnen Quotienten der Muster fiel jedoch ein Nachteil dieser Vorgehensweise auf: Die Werte waren meist näher an 1 als bei den anderen Verfahren. Das hat seine Ursache darin, daß nur einige Muster für das Lernen verwendet werden und die Quotienten der anderen gar nicht optimiert werden. (Siehe dazu auch die Beispiele für das Lernen von vier Klassen im Abschnitt 5.3.2.)

Batchbetrieb: Der Trainingsaufwand im Batchbetrieb war etwa mit dem für Random zu vergleichen, er war etwas geringer als für Round Robin. Offensichtlich sind die oben beschriebenen Probleme auch hierfür als Ursache anzusehen.

Die sich einstellende Funktionsweise des Netzes ist, wie zu erwarten war, sehr einfach: Das Q-Neuron unterscheidet die beiden Klassen und liefert dementsprechend eine 1 oder eine -1 als Ausgabe. Die Gewichte der Ausgabeneuronen haben sich so eingestellt, daß das eine genau dann eine 1 als Ausgabe liefert, wenn am Eingang eine 1 anliegt, und das andere genau dann, wenn an seinem Eingang eine -1 anliegt.

5.3.2 Unterscheidung von 4 Klassen

Im nächsten Versuch sollte mit einem Netz gelernt werden, vier verschiedene Klassen von Mustern zu unterschieden. Als Beispiel dazu wurden wieder die Postleitzahlen verwendet, die zu trennenden Muster gehörten zu den Klassen 0, 4, 6 und 7. Um das Lernen besser beobachten zu können wurden aus jeder Klasse nur je 3 Muster ausgewhlt.

1. Zwei Q-Neuronen

Im ersten Versuch bestand die verdeckte Schicht aus zwei Q-Neuronen. Die ist die minimale Anzahl, um die Aufgabe überhaupt erfüllen zu können. Es war zu erwarten, daß durch diese Neuronen eine Art Binärkodierung für die angelegten Muster erzeugt werden würde, welche dann von den Ausgabeneuronen in eine 1 aus N Kodierung umgesetzt wird. Dazu müßte z.B. ein Q-Neuron lernen, die Klassen 0 und 4 von den Klassen 6 und 7 zu unterscheiden, während das andere die Klassen 0 und 6 von den Klassen 4 und 7 trennt.

Das Ergebnis der Versuche hat die Erwartungen im vollen Umfang bestätigt. Die Ausgaben der verdeckten Neuronen für die einzelnen Muster für einen Beispiellauf sind in der Tabelle 6 dargestellt. Die Grundstruktur des Netzes (Verteilung der Muster in die jeweiligen Mustermengen) stellte sich bereits nach relativ kurzer Zeit ein, danach mußten die Gewichte der Ausgabeneuronen entsprechend adaptiert werden.

	Klasse 0	Klasse 4	Klasse 6	Klasse 7
	Nr. 1, 2, 3	Nr. 52, 53, 54	Nr. 77, 78, 79	Nr. 94, 95, 96
H_0	-1	1	-1	1
H_1	1	-1	-1	1

Tabelle 6: *Ausgaben der Q-Neuronen für die angelegten Muster*

Dieses Beispiel wurde mit den Präsentationstrategien Round Robin und Worst Pattern Next untersucht. Die Lerndauer unterschied sich jetzt jedoch kaum noch. Durch die steigende Komplexität der Aufgabe gewinnen nun offensichtlich die Probleme des Worst-Pattern-Next-Verfahren an Bedeutung und kompensieren seine Vorteile. Die wesentlichen zu beobachtenden Unterschiede kann man wie folgt zusammenfassen:

- Mit dem Worst-Pattern-Next-Verfahren wurde der Gesamtfehler (siehe Abbildung 51) wesentlich schneller relataiv klein (etwa 1.5), als beim Round-Robin-Verfahren (siehe Abbildung 54). Damit steht natürlich eng im Zusammenhang, daß sich durch die ständige Presentation der Muster, die besonders starke Fehler verursachen, sehr schnell die gewünschten Ausgaben der Q-Neuronen einstellen.

- Im weiteren Lernverlauf traten bei dem Worst-Pattern-Next-Verfahren gelegentlich kurzzeitige, aber starke Anstiege des Gesamtfehlers auf. Die Ursache liegt in dem bereits oben angesprochenen geringeren Abstand der Quotienten von 1, welches der Schwellwert der Ausgabefunktion der Q-Neuronen ist. Dadurch kommt es gelegentlich zu einer Umkehr der Ausgabe eines verdeckten Neurons, was fatale Auswirkungen auf den Fehler hat. Die Abbildungen 52 und 53 verdeutlichen dies. Das Round-Robin-Verfahren zeigte dieses Verhalten überhaupt nicht, was an den Abbildungen 55 und 56 verdeutlicht wird.

2. Vier Q-Neuronen

Im zweiten Versuch wurde die verdeckte Schicht aus vier Q-Neuronen gebildet. Dadurch erhält das Netz eine redundante Struktur, denn es sind nicht alle vier Q-Neuronen notwendig, um für die präsentierten Muster eine Binärkodierung zu bilden.

Da dem Netz mehr Freiheiten bei der Ausbildung der Gewichtsstruktur bleiben, war zu erwarten, daß einige Gewichte verkümmern, also einen Wert nahe 0 annehmen. Damit stellen die Ausgaben des dazugehörenden Q-Neurons einen "Don't care - Wert" dar, sie haben keinen wesentlichen Einflußauf die Ausgaben des Netzes. In der Tabelle 7 sind die Ausgaben der Q-Neuronen für die jeweiligen Muster dargestellt, welche sich in einem Testlauf ergeben haben. Stark schwankene Ausgaben der verdeckten Schicht bei einer Mustermklasse gehen einher mit kleinen Verbindungsgewichten von diesem Neuron zu dem diese Klasse identifizierenden Ausgabeneuron. Dazu sei auf die Beispiele im Abschnitt 5.3.3 verwiesen.

Es ist schwer, hier Ursache und Wirkung zu unterscheiden. Ist das kleine Gewicht die Ursache dafür, daß die unsichere Ausgabe des Q-Neurons keine Auswirkung auf die Ausgabe des Perzeptrons und somit auf den Fehler hat und deshalb diese Schwankungen nicht behoben wird? Oder sorgt die sich ständig ändernde Ausgabe des Q-Neurons und damit ein anfangs auftretender großer Fehler dafür, daß das Gewicht 0 wird? Was man mit Sicherheit sagen kann ist nur, daß sich bei einem kleinen Gewicht der Fehler

	Klasse 0	Klasse 4	Klasse 6	Klasse 7
	Nr. 1, 2, 3	Nr. 52, 53, 54	Nr. 77, 78, 79	Nr. 94, 95, 96
H_0	1	-1	1	?
H_1	-1	?	1	1
H_2	-1	?	1	?
H_3	?	1	1	-1

Tabelle 7: *Ausgaben der Q-Neuronen für die angelegten Muster. "?" bedeutet, daß dieser Wert stark zwischen 1 und -1 schwankt.*

des entsprechenden Ausgabeneurons nicht besonders stark auf das Lernen des verdeckten Neurons auswirkt. Allerdings kann selbst dieses kleine Fehlersignal (insbesondere in Verbindung mit den anderen Fehlersignalen) das Lernen immer noch erheblich beeinflußen, da der Betrag des Fehlersignals nicht in die Korrektur einfließt, sondern nur dessen Vorzeichen.

Die Lerngeschwindigkeit, bezogen auf die Anzahl der Lernschritte (die natürlich etwas aufwendiger waren), wurde durch das Hinzufügen der Redundanz etwa verdoppelt. Hier macht sich die höhere Fehlertolerenz gegenüber abweichenden Ausgaben in verdeckten Schicht positiv bemerkbar (siehe dazu auch Abbildung 57).

5.3.3 Unterscheiden aller Ziffern

In diesem Versuch sollte das Netzwerk alle zehn Ziffern unterscheiden. Um auch hier wieder den Trainingsaufwand zu verringern und die Auswertung zu erleichtern, wurde nur ein Muster je Klasse ausgewählt.

Da sich in den ersten Tests gezeigt hatte, daß das adaptieren der Perzeptron-Gewichte einen erheblichen Teil des Lernens ausmacht, wurde die Lernrate auf 2.0 erhöht, um eine schnellere Konvergenz zu erreichen. Die Gewichtskorrekturen fallen damit etwas heftiger aus, sind jedoch noch nicht übermässig groß.

Außerdem ist aufgefallen, daß der Fehler im Endbereich nur sehr langsam konvergiert. Deshalb wurde der zugelassene Fehler des Netzes angehoben, um diese Phase ebenfalls zu verkürzen. Er wurde auf 0.2 gesetzt. Damit ist immer noch sichergestellt, daß ein Netz, welches diese Bedingung erfüllt, alle

Trainingsmuster korrekt erkennt, d.h. die Ausgabe eines jeden Neurons um weniger als 0.5 vom geforderten Wert abweicht.

1. Vier Q-Neuronen

Das Netz bestand aus vier verdeckten Q-Neuronen und zehn Perzeptrons in der Ausgabeschicht. Auch hier ist die verdeckte Schicht bereits redundant, denn mit den vier Neuronen könnten theoretisch 16 verschiedene Klassen repräsentiert werden.

Demzufolge kam es auch in diesem Versuch wieder zur Ausbildung von "don't care"-Werten. Im weiteren sollen die Ergebnisse anhand des Worst Pattern Next - Verfahrens dargestellt werden. Die Q-Ausgaben eines Beispiellaufes sind in der Tabelle 8 dargestellt.

Klasse	0	1	2	3	4	5	6	7	8	9
Muster	1	12	26	39	52	64	77	93	106	119
H_0	?	-1	1	?	-1	1	1	-1	1	-1
H_1	1	-1	-1	1	-1	-1	-1	1	1	1
H_2	-1	?	1	-1	-1	1	-1	1	1	1
H_3	-1	1	1	1	-1	-1	?	1	?	-1

Tabelle 8: *Ausgaben der Q-Neuronen für die angelegten Muster*

Auch hier wurde der Gesamtfehler des Netzes relativ schnell (etwa nach 400 bis 500 Lernschritten) sehr klein (siehe Abbildung 59, ein Großteil des Fehlers entstand durch die unzureichende Anpassung der Verbindungsgewichte. Da jedoch danach wiederum nur mit wenigen Mustern gelernt wurde, verschlechterten sich die Quotienten der einzelnen Q-Neuronen wieder. Dies hatte eine erneute Unruhe im Netz zur Folge, welche natürlich auch das weitere Lernen der Verbindungsgewichte beeinträchtigte. Erst nach etwa 2000 bis 2500 Lernschritten kam das Lernverfahren zum Erfolg.

Wie man am Verlauf des Gesamtfehlers sehen kann, weist dieser auch noch kurz vor dem Ende des Lernens ständig erhebliche Sprünge auf. Somit ist der letzendlich erreichte Lernerfolg eher das Ergebnis einer sich mehr oder weniger zufällige einstellenden günstigen Verteilung der Q-Neuronen-Ausgaben. Diese Vermutung wird dadurch untermauert, daß viele Quotienten sehr nahe an 1 liegen.

Die Ausgabe des Q-Neurons H_0 für das Muster der Klasse 0 ist wieder ein sehr typisches Beispiel für einen "don't care"-Wert, man betrachte dazu die Abbildungen 60, 61 und 62. Das Gewicht liegt fast während der gesamten Lernphase bei einem Wert nahe 0. Das gegen Ende zu verzeichnende Ansteigen des Gewichtes steht mit den oben beschriebenen Fehlereffekten im Zusammenhang, es ist jedoch offensichtlich nicht von Bedeutung, da die anderen Eingaben von O_0 diese Auswirkungen kompensieren können. Wenn man sich die für das Muster der Klasse 0 geforderte Mengenzugehörigkeit im Neuron H_0 betrachtet, so erkannt man, daß diese ständig wechselt. Dieses Verhalten ist für die Einstellung eines "don't care"-Wertes absolut typisch. Dies ist natürlich als eine Ursache für den sich nahe bei 1 einstellenden Quotienten anzusehen, da sich die Korrekturen gegenseitig kompensieren.

Am Beispiel des Musters 12 (Klasse 1) erkennt man sehr schön die oben beschriebene Problematik des im weiteren Lernverlauf wieder auftauchenden Fehlers, hierzu vergleiche man die Abbildungen 63, 64 und 65. Die Ausgabe des Neurons H_2 hatte sich anfänglich fest auf den Wert 1 eingestellt, der Quotient hatte einen recht großen Wert von deutlich über 1.2 bis hin zu 1.5. Dies hatte sich auf grund der festen Mengenzugehörigkeit des Musters zur zweiten Mustermenge dieses Neurons so ergeben. Dann kam es jedoch zu einem häufigen Wechseln der Mengenzugehörigkeit, woraufhin sich der Quotient schnell deutlich verschlechteret, bis letztendlich die Ausgaben des Neurons ebenfalls zu schwanken anfingen. Da jedoch das entsprechende Verbindungsgewicht sehr groß war, hatte dies sehr starke Auswirkungen auf den Fehler des Musters.

Da ein solcher Effekt bei sehr viele Neuronen auftrat, führte dies zu einem ständigen Wechseln des von einem Muster erzeugten Binärkodes der Zwischenschicht. Dies behinderte einerseits das Lernen der Verbindungsgewichte und sorgte andererseits dafür, daß viele Quotienten einen Wert nahe 1 annahmen.

2. Acht Q-Neuronen

In einem weiteren Versuch wurde die verdeckte Schicht aus 8 Q-Neuronen gebildet.

Das Lernen war damit wesentlich besser durchführbar (siehe Abbildunge 67 und 67, da durch die gestiegene Redundanz im Netz schlechte Trennungsleistungen der einzelnen Q-Neuronen nicht mehr so stark ins Gewicht fielen.

Dies erkennt man in der Tabelle 9 insbesondere an der deutlich gestiegenen Anzahl von “don’t care”-Werten.

Klasse	0	1	2	3	4	5	6	7	8	9
Muster	1	12	26	39	52	64	77	93	106	119
H_0	1	?	-1	1	-1	-1	-1	1	-1	1
H_1	?	-1	?	-1	1	1	1	1	-1	1
H_2	-1	1	1	1	-1	1	-1	-1	-1	?
H_3	1	-1	?	1	?	1	?	1	?	1
H_4	1	-1	?	1	-1	-1	1	1	1	-1
H_5	1	1	-1	-1	?	-1	1	1	-1	1
H_6	1	1	-1	-1	-1	1	-1	?	1	?
H_7	1	-1	-1	1	1	1	?	?	-1	-1

Tabelle 9: *Ausgaben der Q-Neuronen für die angelegten Muster*

Auch hier gingen, wie zu erwarten war, diese “don’t care”-Werte mit starke Schwankungen in der Mengenzuordnung und kleinen Gewichten um 0 einher.

Die Trainingszeiten konnten jedoch erheblich verkürzt werden. Es waren meist nur etwa ein Zehntel der Lernschritte des vorherigen Tests erforderlich. Dies ist auch bei gestiegener Netzkomplexität eine erhebliche Einsparung.

3. Zwei Muster je Klasse

In weiteren Tests wurden nun jeweils zwei Muster pro Klasse zum trainieren benutzt.

Der Trainingsaufwand stieg dadurch erheblich, im Schnitt auf mehr als das Doppelte. Die gestiegene Anzahl der Muster hatte auch eine steigende Anzahl von sich widersprechenden Korrekturen zur Folge. Außerdem steigt der Netzwerkfehler allein durch die Anzahl der Muster an.

Die Test haben keine qualitativ neuen Ergebnisse gebracht. Wie zu erwarten war, konnten die Netze mit acht Neuronen in der verdeckten Schicht wesentlich schneller lernen, als die mit vier. Auch war das Lernen mir Worst Pattern Next in der Regel schneller als mit Round Robin, was allerdings bei vier Q-Neuronen kaum noch zur Geltung kam. Dies Boabachtungen sind in den Abbildungen 68 bis 71 nachzuvollziehen.

Die Fehlerverläufe der einzelnen Muster wiesen wieder die für Round Robin und Worst Pattern Next typischen Charakteristika auf:

- Bei Round Robin wurden oft Muster erst relativ gut angelernt und wiesen später durchgängig einen großen Fehler auf. Andererseits wurden Muster, welche anfangs einen recht großen Fehler hatten irgend wann plötzlich gut erkannt und behielten danach einen sehr kleinen Fehler.
- Bei Worst Pattern Next erreichten alle Muster relativ schnell einen kleinen Fehler. Im weiteren Lernverlauf traten kurzzeitig und sporadisch wieder größere Fehler auf.

5.3.4 Vergleich mit Backpropagation

Um wiederum einen Effizienzvergleich zu haben, wurde auch für diese Tests ein Perzeptron-Netzwerk mit dem Lernverfahren Backpropagation angelernt. Dabei wurden sowohl unterschiedliche Netzstrukturen als auch verschiedene Musterpräsentationsstrategien untersucht. Die Ergebnisse sind in den Bildern 72 bis 76 dargestellt.

Um die Konvergenzgeschwindigkeit zu steigern, wurde wieder mit einer extrem großen Lernrate von 10.0 gearbeitet.

Einfluß der Präsentationsstrategie

Der grundsätzliche Verlauf des Gesamtfehlers wird durch die Präsentationstrategie wesentlich mitbestimmt. Bei Round Robin nimmt der Fehler relativ gleichmäßig ab. Bei Worst Pattern Next kann man ein Art strukturierende Phase bobachten, in Abbildung 75 etwa die Lernschritte 250 bis 500. In diesem Bereich wird der Fehler (unter starken Schwankungen) sehr schnell verringert, während davor und danach einen wesentlich geringerer Abstieg zu verzeichnen ist.

Je geringer die Redundanz im Netz ist (d.h. je weniger Neuronen in der verdeckten Schicht enthalten sind), um so deutlicher wird der Einfluß der Präsentationstrategie auf die Lerngeschwindigkeit. Worst Pattern Next ist besonders bei geringer Redundanz schneller, weil bei diesem Verfahren die Muster, welche große Fehler produzieren, besonders häufig präsentiert werden.

Einfluß der Redundanz

Wie zu erwarten war, wurde schneller gelernt, je größer die Redundanz im Netz war.

Interessant ist noch, daß es gelang, ein Netz mit nur 3 Neuronen in der verdeckten Schicht korrekt anzulernen. Damit konnte das Netz in der verdeckten Schicht nicht mehr eine einfache binäre Repräsentation ausbilden, wie dies sonst immer der Fall war. Es entstand eine ternäre Kodierung.

Insgesamt erscheint die Anzahl der benötigten Lernschritte relativ gering, zumeist noch geringer als bei den hybriden Netzen. Es sei jedoch noch einmal die sehr große Lernrate von $\eta = 10.0$ hervorgehoben. Unter Umständen ist diese bei anderen Mustern nicht mehr haltbar und müßte verringert werden. In dem getesteten Bereich von $\eta = 1.0 \ldots 10.0$ war der Zusammenhang von η und der Anzahl der benötigten Lernschritte nahezu linear.

Für weitere Betrachtungen der Konvergenzgeschwindigkeit siehe Abschnitt 5.4

5.4 Zusammenfassung

Es war prinzipiell möglich, ein hybrides Netz aus Q-Neuronen und Perzeptrons anzulernen. Die Trainingszeiten sind für verschiedene Aufgaben und Netze in der Tabelle 10 zusammengestellt. Die Schwierigkeiten stiegen mit der Anzahl der zu trainierenden Muster sehr stark an. Wie man in den letzten beiden Zeilen der Tabelle erkennt, war es nicht möglich, ein Q-Netz mit allen in der ausgewählten Trainingsmenge enthaltenen Mustern anzulernen. Generell kann man feststellen, daß die Q-Netze insbesondere bei starker Redundanz im Netz den MLP's bei den benutzten Lernparametern etwas überlegen sind. Die für die MLP's gewählte Lernrate ist mit 2.0 zwar schon sehr groß, könnte aber für diese Beispiele ohne Qualitätsverlust auch auf 10.0 erhöht werden, wodurch ihre Lernzeit auf etwa 20% sinken würde.

Es sei noch einmal explizit darauf hingewiesen, daß das in diesem Vergleich verwendete MLP mit seiner Struktur als Mustererkenner für dieses Problem völlig ungeeignet ist, darauf deuten auch die Fehlerraten von über 50% auf einer Testmenge hin. Moderne Mustererkenner sind sowohl in ihrer Leistungsfähigkeit als auch in ihrer Lerngeschwindigkeit einem solchem MLP weit überlegen.

Aufgabe und Musteranzahl		Netzart, η, Struktur und Präsentationsstrategie				Lern-schritte	Zeit in s
Klassen 0, 6	27	Q-Neuron	n.a.	n.a.	n.a.	426	3.5
Klassen 0, 6	27	Perzeptron	1.0	n.a.	RR	4010	7.0
1 je Klasse	10	Q-Netz	1.0	4x10	RR	2107	50.9
1 je Klasse	10	Q-Netz	1.0	4x10	WPN	2740	65.0
1 je Klasse	10	Q-Netz	1.0	8x10	RR	490	22.8
1 je Klasse	10	Q-Netz	1.0	8x10	WPN	318	14.6
1 je Klasse	10	MLP	2.0	4x10	RR	2983	142.3
1 je Klasse	10	MLP	2.0	4x10	WPN	2913	142.5
1 je Klasse	10	MLP	2.0	8x10	RR	1169	109.6
1 je Klasse	10	MLP	2.0	8x10	WPN	1117	106.5
2 je Klasse	10	Q-Netz	1.0	4x10	RR	4790	169.8
2 je Klasse	10	Q-Netz	1.0	4x10	WPN	4470	154.5
2 je Klasse	10	Q-Netz	1.0	8x10	RR	1108	74.4
2 je Klasse	10	Q-Netz	1.0	8x10	WPN	638	43.1
2 je Klasse	10	MLP	2.0	4x10	RR	5917	297.5
2 je Klasse	10	MLP	2.0	4x10	WPN	4840	243.2
2 je Klasse	10	MLP	2.0	8x10	RR	2272	227.4
2 je Klasse	10	MLP	2.0	8x10	WPN	2231	224.8
alle	134	MLP	2.0	8x10	WN	4450	520.1
alle	134	Q-Netz	1.0	8x10	WN	∞	∞

Tabelle 10: *Lernaufwand für verschiedene Aufgaben und Netze*

Das zweifellos größte Problem diese Netzwerk-Lernverfahrens ist es, daß immer noch für jedes Neuron alle Muster in jeweils eine Mustermenge eingeordnet werden. Somit versucht jedes Q-Neuron also, die gesamte Lernmenge in zwei Teilmengen zu zerlegen. Dies ist, wie bereits in der Herleitung des Lernverfahrens bemerkt, eine nahezu unlösbare Aufgabe.

Eine Möglichkeit dieses Problem anzugehen wäre, über einen Grenzwert für die Fehlersignale δ_j auch Muster aus den zu unterscheidenden Mustermengen auszuschließen.

Ein anderer Ansatz wäre, die Kardinalität der Lernmenge durch vorherige Clusterung mit Standardalgorithmen (*k-means*, SOM) zu verringern und nur noch mit den so erhalten Prototypem zu arbeiten.

Eine weitere Möglichkeit, das Netz flexibler zu gestalten, bestünde darin, den Q-Neuronen an Stelle der einschichtigen Perzeptrons ein mehrschichtiges Netz nachzuschalten. Dies hätte auf jeden Fall den Vorteil, daß ein Q-Neuron nicht mehr alle Muster einer Klasse ein und der selben Mustermenge zuordnen muß.

Ein weiteres sehr großes Problem des Verfahrens besteht darin, daß häufige Wechsel der Mengenzuordnung auftreten und somit schnell nacheinander widersprüchliche Korrekturen erzeugt werden, was zumeist einen Quotienten um 1 und somit instabile Ausgaben zur Folge hat.

Als Ansatz für die Lösung dieses Problems wäre es denkbar, die Änderungsfrequenz zu begrenzen, indem man:

- eine Zuordnungsänderung erst eine bestimmte Anzahlvon Lernschritten nach der letzten zuläßt
- nach einer gewissen Anzahl von Zuordnungsänderungen in den letzten Lernschritten weitere Musteränderungen für die nächsten Lernschritte verbietet
- für eine gewisse Anzahl von Lernschritten nur eine bestimmte Anzahl von Zuordnungsänderungen zuläßt, was dem letzten Punkt jedoch sehr nahe kommt.

Eine weitere Idee wäre es, die Zuordnung über eine Art Summierung über die letzten ermittelten Wert mit einem "Vergessensterm" einzuführen. Somit wird ein Muster aus einer Menge automatisch entfernt, wenn es längere

Zeit nicht wiederholt wurde. Dies stellt eine Vereinigung der ersten beiden Lösungsansätze dar.

Speziell beim Lernen mit dem Round-Robin-Verfahren traten häufig Fälle auf, in denen eine "don't care"-Verbindung mit einem Gewicht nahe 0 nach längerer Zeit wieder aufgelöst wurde, was ein Ansteigen des Fehlers bewirkte.

Um diesem Effekt entgegenzuwirken, könnte man Pruning einsetzten, um Gewichte, welche über einen längeren Zeitraum immer nahe 0 waren, gänzlich aus dem Netz zu entfernen.

Eine weitere häufige Ursache für Fehler waren instabile Ausgaben der Q-Neuronen, welche für ein Muster einen Quotienten nahe 1 lieferten.

Durch die Verwendung einer dreistufigen Ausgabefunktion der Art

$$f_{out}(Q) = \begin{cases} -1 & \text{falls} \quad Q < 0.8 \\ 0 & \text{falls} \quad 0.8 \leq Q \leq 1.2 \\ 1 & \text{falls} \quad 1.2 < Q \end{cases} \tag{138}$$

könnte man deren Wirkung verringern. Es ist jedoch fraglich, ob man damit nicht nur die Instabilität zu einem der beiden neuen Schwellwerte verlagert.

Alles in allem gibt es noch eine Reihe von Möglichkeiten, dieses Verfahren zu verfeinern.

6 Integration in den SNNS

6.1 Einführung in den SNNS

Der Stuttgarter Neuronale Netze Simulator (SNNS) wurde am Institut für Verteilte Höchstleistungsrechner (IPVR) der Universität Stuttgart unter Leitung von Andreas Zell entwickelt ([Zell94], [SNNS94]). Er ist ein Hilfsmittel zur effizienten Simulation von künstlichen neuronalen Netzen und beinhaltet Möglichkeiten zur Generierung, zum Training, Test und zur Visualisierung von neuronalen Netzen.

Der SNNS besteht aus drei Hauptkomponenten: einem Simulatorkern, einer graphischen Benutzeroberfläche (basierend auf X-Windows) und einem Compiler zur Umsetzung einer höheren Netzwerkbeschreibungssprache.

Der Simulatorkern arbeitet auf der internen Repräsentation der Netze. Er ist so gestaltet, daß er ohne eingreifende Änderungen erweitert werden kann. Sämtliche Aufgaben, welche mit der Lern- und Arbeitsphase in Verbindung stehen, werden durch ihn übernommen.

Der Kern selbst ist wiederum in drei Teile gegliedert. Der innerste Teil, eine Speicherverwaltung, sorgt für eine effiziente Arbeit mit den Daten. Dies wird vor allem dadurch erreicht, daß sämtliche Daten immer in größeren Blöcken angefordert werden. Die nächste Schicht wird von den Funktionen zur Manipulation des Netzes gebildet. Dabei handelt es sich unter anderem auch um die Propagierungs- und Lernfunktionen. Die äußere Hülle bilden die Funktionen, welche die Schnittstelle zur graphischen Oberfläche und zur Dateisystem darstellen.

Die graphische Oberfläche bietet auch einem weniger geübten Nutzer die Möglichkeit, Netze nach eigenen Bedürfnissen zu konfigurieren, Muster anzulernen, abzurufen, usw. Ein sehr wesentlicher Bestandteil ist die Visualisierung von Netzen. Sie ist ebenfalls komfortabel konfigurierbar.

Mit der Netzwerkbeschreibungssprache Nessus wird ermöglicht, z.B. die Netzstruktur in einer höheren Sprache zu beschreiben. Sie wird dann vom Compiler in eine vom SNNS verarbeitbare Eingabedatei umgesetzt.

6.2 Interne Datenstrukturen

Units

Eine Unit (auch Zelle genannt) stellt im SNNS ein Neuron im allgemeinsten Sinne dar. Es werden verschiedene Typen unterschieden: Eingabe-, verdeckte und Ausgabeunits. Außerdem gibt es noch einige spezielle Typen, welche in der Regel nur mit selbstdefinierten Funktionen zu verwenden sind.

Für jede Zelle werden in einer entsprechenden Struktur eine Reihe von Attributen gespeichert. Die wichtigsten sind:

- **Nummer:** Anhand dieser wird die Unit eindeutig identifiziert und angesprochen.
- **Aktivierung:** Die Aktivierung der Unit, welche sich über die Aktivierungsfunktion aus der Netzeingabe ergibt.
- **Ausgabe:** Die Ausgabe der Unit, welche sich über die Ausgabefunktion aus der Aktivierung ergibt.
- **Schwellwert**
- **Aktivierungsfunktion:** Sie berechnet aus der Netzeingabe, der alten Aktivierung und dem Schwellwert die neue Aktivierung der Unit. Diese Funktion kann vom Nutzer in C implementiert und an den existierenden Simulatorkern angebunden werden.
- **Ausgabefunktion:** Sie berechnet die Ausgabe in Abhängigkeit von der aktuelle Aktivierung. Auch diese Funktion kann vom Nutzer selbst implementiert werden.

Außerdem gibt es noch eine Reihe von Elementen, welche vor allem für die Topologie und die Darstellung von Bedeutung sind.

Verbindungen

Jede Verbindung ist gerichtet, d.h. sie hat eine Quelle und ein Ziel. Des weiteren wird jeder Verbindung ein Gewicht zugeordnet, welches die Netzeingabe des Ziels beeinflußt.

Units können auf zwei verschiedene Arten verbunden sein, entweder direkt oder über Sites. Bei direkten Verbindungen erhalten dieselben ein Gewicht, wie oben beschrieben. Bezieht eine Unit ihre Aktivierungen über Sites, so sind die Verbindungen von diesen zur Unit selbst nicht mit Gewichten versehen.

Sites

Sites fassen die gewichteten Eingaben von mehreren Vorgängerunits zusammen. Die dafür verwendete Site-Funktion ist ebenfalls durch den Nutzer implementierbar. Die Ergebnisse dieser Funktionen werden als Site-Values bezeichnet und dienen den nachfolgenden Units als Netzeingabe.

Generell sind parallele Verbindungen im Netz von einer Quelle zu einem Ziel nicht zugelassen, weil redundant. Es ist jedoch erlaubt, Verbindungen von einer Unit zu mehreren Sites einer anderen Unit herzustellen.

Musterpropagation

Die Ausgaben der einzelnen Units werden nicht ständig aktualisiert. Dies wäre für eine Visualisierung von Zwischenschritten auch gar nicht erwünscht. Der Simulatorkern stellt verschiedene Funktionen zur Verfügung, einzelne Werte zu berechnen. Es gibt jedoch auch Propagationsfunktionen, welche die Neuberechnung des gesamten Netzes zusammenfassen. Sie aktualisieren die Neuronen z.B. quasiparallel oder in einer zufälligen oder einer festen Ordnung. Auch sie können vom Nutzer ergänzt werden.

Lernen in neuronalen Netzen

Das Lernen in neuronalen Netzenbesteht in der Regel aus zwei Phasen:

1. Anlegen eines Muster und berechnen der Netzausgabe (Vorwärtspropagation)
2. Berechnung der Fehler der einzelnen Neuronen und der Gewichtsänderungen (Rückwärtspropagation)

Bei den Online-Lernverfahren werden die Gewichte nach jedem einzelnen Muster geändert. Bei den Batchverfahren hingegen werden die Gewichtsänderungen erst nach dem Anlegen eines jeden Musters aus der Lernmenge ausgeführt. Deshalb werden die Änderungen, welche durch die einzelnen Muster entstehen, in einer Datei zwischengespeichert.

Auch die Lernfunktion ist benutzerimplementierbar.

6.3 Implementation von Q-Neuronen im SNNS

Implementation des Q-Neurons

Wenn man sich mit der Implementation des Q-Neurons im SNNS beschäftigt, fällt als erstes auf, daß die zwei Gewichtsvektoren des Neurons nicht in dieser Art und Weise in die Unit-Struktur des SNNS übernommen werden können. Jeder dieser beiden Gewichtsvektoren muß auf ein Site abgebildet werden, er ist mit der Eingabeschicht vollständig zu vernetzen. Ihre Site-Funktionen berechnen das Skalarprodukt aus dem Eingabevektor mit dem jeweiligen Gewichtsvektor. Die Aktivierungsfunktion muß dann die Aktivierung des Neurons als Quotienten der beiden Site-Values berechnen.

Die für das Lernen von hybriden Netzen notwendigen Hilfsmustervektoren $\vec{x}^{(1)}$ und $\vec{x}^{(2)}$ sowie die Zugehörigkeit der Muster zur jeweiligen Mustermenge müssen ebenfalls in jedem Neuron gespeichert werden. Dazu kann man die Elemente *value_a, value_b,* und *value_c* verwenden, welche in jeder Unit für die allgemeine Verwendung in der Lernfunktion zur Verfügung stehen. Dabei wird sicherlich ein programmiertechnischer Kunstgriff erforderlich sein, um die entsprechenden Datenmengen anzubinden. Zur Speicherung der jeweiligen Daten sollte man unter Umständen auf ein ähnliches Prinzip wie das des SNNS-Speichermanagements zurückgreifen, um eine zu starke Fragmentierung des Hauptspeichers zu vermeiden und effiziente Zugriffsverfahren zur Verfügung zu stellen.

Implementation des Lernverfahrens für hybride Netze

Die Lernfunktion kann prinzipiell direkt aus dem in Abschnitt 5.2 beschriebenen Lernverfahren hergeleitet werden. Wird es als Batchverfahren implementiert, so sind die einzelnen Korrekturterme eventuell in einer Datei zwischenzuspeichern.

7 Zusammenfassung und Ausblick

Es ist gelungen, ein einzelnes Q-Neuron für einige Klassifizierungsaufgaben einzusetzen. Des weiteren wurde eine Netzstruktur und ein Verfahren hergeleitet, mit welchem unter Verwendung von Q-Neuronen prinzipiell beliebige Lernaufgaben automatisch bearbeitet werden können.

Die Konvergenzgeschwindigkeit des Verfahrens war im Vergleich zum Backpropagation-Verfahren leider nicht wesentlich besser. Hauptursache dafür sind vor allem die wesentlich komplexeren Korrekturschritte. Erschwerend kommen teilweise starke Oszillationen im Netzwerk hinzu.

Es sei an dieser Stelle noch einmal darauf hingewiesen, daß das Hauptziel dieser Arbeit die Untersuchung des Lernverfahrens war. Dabei diente die Ziffernerkennung nur als ein Beispiel, um die Probleme zu verdeutlichen. Wie sich herausgestellt hat, ist das hergeleitete Lernverfahren in dieser Form für die Mustererkennung auf der Basis von Pixeldaten nicht sinnvoll einsetzbar. Jedoch gibt es noch einige Ideen, die auf diesem Gebiet noch eine wesentliche Verbesserung herbeiführen könnten.

Für die zukünftige Arbeit gibt es folgende Hauptgebiete, deren Untersuchung eine erfolgreiche Verbesserung des Verfahrens erhoffen lassen:

1. **Problemklassen:** Das in dieser Arbeit benutzte Beispiel der Mustererkennung auf der Grundlage von Grauwertbildern ist zwar sehr praktisch, um die Lernergebnisse zu beobachten, es kommt jedoch der Arbeitsweise der Q-Neuronen wahrscheinlich nicht sehr entgegen. Es wäre von Interesse, das Verhalten bei anderen Mustertypen, z.B. fouriertransformierten Bildern zu untersuchen. Außerdem könnte eine höhere Dimension der Muster das Zeitverhalten des Lernens von Q-Neuronen anders beeinflussen, als das von Perzeptrons.

2. **Korrekturterme:** In dieser Arbeit konnte nur mit Korrekturformeln der Stufe 1 sinnvolle Ergebnisse erzielt werden. Eine Modifikation der anderen Formeln, insbesondere unter Einbeziehung der Gewichtsberechnungsfunktion $W(x)$ könnte ebenfalls noch weitere Verbesserungen bringen. Auch wäre es eine interessante Aufgabe, den Einsatz von solchen Formeln beim Lernen eines Netzes mit Q-Neuronen (in dem die Werte aus vergangenen Korrekturschritten nicht zuverlässig sind) zu ermöglichen.

3. **Netzlernen:** Die in Abschnitt 5.4 angedachten Verbesserungsmöglichkeiten für das Anlernen von Netzwerken scheinen ebenfalls Erfolgsmöglichkeiten zu bieten. Insbesondere ist hier auch über die Einbeziehung von Standardverfahren zur Konvergenzverbesserung nachzudenken.

Dabei versprechen die Methoden, welche das Lernen von größeren Lernmengen ermöglichen sollen (vorherige Clusterung der Lernmenge, MLP als nachgeschaltete Schicht), wahrscheinlich den größten Erfolg.

A Lernkurven

Die im folgenden dargestellten Graphen von Lernkurven wurden mit einem selbstgeschriebenen Visualisierungstool erstellt und mit einem Framegrabber als Bilder gespeichert. Mit dem Visualisierungstool lassen sich die Werteverläufe von einzelnen Variablen darstellen. Die Skalierung der Y-Achse erfolgt dabei automatisch. Es ist möglich, den gesamten Definitionsbereich in eine Anzahl von gleich großen Sektionen einzuteilen, diese sind in der Darstellung jeweils durch einen senkrechten Strich getrennt. Die Kurven können von einem beliebigen Punkt bis zu Anfang der nächsten Sektion springen. Von dieser Möglichkeit wurde Gebrauch gemacht, um die Äußeren Lernschleifen des IWD-Verfahrens voneinander zu trennen.

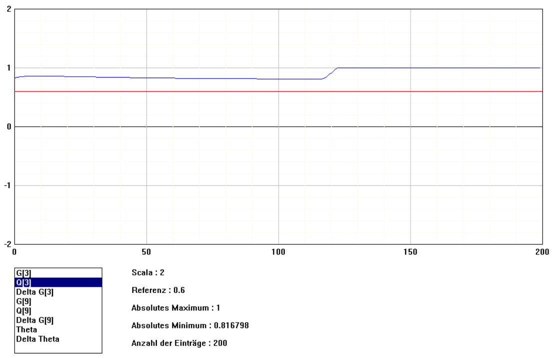

Abbildung 9: *Verlauf von Q_3 beim Lernen der Muster 3 und 9 mit Korrekturtermen der Stufe 1.*

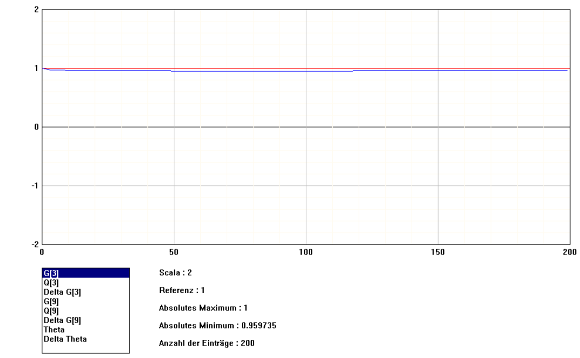

Abbildung 10: *Verlauf von g_3 beim Lernen der Muster 3 und 9 mit Korrekturtermen der Stufe 1.*

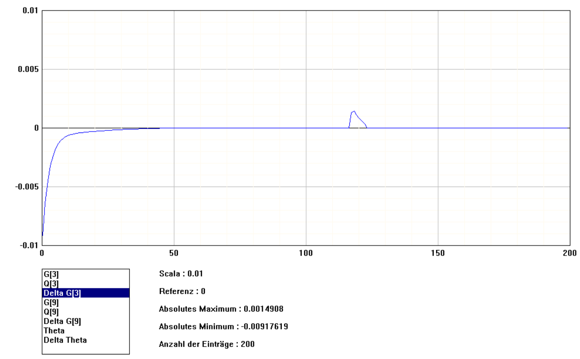

Abbildung 11: *Verlauf von Δg_3 beim Lernen der Muster 3 und 9 mit Korrekturtermen der Stufe 1.*

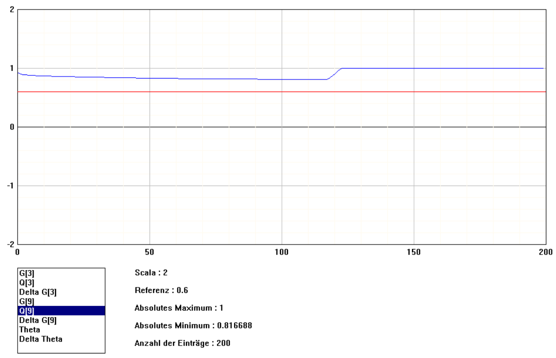

Abbildung 12: *Verlauf von Q_9 beim Lernen der Muster 3 und 9 mit Korrekturtermen der Stufe 1.*

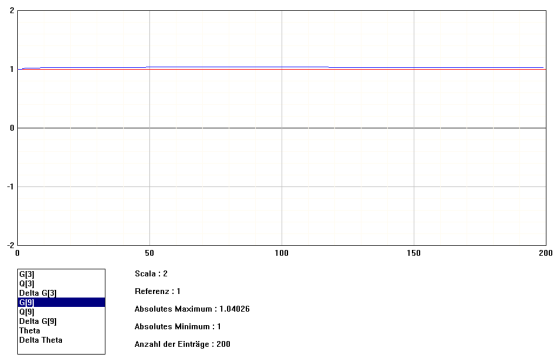

Abbildung 13: *Verlauf von g_9 beim Lernen der Muster 3 und 9 mit Korrekturtermen der Stufe 1.*

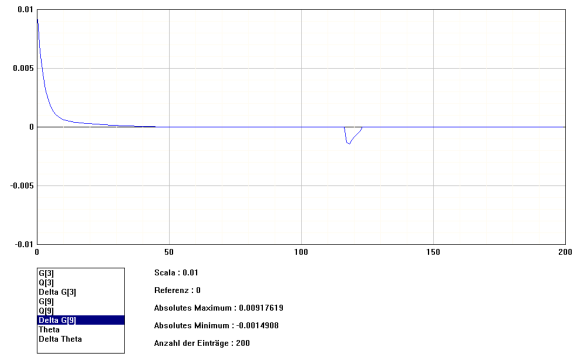

Abbildung 14: *Verlauf von Δg_9 beim Lernen der Muster 3 und 9 mit Korrekturtermen der Stufe 1.*

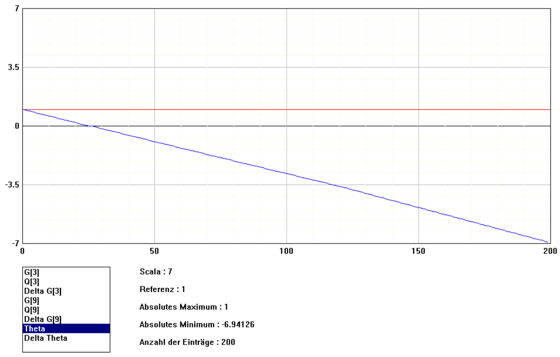

Abbildung 15: *Verlauf von Θ beim Lernen der Muster 3 und 9 mit Korrekturtermen der Stufe 1.*

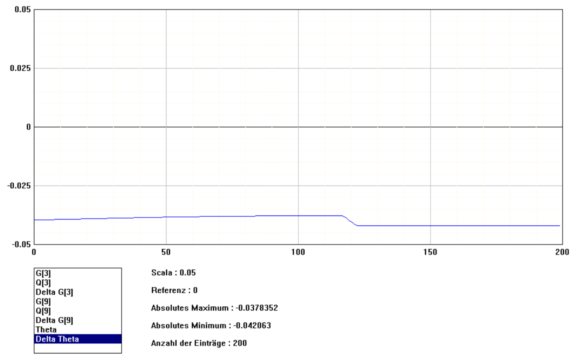

Abbildung 16: *Verlauf von $\Delta\Theta$ beim Lernen der Muster 3 und 9 mit Korrekturtermen der Stufe 1.*

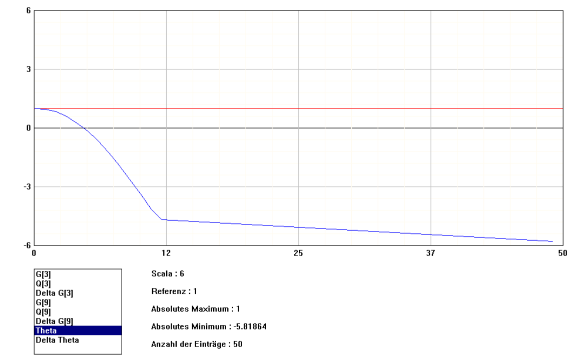

Abbildung 17: *Verlauf von Θ beim Lernen der Muster 3 und 9 mit Korrekturtermen der Stufe 4.*

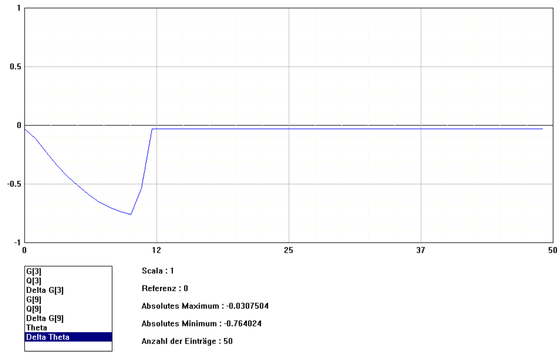

Abbildung 18: *Verlauf von $\Delta\Theta$ beim Lernen der Muster 3 und 9 mit Korrekturtermen der Stufe 4.*

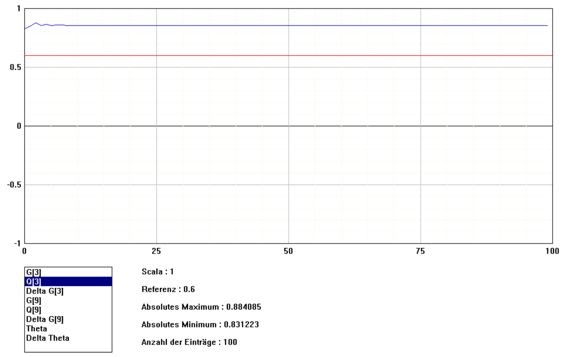

Abbildung 19: *Verlauf von Q_3 beim Lernen der Muster 3 und 9 mit Korrekturtermen der Stufe 6.*

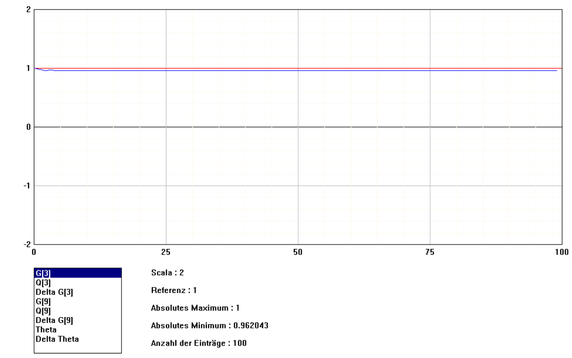

Abbildung 20: *Verlauf von g_3 beim Lernen der Muster 3 und 9 mit Korrekturtermen der Stufe 6.*

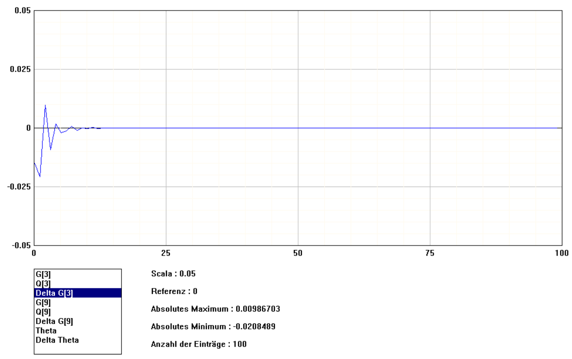

Abbildung 21: *Verlauf von Δg_3 beim Lernen der Muster 3 und 9 mit Korrekturtermen der Stufe 6.*

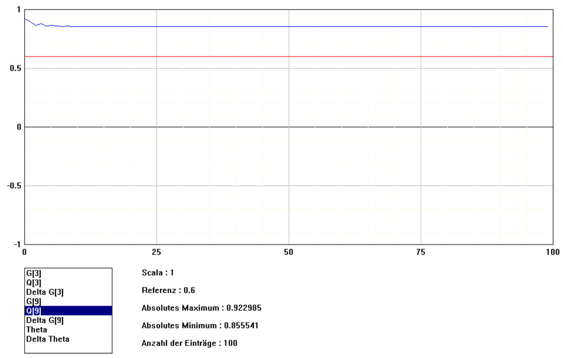

Abbildung 22: *Verlauf von Q_9 beim Lernen der Muster 3 und 9 mit Korrekturtermen der Stufe 6.*

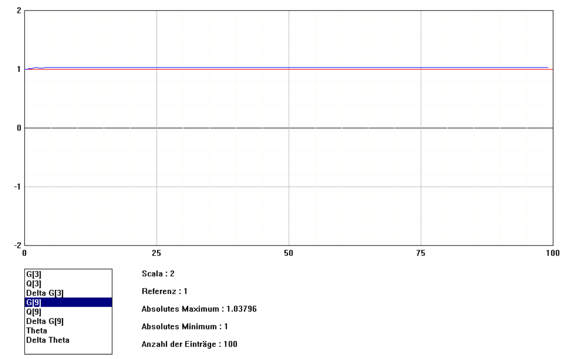

Abbildung 23: *Verlauf von g_9 beim Lernen der Muster 3 und 9 mit Korrekturtermen der Stufe 6.*

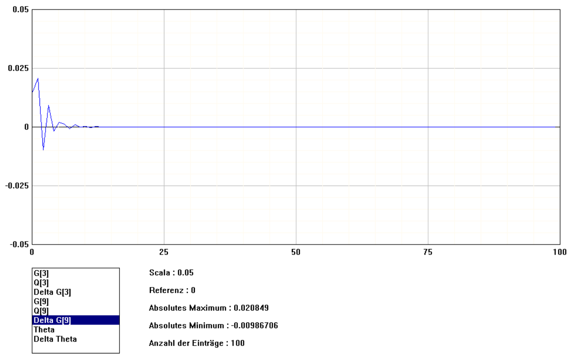

Abbildung 24: *Verlauf von Δg_9 beim Lernen der Muster 3 und 9 mit Korrekturtermen der Stufe 6.*

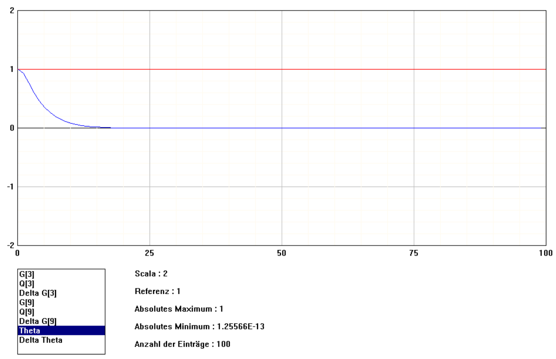

Abbildung 25: *Verlauf von Θ beim Lernen der Muster 3 und 9 mit Korrekturtermen der Stufe 6.*

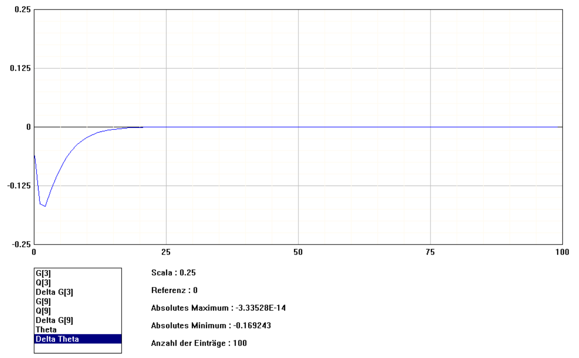

Abbildung 26: *Verlauf von $\Delta\Theta$ beim Lernen der Muster 3 und 9 mit Korrekturtermen der Stufe 6.*

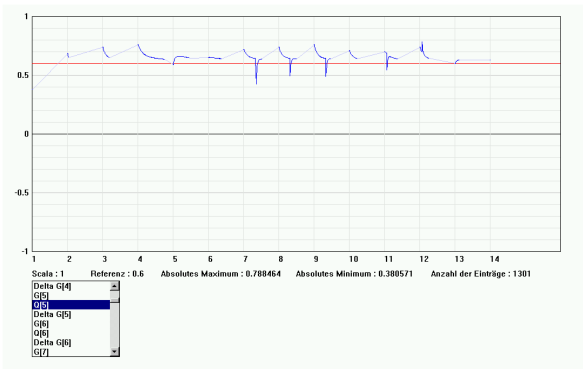

Abbildung 27: *Verlauf von Q_5 beim Lernen der Muster der Klassen 0 und 6 mit Korrekturtermen der Stufe 1.*

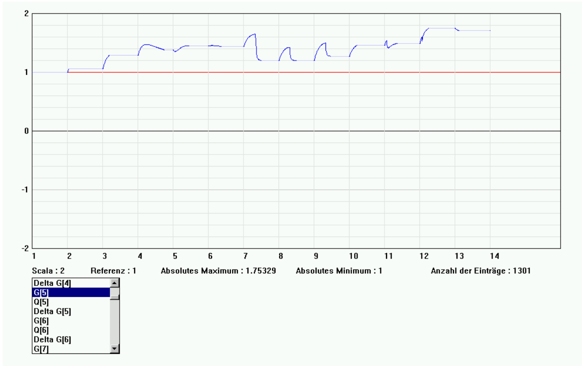

Abbildung 28: *Verlauf von g_5 beim Lernen der Muster der Klassen 0 und 6 mit Korrekturtermen der Stufe 1.*

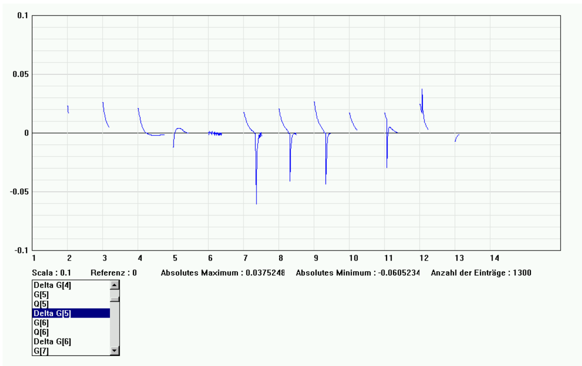

Abbildung 29: *Verlauf von Δg_5 beim Lernen der Muster der Klassen 0 und 6 mit Korrekturtermen der Stufe 1.*

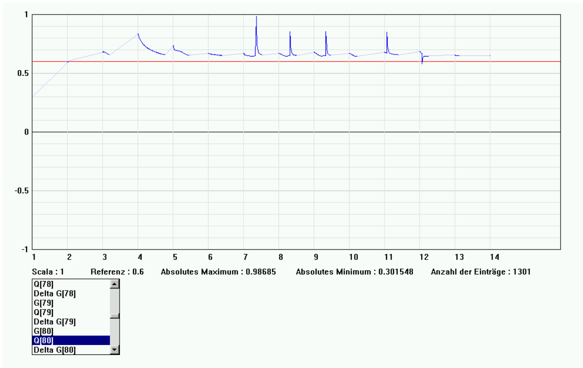

Abbildung 30: *Verlauf von Q_{80} beim Lernen der Muster der Klassen 0 und 6 mit Korrekturtermen der Stufe 1.*

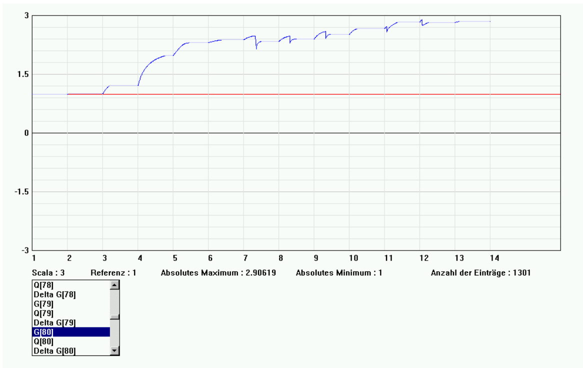

Abbildung 31: *Verlauf von g_{80} beim Lernen der Muster der Klassen 0 und 6 mit Korrekturtermen der Stufe 1.*

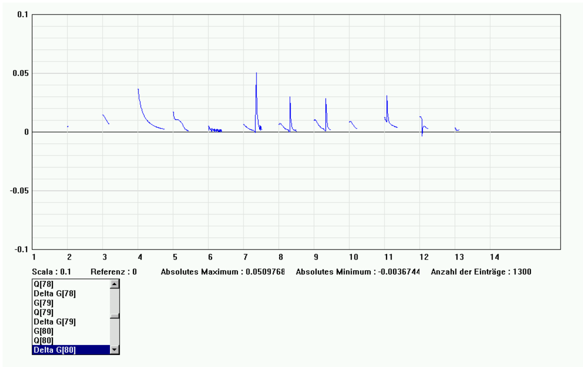

Abbildung 32: *Verlauf von* Δg_{80} *beim Lernen der Muster der Klassen 0 und 6 mit Korrekturtermen der Stufe 1.*

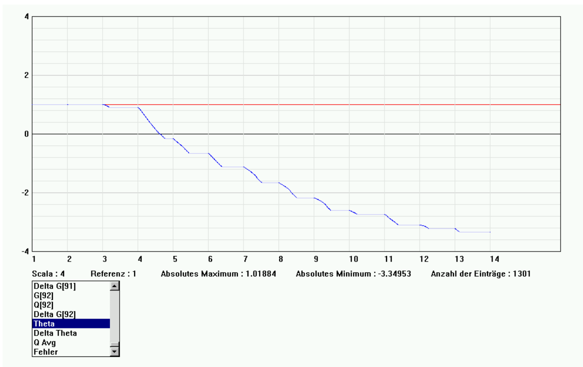

Abbildung 33: *Verlauf von* Θ *beim Lernen der Muster der Klassen 0 und 6 mit Korrekturtermen der Stufe 1.*

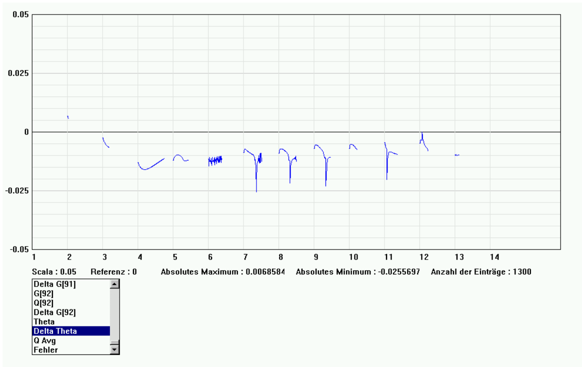

Abbildung 34: *Verlauf von $\Delta\Theta$ beim Lernen der Muster der Klassen 0 und 6 mit Korrekturtermen der Stufe 1.*

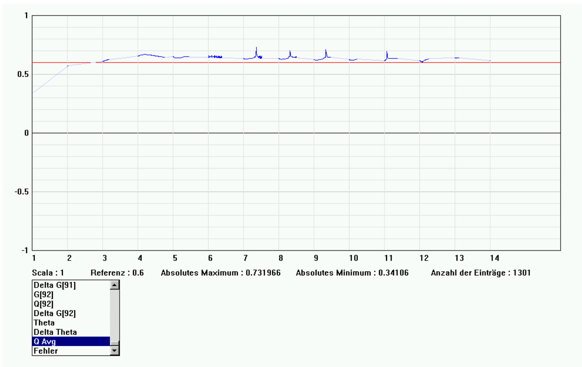

Abbildung 35: *Verlauf von $\bar{Q}$ beim Lernen der Muster der Klassen 0 und 6 mit Korrekturtermen der Stufe 1.*

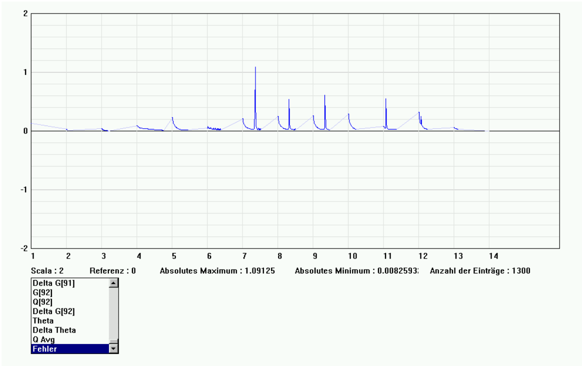

Abbildung 36: *Verlauf des Gesamtfehlers beim Lernen der Muster der Klassen 0 und 6 mit Korrekturtermen der Stufe 1.*

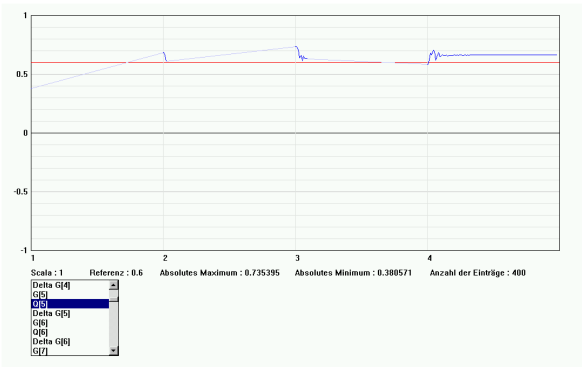

Abbildung 37: *Verlauf von Q_5 beim Lernen der Muster der Klassen 0 und 6 mit Korrekturtermen der Stufe 6.*

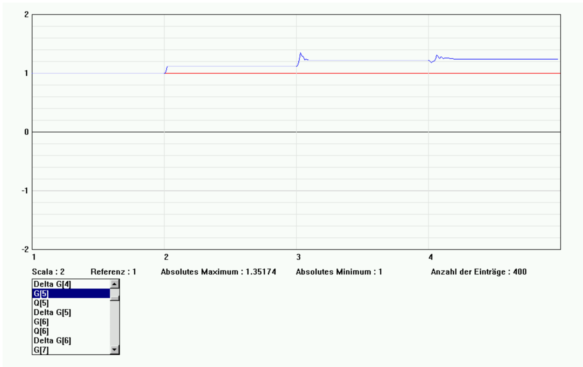

Abbildung 38: *Verlauf von g_5 beim Lernen der Muster der Klassen 0 und 6 mit Korrekturtermen der Stufe 6.*

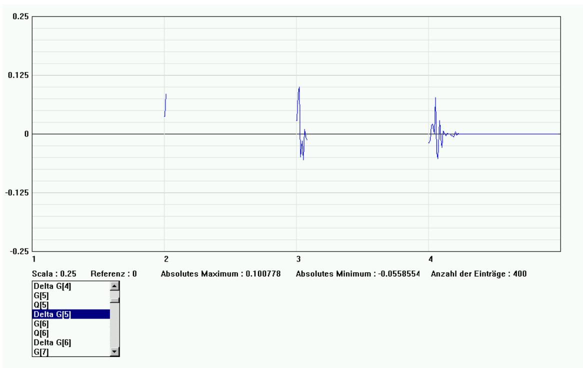

Abbildung 39: *Verlauf von Δg_5 beim Lernen der Muster der Klassen 0 und 6 mit Korrekturtermen der Stufe 6.*

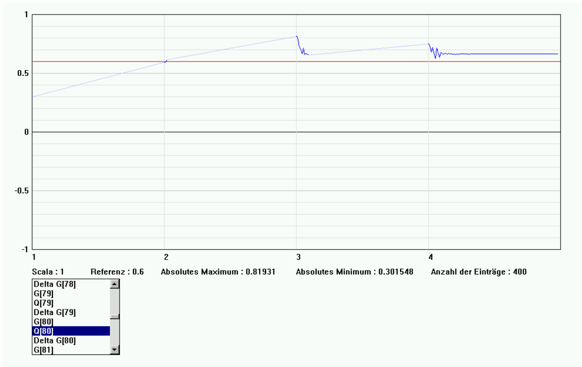

Abbildung 40: *Verlauf von Q_{80} beim Lernen der Muster der Klassen 0 und 6 mit Korrekturtermen der Stufe 6.*

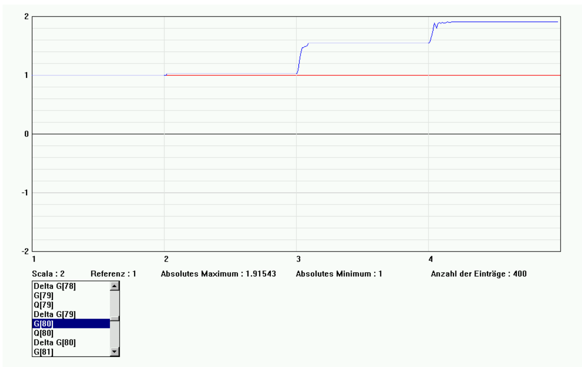

Abbildung 41: *Verlauf von g_{80} beim Lernen der Muster der Klassen 0 und 6 mit Korrekturtermen der Stufe 6.*

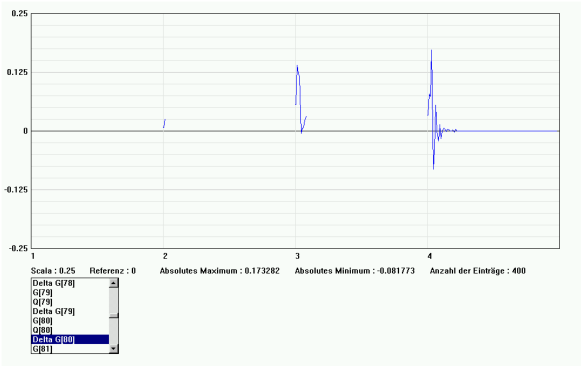

Abbildung 42: *Verlauf von* Δg_{80} *beim Lernen der Muster der Klassen 0 und 6 mit Korrekturtermen der Stufe 6.*

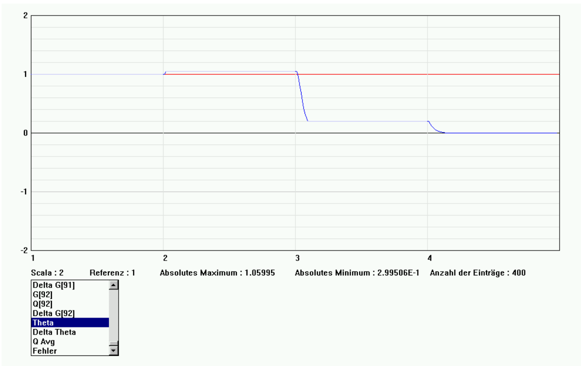

Abbildung 43: *Verlauf von* Θ *beim Lernen der Muster der Klassen 0 und 6 mit Korrekturtermen der Stufe 6.*

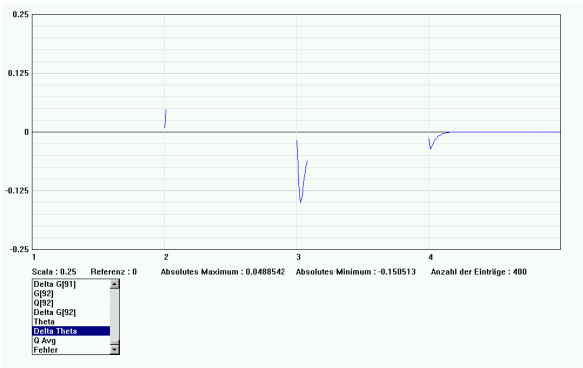

Abbildung 44: *Verlauf von* $\Delta\Theta$ *beim Lernen der Muster der Klassen 0 und 6 mit Korrekturtermen der Stufe 6.*

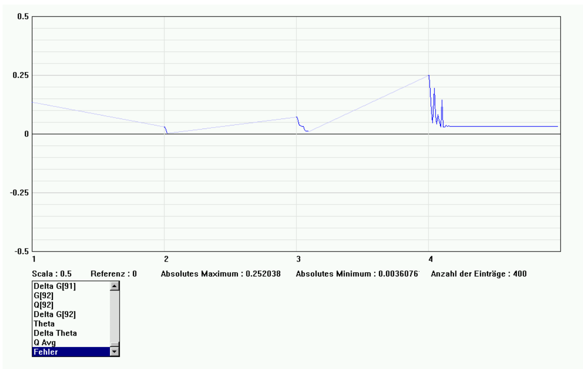

Abbildung 45: *Verlauf des Gesamtfehlers beim Lernen der Muster der Klassen 0 und 6 mit Korrekturtermen der Stufe 6.*

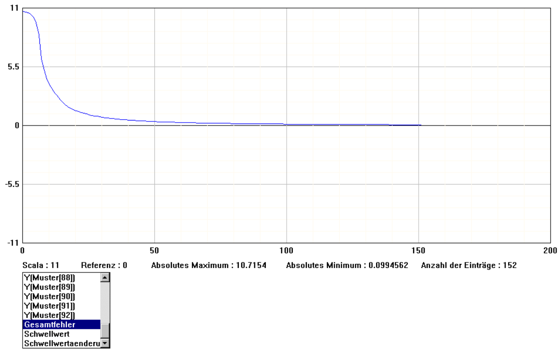

Abbildung 46: *Verlauf des Gesamtfehlers beim Lernen der Muster der Klassen 0 und 6 mit einem Perzeptron.*

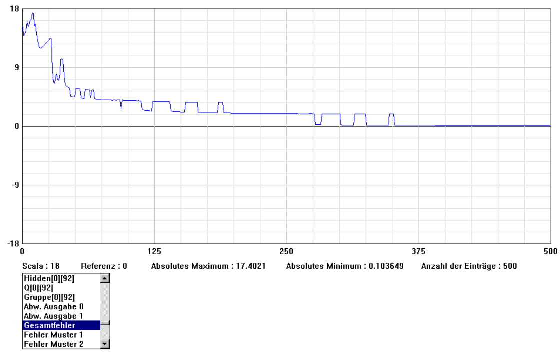

Abbildung 47: *Verlauf des Gesamtfehlers beim Lernen der Muster der Klassen 0 und 6 mit der Präsentationsstrategie Round Robin.*

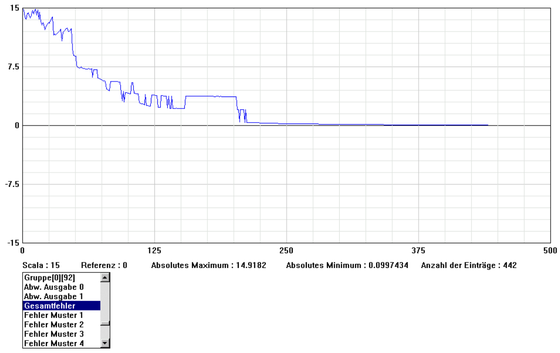

Abbildung 48: *Verlauf des Gesamtfehlers beim Lernen der Muster der Klassen 0 und 6 mit der Präsentationsstrategie Random.*

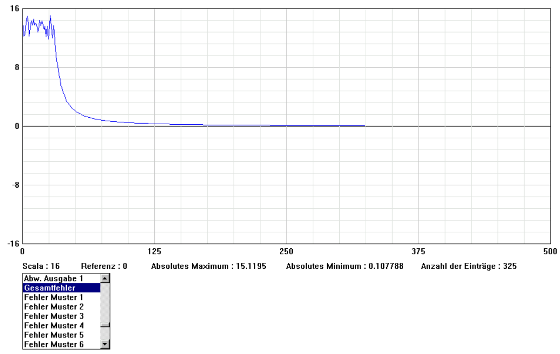

Abbildung 49: *Verlauf des Gesamtfehlers beim Lernen der Muster der Klassen 0 und 6 mit der Präsentationsstrategie Worst Pattern Next.*

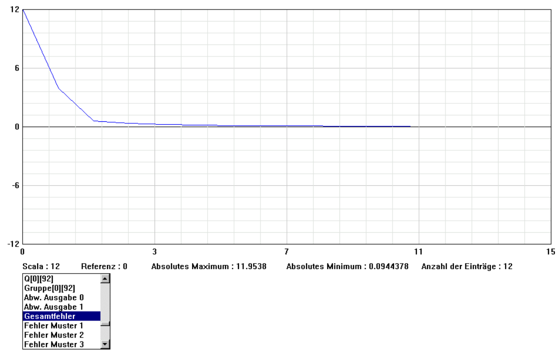

Abbildung 50: *Verlauf des Gesamtfehlers beim Lernen der Muster der Klassen 0 und 6 im Batch-Verfahren.*

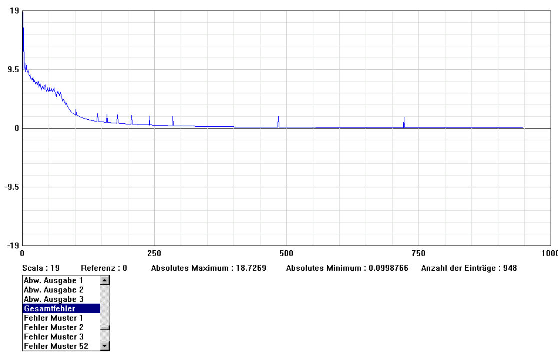

Abbildung 51: *Verlauf des Gesamtfehlers beim Lernen der Muster der Klassen 0, 4, 6 und 7 mit der Präsentationsstrategie Worst Pattern Next.*

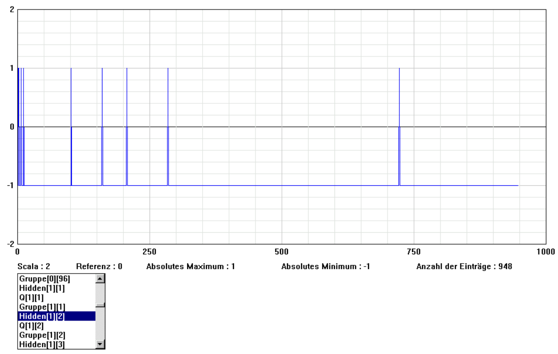

Abbildung 52: *Verlauf der Ausgabe des Neurons H_1 bei angelegtem Muster 2 mit der Präsentationsstrategie Worst Pattern Next.*

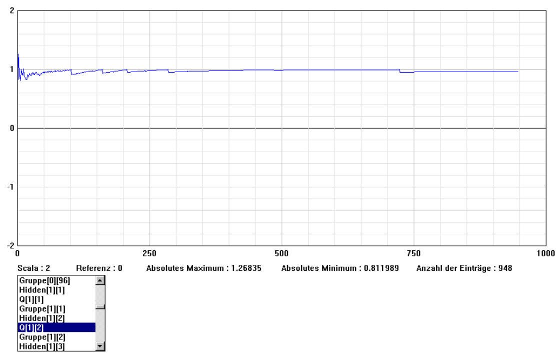

Abbildung 53: *Verlauf des Quotienten des Neurons H_1 bei angelegtem Muster 2 mit der Präsentationsstrategie Worst Pattern Next.*

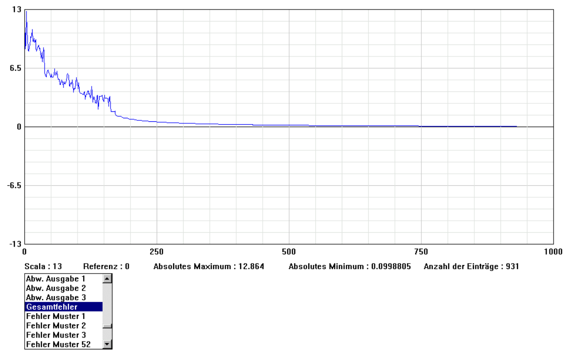

Abbildung 54: *Verlauf des Gesamtfehlers beim Lernen der Muster der Klassen 0, 4, 6 und 7 mit der Präsentationsstrategie Round Robin.*

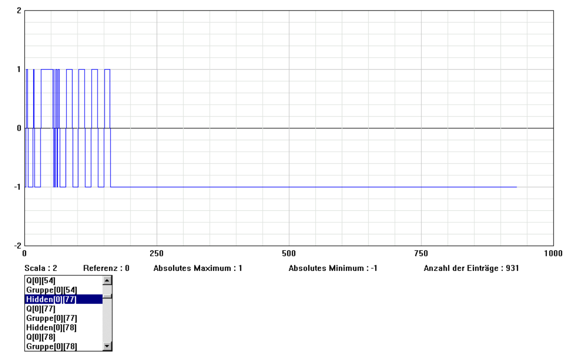

Abbildung 55: *Verlauf der Ausgabe des Neurons H_0 bei angelegtem Muster 77 mit der Präsentationsstrategie Round Robin.*

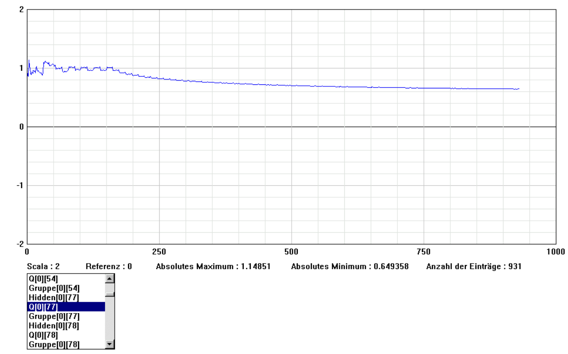

Abbildung 56: *Verlauf des Quotienten des Neurons H_0 bei angelegtem Muster 77 mit der Präsentationsstrategie Round Robin.*

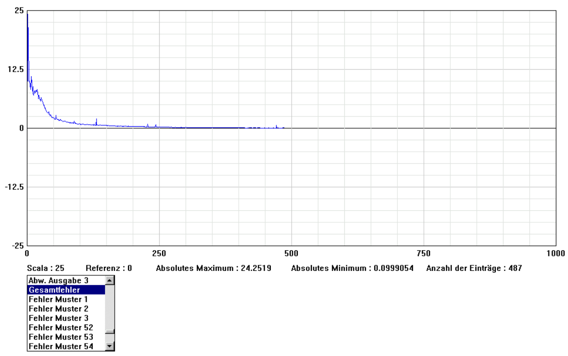

Abbildung 57: *Verlauf des Gesamtfehlers beim Lernen der Muster der Klassen 0, 4, 6 und 7 mit der Präsentationsstrategie Worst Pattern Next mit 4 Q-Neuronen.*

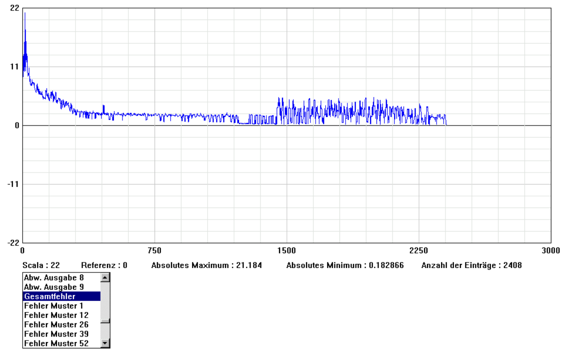

Abbildung 58: *Verlauf des Gesamtfehlers beim Lernen eines Muster jeder Klasse mit der Präsentationsstrategie Round Robin.*

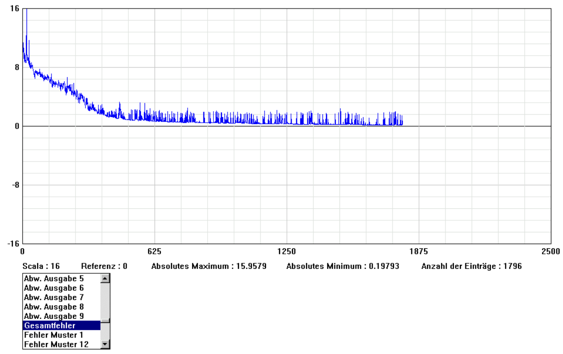

Abbildung 59: *Verlauf des Gesamtfehlers beim Lernen eines Muster jeder Klasse mit der Präsentationsstrategie Worst Pattern Next.*

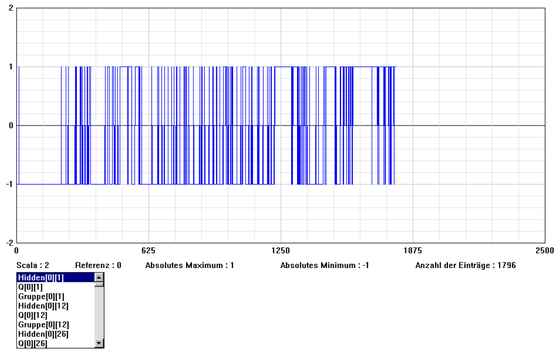

Abbildung 60: *Verlauf der Ausgabe des Neurons H_0 bei angelegtem Muster 1.*

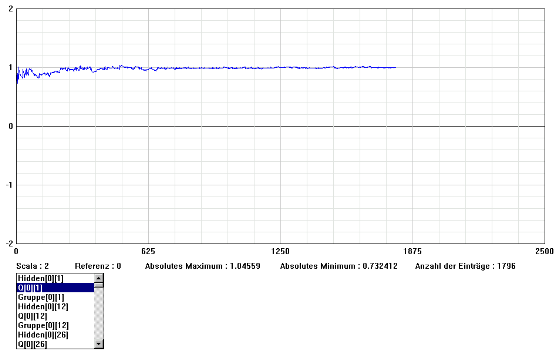

Abbildung 61: *Verlauf des Quotienten des Neurons H_0 bei angelegtem Muster 1.*

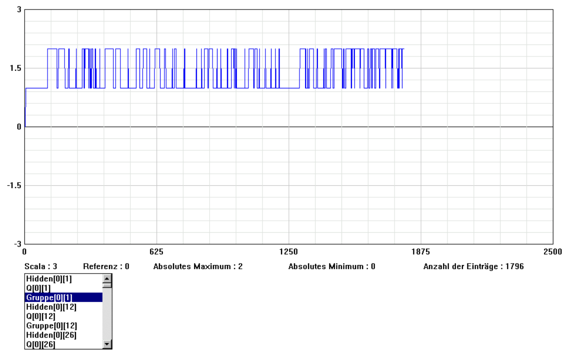

Abbildung 62: *Verlauf der Mengenzugehörigkeit des Musters 1 für Neuron H_0.*

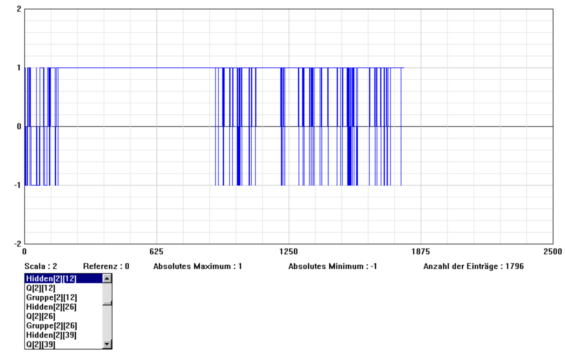

Abbildung 63: *Verlauf der Ausgabe des Neurons H_2 bei angelegtem Muster 12.*

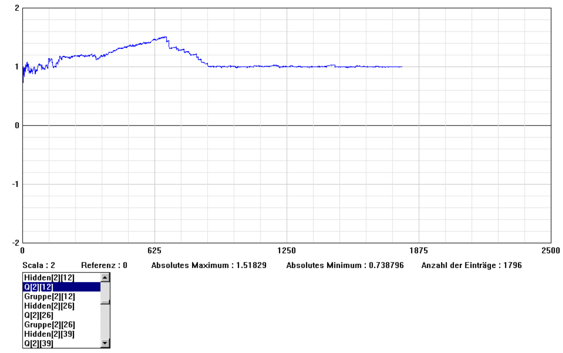

Abbildung 64: *Verlauf des Quotienten des Neurons H_2 bei angelegtem Muster 12.*

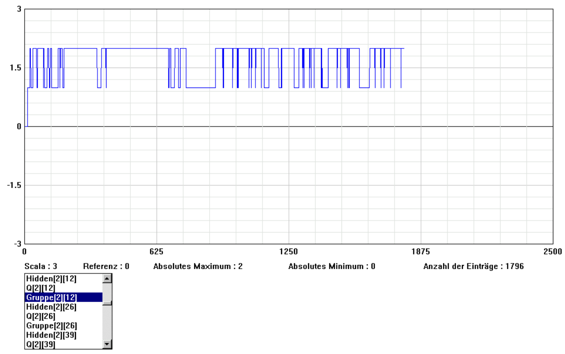

Abbildung 65: *Verlauf der Mengenzugehörigkeit des Musters 12 für Neuron H_2.*

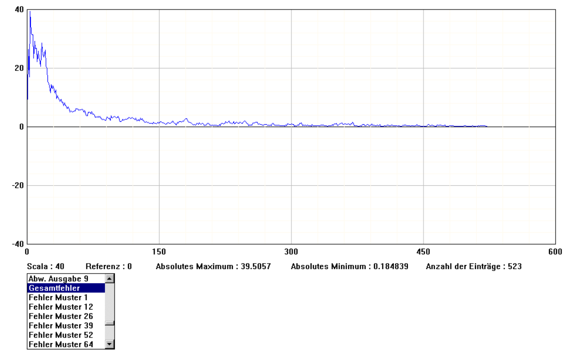

Abbildung 66: *Verlauf des Gesamtfehlers beim Lernen eines Muster jeder Klasse mit der Präsentationsstrategie Round Robin mit 8 Q-Neuronen.*

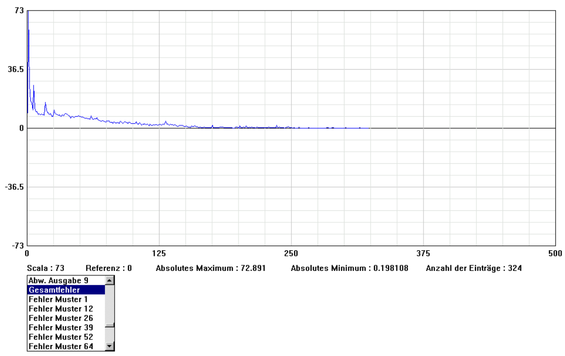

Abbildung 67: *Verlauf des Gesamtfehlers beim Lernen eines Muster jeder Klasse mit der Präsentationsstrategie Worst Pattern Next mit 8 Q-Neuronen.*

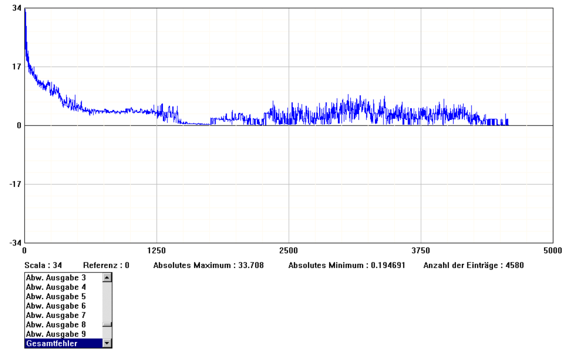

Abbildung 68: *Verlauf des Gesamtfehlers beim Lernen von zwei Mustern jeder Klasse mit der Präsentationsstrategie Round Robin.*

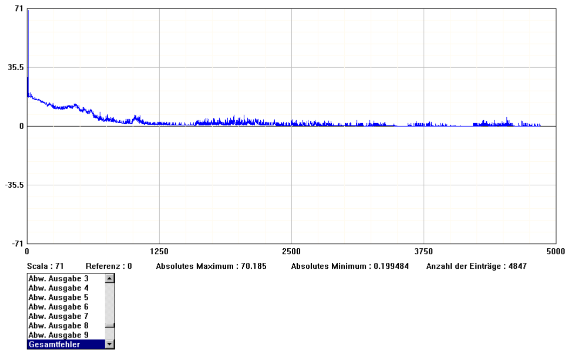

Abbildung 69: *Verlauf des Gesamtfehlers beim Lernen von zwei Mustern jeder Klasse mit der Präsentationsstrategie Worst Pattern Next.*

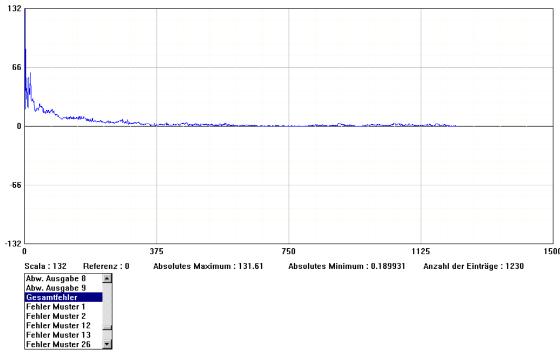

Abbildung 70: *Verlauf des Gesamtfehlers beim Lernen von zwei Mustern jeder Klasse mit der Präsentationsstrategie Round Robin mit 8 Q-Neuronen.*

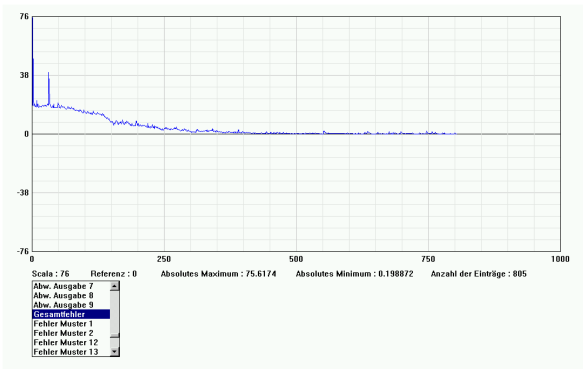

Abbildung 71: *Verlauf des Gesamtfehlers beim Lernen von zwei Mustern jeder Klasse mit der Präsentationsstrategie Worst Pattern Next mit 8 Q-Neuronen.*

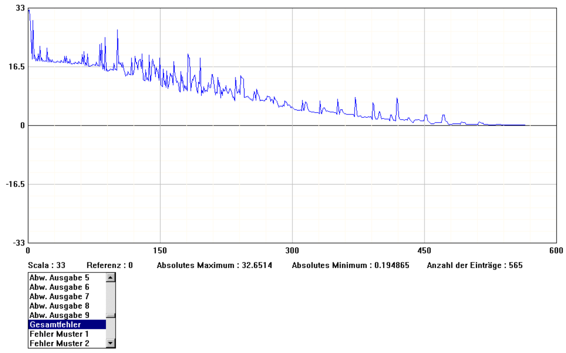

Abbildung 72: *Verlauf des Gesamtfehlers beim Backpropagation-Lernen von zwei Mustern jeder Klasse mit der Präsentationsstrategie Round Robin mit 8 Perzeptrons in der verdeckten Schicht.*

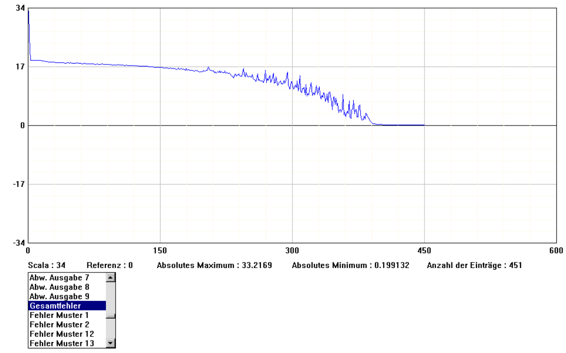

Abbildung 73: *Verlauf des Gesamtfehlers beim Backpropagation-Lernen von zwei Mustern jeder Klasse mit der Präsentationsstrategie Worst Pattern Next mit 8 Perzeptrons in der verdeckten Schicht.*

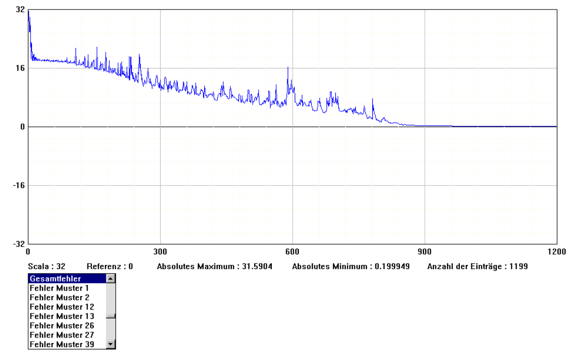

Abbildung 74: *Verlauf des Gesamtfehlers beim Backpropagation-Lernen von zwei Mustern jeder Klasse mit der Präsentationsstrategie Round Robin mit 4 Perzeptrons in der verdeckten Schicht.*

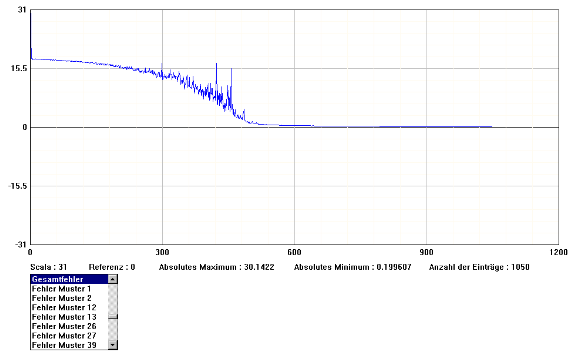

Abbildung 75: *Verlauf des Gesamtfehlers beim Backpropagation-Lernen von zwei Mustern jeder Klasse mit der Präsentationsstrategie Worst Pattern Next mit 4 Perzeptrons in der verdeckten Schicht.*

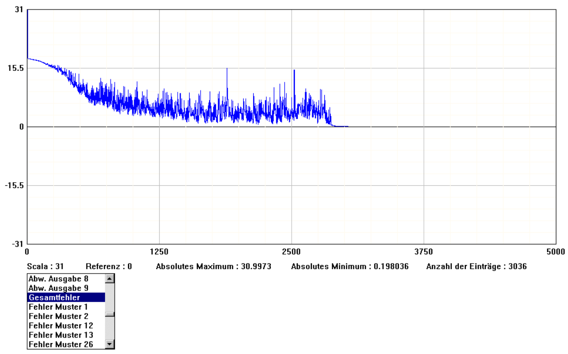

Abbildung 76: *Verlauf des Gesamtfehlers beim Backpropagation-Lernen von zwei Mustern jeder Klasse mit der Präsentationsstrategie Worst Pattern Next mit 3 Perzeptrons in der verdeckten Schicht.*

B Erstes Beispiel - Ziffernprototypen

Die hier dargestellten Muster wurden mit einem Mustereditor als binäre 8x8 Bilder erzeugt. Sie sind als Datei `DIGITS.PAT` über den Autor verfügbar.

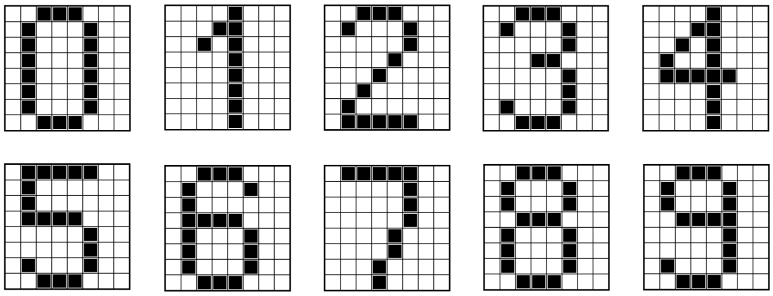

Abbildung 77: *Muster in DIGITS.PAT*

C Zweites Beispiel - Ziffern aus Postleitzahlen

Die hier dargestellten Muster wurden aus einem speziellen Dateiformat konvertiert. Sie liegen als 16x16 Bilder mit 256 Graustufen vor. Sie sind als Datei `PLZ.PAT` ebenfalls über den Autor vefügbar.

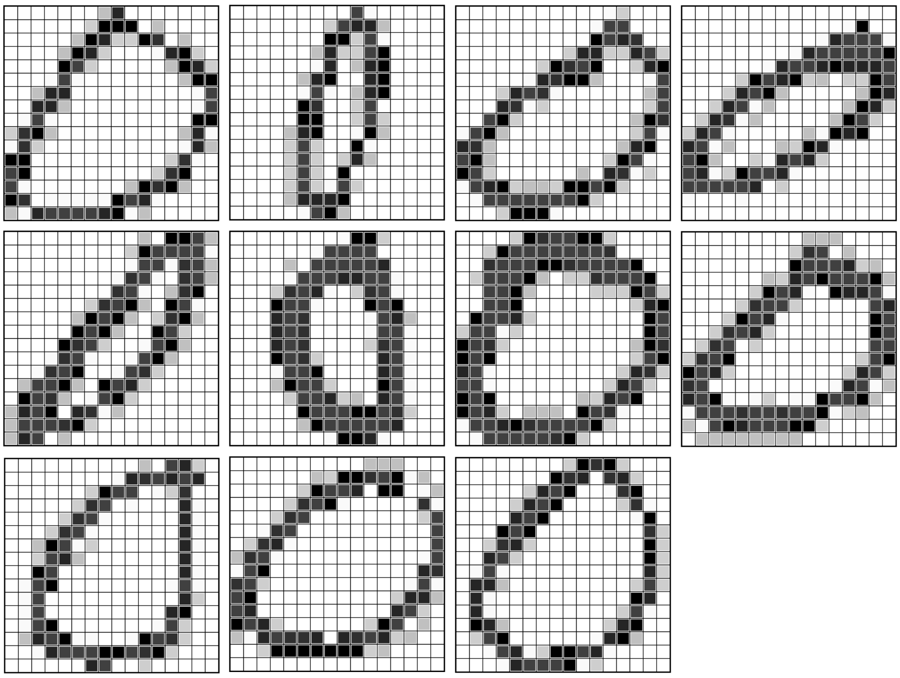

Abbildung 78: *Muster in PLZ.PAT, Klasse 0, Muster Nr. 1 bis 11*

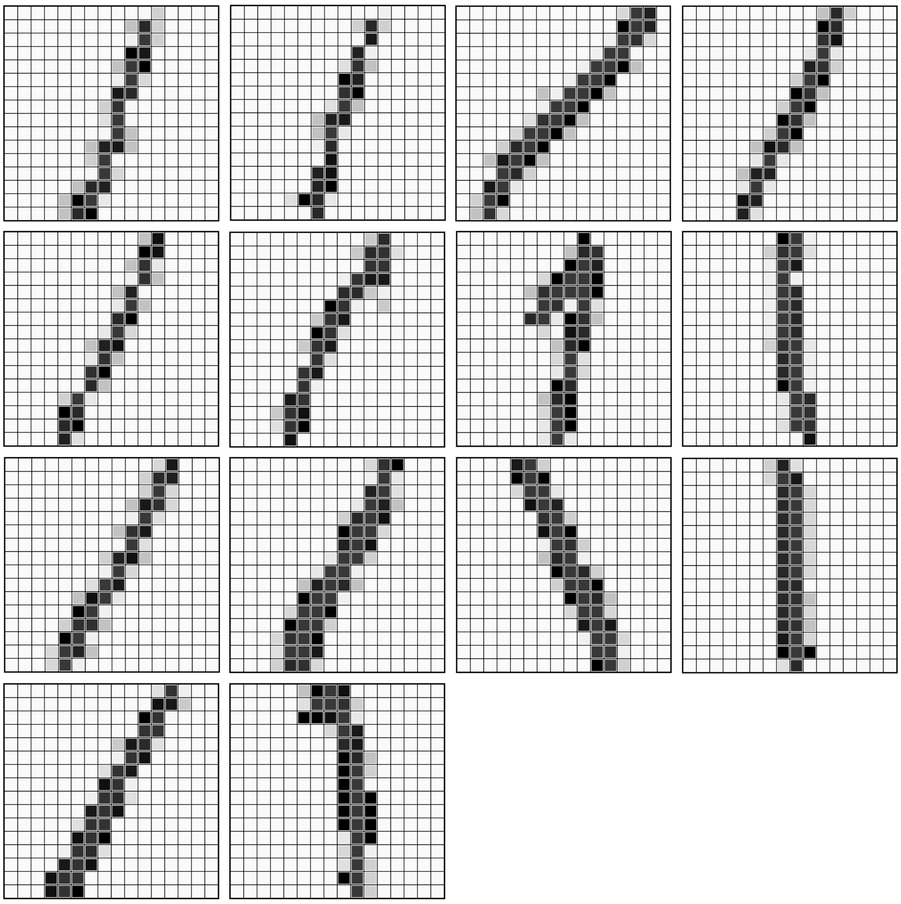

Abbildung 79: *Muster in PLZ.PAT, Klasse 1, Muster Nr. 12 bis 25*

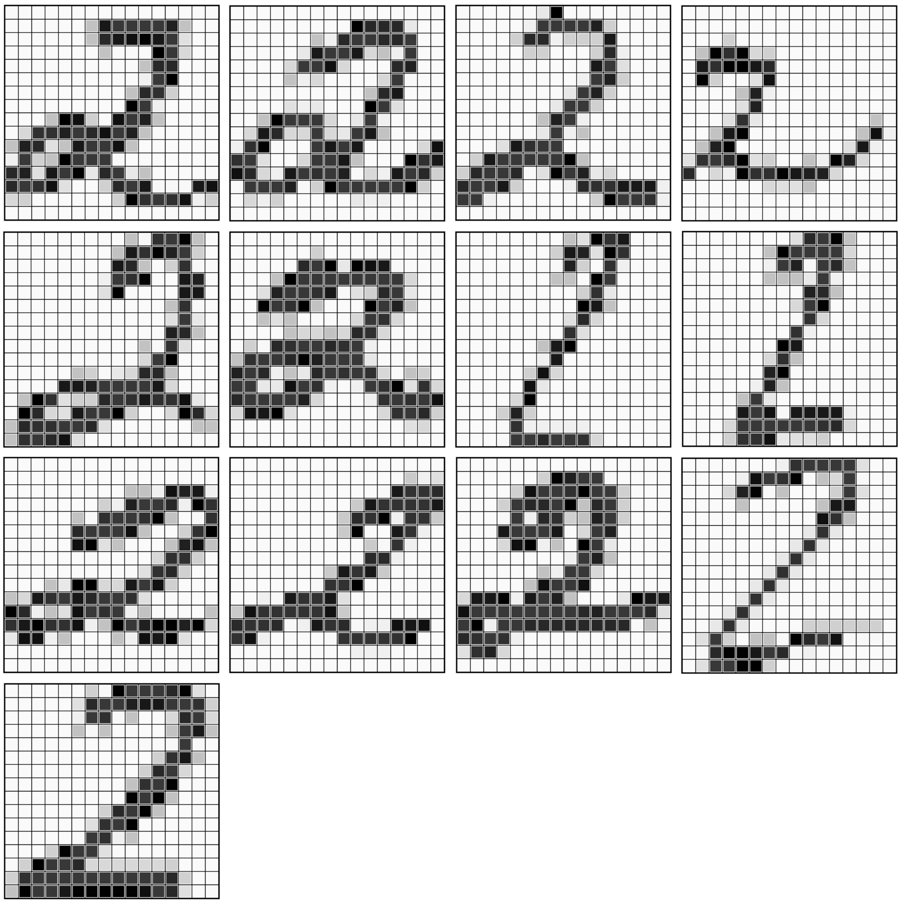

Abbildung 80: *Muster in PLZ.PAT, Klasse 2, Muster Nr. 26 bis 38*

Abbildung 81: *Muster in PLZ.PAT, Klasse 3, Muster Nr. 39 bis 51*

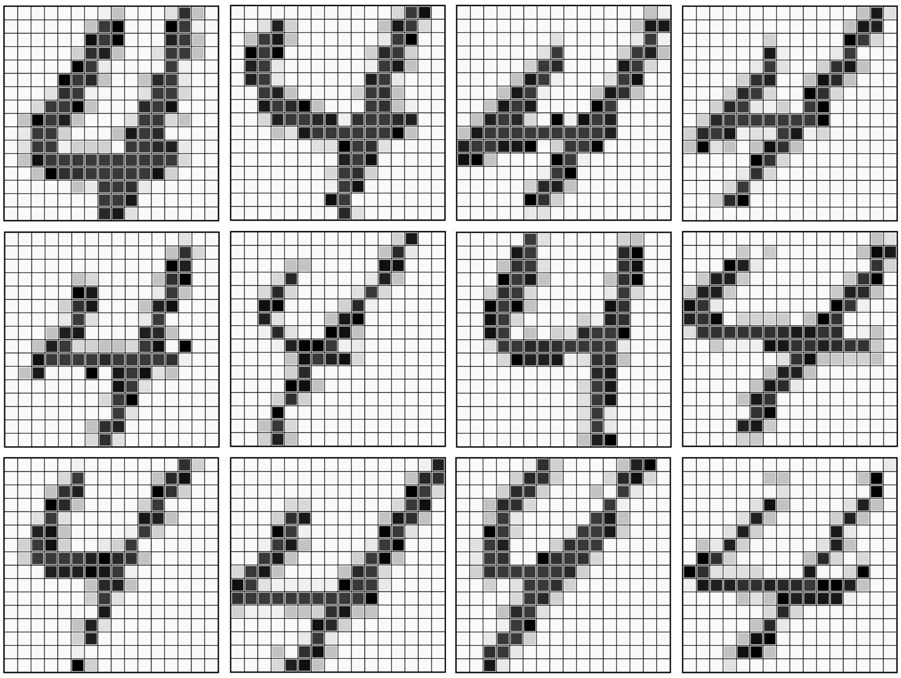

Abbildung 82: *Muster in PLZ.PAT, Klasse 4, Muster Nr. 52 bis 63*

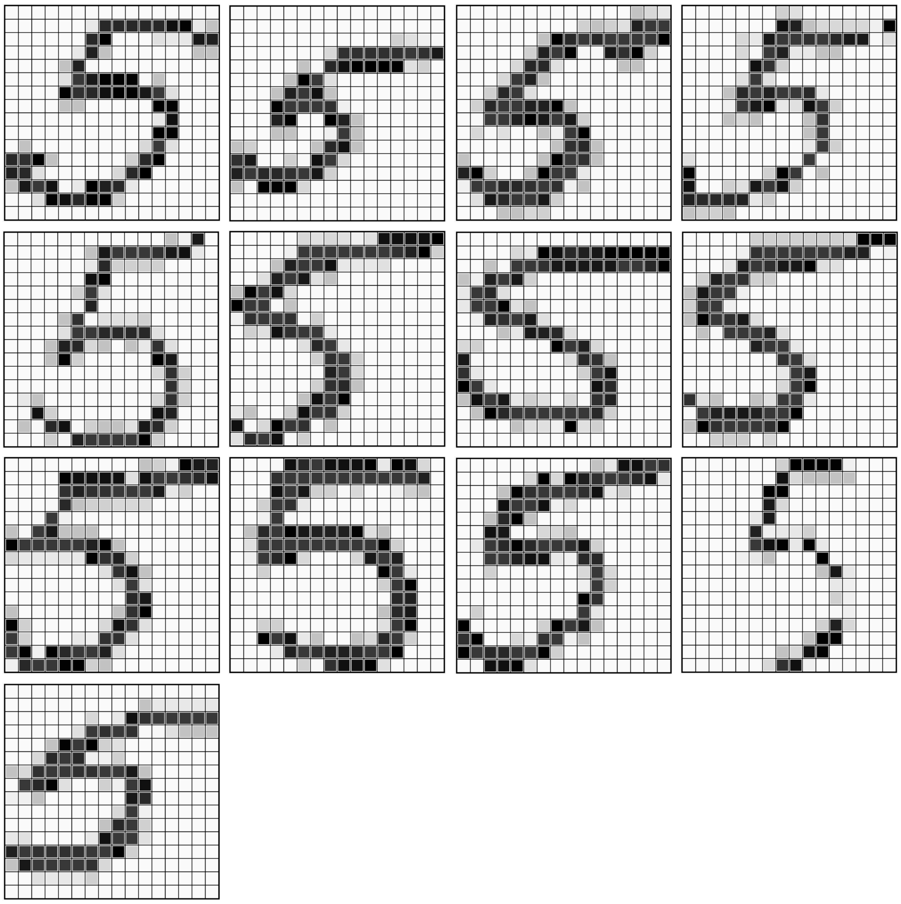

Abbildung 83: *Muster in PLZ.PAT, Klasse 5, Muster Nr. 64 bis 76*

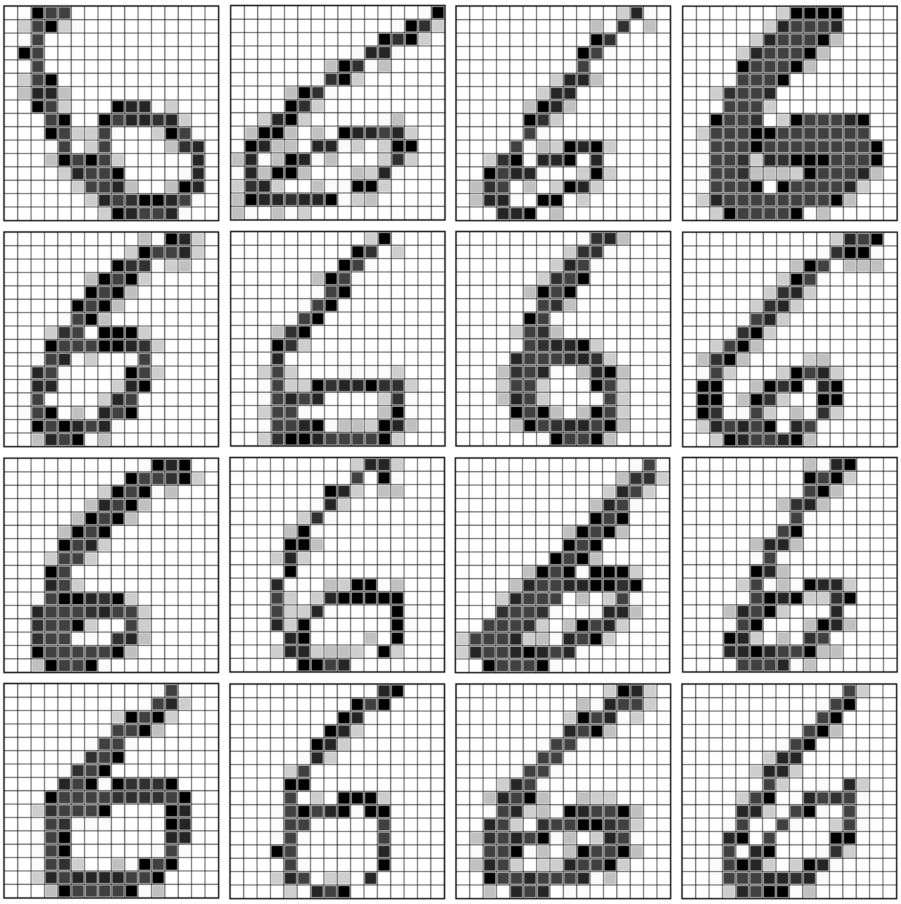

Abbildung 84: *Muster in PLZ.PAT, Klasse 6, Muster Nr. 77 bis 92*

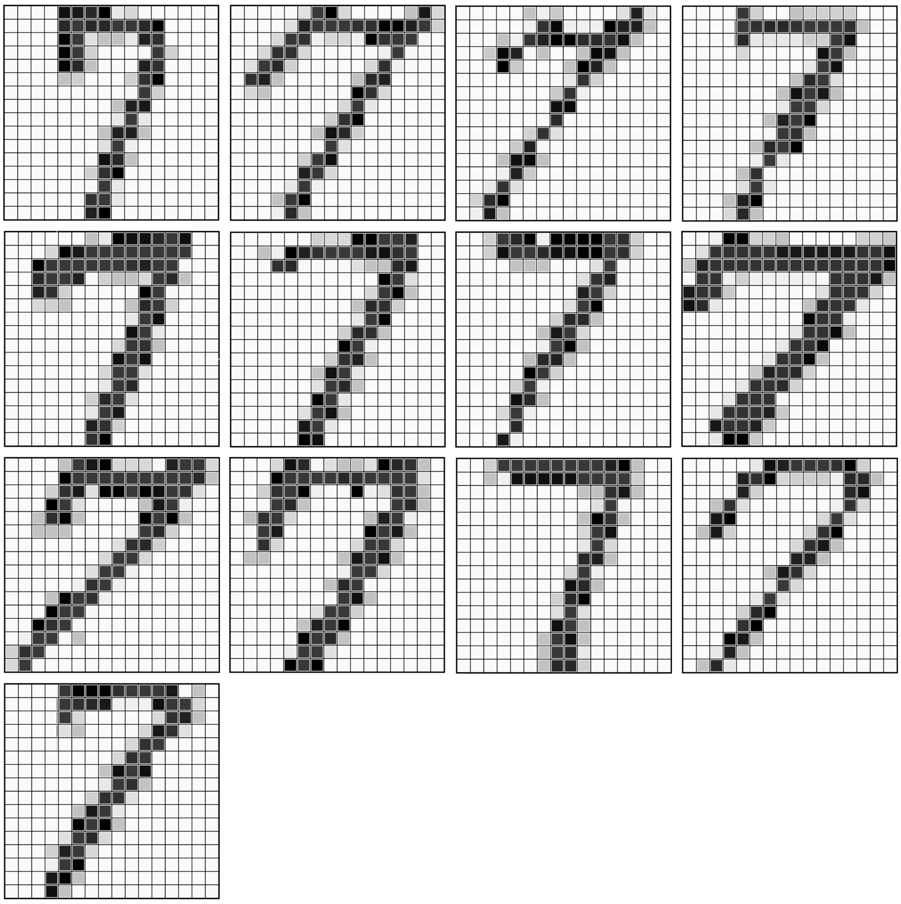

Abbildung 85: *Muster in PLZ.PAT, Klasse 7, Muster Nr. 93 bis 105*

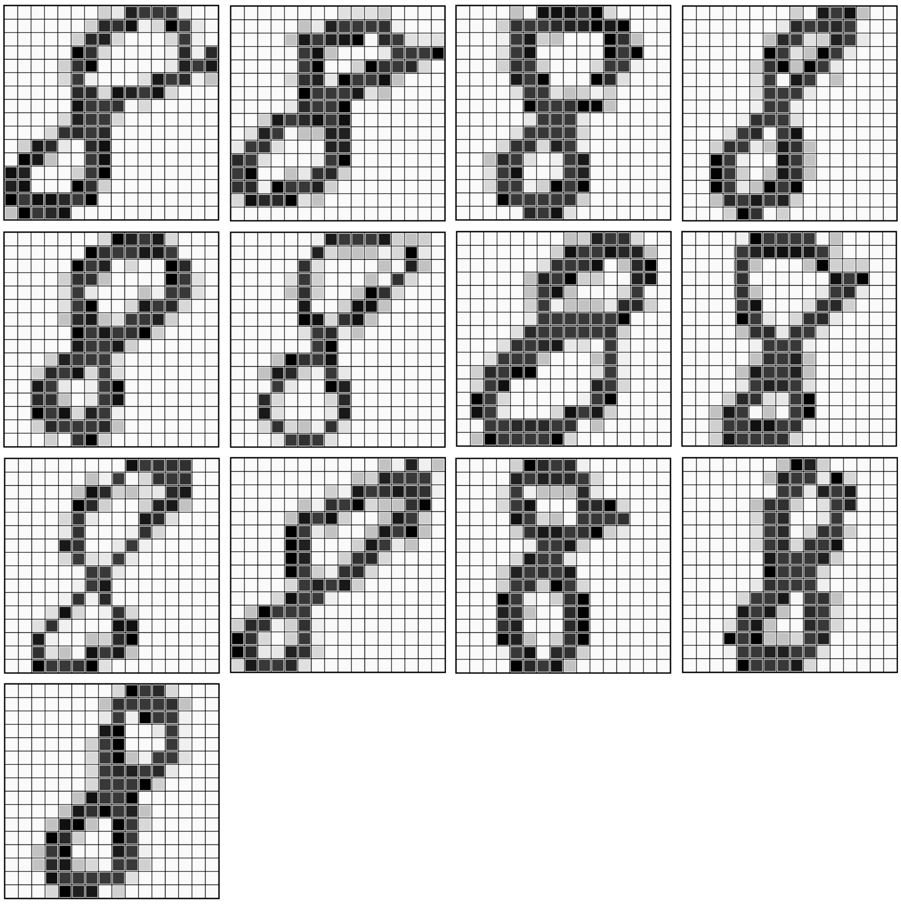

Abbildung 86: *Muster in PLZ.PAT, Klasse 8, Muster Nr. 106 bis 118*

Abbildung 87: *Muster in PLZ.PAT, Klasse 9, Muster Nr. 119 bis 133*

Literatur

[Cho97] S.-B. Cho. "Neural-Network Classifiers for Recognizing Totally Unconstrained Handwritten Numerals". In *IEEE Transactions on Neural Networks*, vol. 8, no. 1. 1997

[ChSa95] B.Chakraborty, Y.Sawada "A Hierarchical Fractal Net for Pattern Classification". In *Proceedings of the IEEE International Conference on Neural Networks*, vol. 1, S. 127-131. 1995

[ChWuYa94] Z.Chi, J.Wu, H.Yan. "Handwritten Numeral Recognition using Self-Organizing Maps and Fuzzy Rules". In *Pattern Recognition*, vol. 28. no. 1, S.59-66. 1995

[FoNe95] W.E.Foor, M.A.Neifeld. "Adaptive, optical, radial basis function neural network for handwritten digit recognition". In *Applied Optics*, vol. 34, no. 32, S. 7545-7555. 1995

[Fri93] B.Fritzke. "Growing Cell Structures - A Self-Organizing Network for Unsupervised and Supervised Learning". International Computer Science Institute Berkley. 1993

[GuSu94] D.Guillevic, C.Y.Suen. "Cursive Script Recognition: A Sentence Level Recognition Scheme". CENPARMI, Concordia University Montreal. 1994

[GuSu95] D.Guillevic, C.Y.Suen. "Cursive Script Recognition applied to the Processing of Bank Cheques". CENPARMI, Concordia University Montreal. 1995

[Holm97] L.Holmström, P.Koistinen, J.Laaksonen, E.Oja. "Neural and Statistical Classifiers—Taxonomy and Two Case Studies". In *IEEE Transactions on Neural Networks*, vol. 8, no. 1. 1997

[Koch81] E. Koch. "Ein kohärent-optisches Filterverfahren für die Zeichenerkennung". Dissertation B, TU Dresden, Fakultät für Naturwissenschaften und Mathematik. 1981

[Koch96] E. Koch. "A Method for Design and Training Feedforward Neural Netzworks". TU Dresden, Technischer Bericht in Vorbereitung. 1996

[Koh78] T.Kohonen. “The Self-Organizing Map”. In *Proceedings of the IEEE*, vol. 9, S. 1464-1480. 1978

[Kov95] Zs.M.Kovacs-V. “A novel Architecture for high Quality hand-printed Character Recognition”. In *Pattern Recognition* Vol.28, No. 11, S. 1685-1692. 1995

[LeCun90] Y.Le Cun, B.Boser, J.S.Denker, D.Henderson, R.E.Howard, W.Hubbard, L.D.Jackel. “Handwritten Digit Recognition with a Back-Propagation Network”. In D.Touretzky, editor, *Neural Information Processing Systems*, vol. 2, 1989

[LoTs97] D.R.Lovell, A.C.Tsoi. “Is the Neocognitron capable of State-of-the-art Digit Recognition?”. University of Queensland. To be published in *IEEE Transactions of Neural Networks*. 1997

[NaKlKr94] D.Nauck, F.Klawonn, R.Kruse. “Neuronale Netze und Fuzzy-Systeme”. Vieweg Verlag. 1994

[Rid96] D.de Ridder. “Shared weights neural networks in image analysis”. M.Sc.Thesis, Delft University of Technology. 1996

[SNNS94] A.Zell und Mitarbeiter. “SNNS, User Manual, Version 3.2”. Universität Stuttgart, Institut für Parallele und Verteilte Höchstleistungsrechner”. 1994

[WeMa94] N.Weymaere, J.-P.Martens. “On the Initialization and Optimization of Multilayer Perceptrons”. In *IEEE Transactions on Neural Networks*, vol. 5, no. 5, S. 738-750. 1994

[WuYa95] J.Wu, H.Yan. “Combined SOM and LVQ Based Classifiers for Handwritten Digit Recognition”. In *Proceedings of the IEEE International Conference on Neural Networks*, vol5, S. 3074-3077. 1995

[Zell94] A. Zell. “Simulation Neuronaler Netze”, Addison Wesley. 1994