Parallele Backpropagation*

Ulrich Middelberg, Frank M. Thiesing, Oliver Vornberger

Universität Osnabrück
Fachbereich Mathematik/Informatik
49069 Osnabrück
frank@informatik.uni-osnabrueck.de

Zusammenfassung. Neuronale Netze eignen sich zur Prognose von Zeitreihen. Dazu wird hier das Backpropagation-Lernverfahren für Multilayer Feedforward Perzeptron Netzwerke in einer modifizierten Form mit Neuronenaufspaltung eingesetzt. Die Parallelisierung dieses zeitaufwendigen Lernverfahrens erfolgt auf zwei Arten: Die Aufteilung des Trainingssets beim Batch-Learning und die Parallelisierung der Matrix-Vektor-Operationen beim On-Line-Training. Die Implementation erfolgt sowohl unter Parix auf Transputernetzwerken als auch unter PVM auf Workstation-Clustern. Ergebnisse über die Güte der modifizierten Backpropagation-Lernregel werden anhand einer zu prognostizierenden Zeitreihe vorgestellt.

1. Einleitung

Neuronale Netze haben sich als geeignet erwiesen, nicht offensichtliche Zusammenhänge komplexer Systeme bis hin zum deterministischen Chaos vergleichsweise gut zu adaptieren und zu verallgemeinern. Prinzipiell können Neuronale Netze beliebige Abbildungen nach einer gewissen Trainingsphase approximieren.

Insbesondere bei der Analyse und Vorhersage von Zeitreihen sind Neuronale Netze eine Alternative zu den klassischen Verfahren. So werden Neuronale Netze bereits heute zur Bedarfsberechnung z.B. für Fernwärme oder elektrischen Strom sowie zur Bestimmung volkswirtschaftlicher Kenngrößen kommerziell eingesetzt [1], [2], [3].

Neuronale Netze lernen durch Präsentation diskreter Funktionswerte, eine den Werten zugrundeliegende Abbildung zu approximieren. Bei der Vorhersage von Zeitreihen lernt das Netz anhand von Werten aus der Vergangenheit, dem sogenannten Trainingsset. Bei Eingabe von n aufeinanderfolgenden Werten soll der Wert zum Zeitpunkt $(n+1)$ als Ergebnis der Abbildung in der Output-Schicht erscheinen. Es wird solange die komplette Trainingsmenge gelernt, bis der Abbildungsfehler unter eine gewisse Schranke gesunken ist. Ein solcher Durchgang wird als *Epoche* bezeichnet. Zur Vorhersage kann das so trainierte Netz eingesetzt werden, indem die letzten n bekannten Werte eingegeben werden und das Netz den Wert für $(n + 1)$ bestimmt.

Ein zur Prognose häufig gewählter Ansatz ist das *Feedforward Multilayer Perceptron (FMP)* Netzwerk mit der *Backpropagation*-Lernregel [4]. Das Trainieren dieses Netzwerktyps erfordert einen hohen Rechenzeitaufwand, der sich mit einer effizienten Parallelisierung des Lernvorgangs signifikant verringern läßt.

*published in: R. Flieger, R. Grebe: *Parallele Datenverarbeitung aktuell: TAT '94*, pp. 419-427, IOS Press 1994. Presented at: Parallelrechner-Anwender-Treffen TAT '94, Aachen, 26.-27. September 1994

2. Modifizierte Backpropagation-Lernregel

Ein FMP-Netzwerk beinhaltet zahlreiche Parameter, deren Einstellungen auf Erfahrungswerten durch Probieren oder auf Intuition beruhen. Ein besonderes Gewicht liegt dabei auf der Anzahl der sogenannten verborgenen *Neuronen* und deren Strukturierung in *Hidden Layer*. Zuviele verborgene Neuronen hemmen die Fähigkeit des Netzes zur gewünschten Abstraktion und erhöhen unnötig die Dauer einer Netzoperation. Hingegen lassen sich mit einer zu geringen Anzahl von verborgenen Neuronen gewisse Fehlerschranken nicht unterschreiten.

Ein FMP-Netzwerk hat die in der Abbildung 1 dargestellte Struktur.

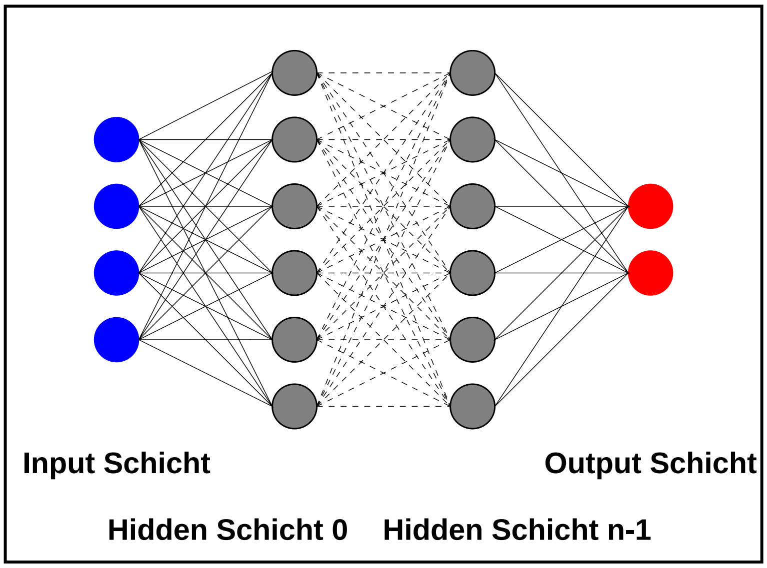

Abbildung 1: Feedforward Multilayer Perzeptron

Jeder in der Eingabeschicht angelegte Vektor wird durch das FMP auf einen Ausgabevektor abgebildet. Die Größen der In- und Output-Schichten sind durch die Modellierung des Problems vorgegeben. Die Aufgabe des Anwenders ist es, die Anzahl der verborgenen Neuronen und deren Strukturierung in Hidden Layer festzulegen.

Die Backpropagation-Lernregel dient dazu, ein fehlerhaftes Abbildungsverhalten des Netzes durch eine Modifikation der Netzgewichte zu korrigieren. In einer sogenannten *Forward*-Phase wird der Fehler als Differenz zwischen Soll- und Istwert in der Output-Schicht bestimmt, in der *Backward*-Phase werden ausgehend von diesen Fehlern die Kantengewichte modifiziert. Die Korrektur der Kantengewichte beruht auf der Minimierung der Fehlerfunktion nach dem Gradientenabstiegsverfahren.

Der *modifizierte* Backpropagation-Algorithmus ist in der Lage, durch eine monotone und fehlerlokale Netzerweiterung die Qualität des trainierten Netzes zu verbessern, ohne seine Fähigkeit zur Abstraktion einzuschränken [5]. Das Training beginnt mit wenigen verborgenen Neuronen. Während der Lernphase werden schlecht trainierte Neuronen in regelmäßigen Abständen einzeln aufgespalten, bis deren Maximalanzahl erreicht ist. Durch dieses Verfahren wird nach kürzerer Zeit ein Fehlerminimum erzielt, welches der gewöhnliche Backpropagation-Algorithmus kaum erreicht.

Die schematische Vorgehensweise und die Wahl der zusätzlichen Kantengewichte sind in Abbildung 2 dargestellt.

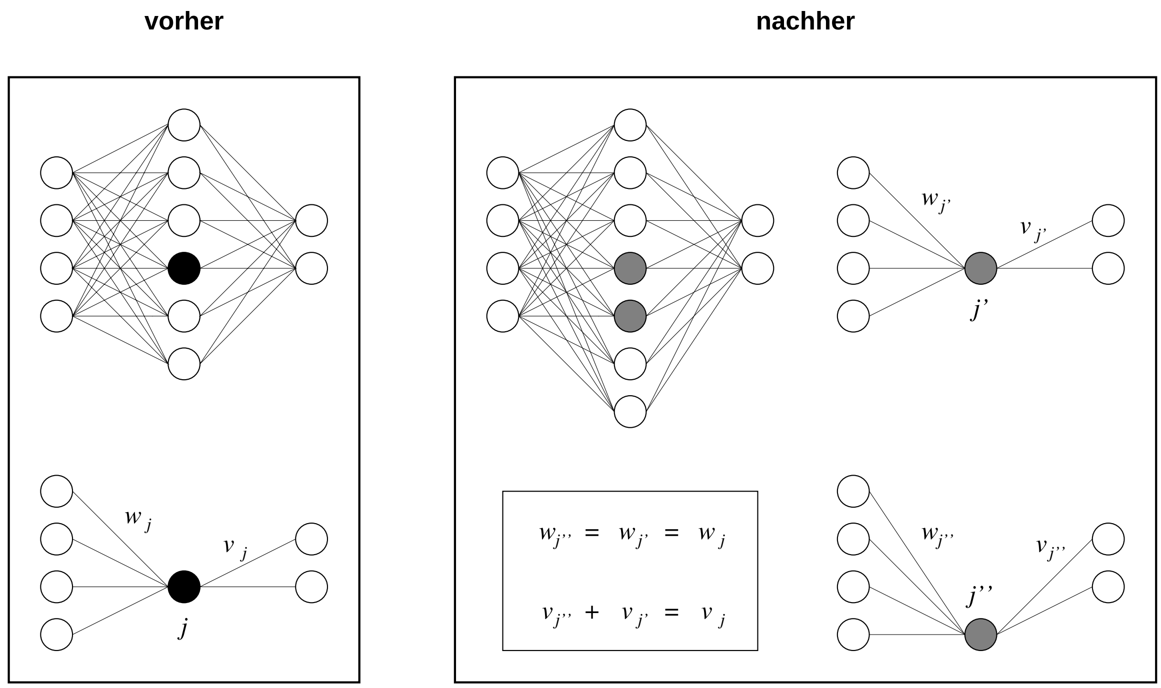

Abbildung 2: Modifizierte Backpropagation-Lernregel mit Neuronenaufspaltung

3. Parallelisierung

Die Backpropagation-Lernregel kann grundsätzlich auf zwei verschiedene Arten durchgeführt werden. Beim *Batch*-Learning wird die Korrektur der Gewichte erst nach der Bestimmung aller Gewichtsänderungen durchgeführt. Als Alternative bietet sich das *On-Line*-Training, bei dem die Gewichtskorrekturen unmittelbar während jeder Backward-Phase nach der Berechnung der Gewichtsänderungen vorgenommen werden. Neben den algorithmischen Unterschieden differieren beide Verfahren auch in ihrer Konvergenzgeschwindigkeit auf unterschiedlichen Trainingsmengen.

3.1. Batch-Learning

Die Parallelisierung des Batch-Learnings besteht darin, die Trainingsmenge aufzuteilen. Die Gewichtsänderungen werden lokal pro Trainingsset berechnet, anschließend werden die Gewichtsdifferenzen aufsummiert und zu den tatsächlichen Gewichten hinzuaddiert. Der Vorgang des Updatens findet einmal nach jeder Epoche statt (vgl. Abb. 3).

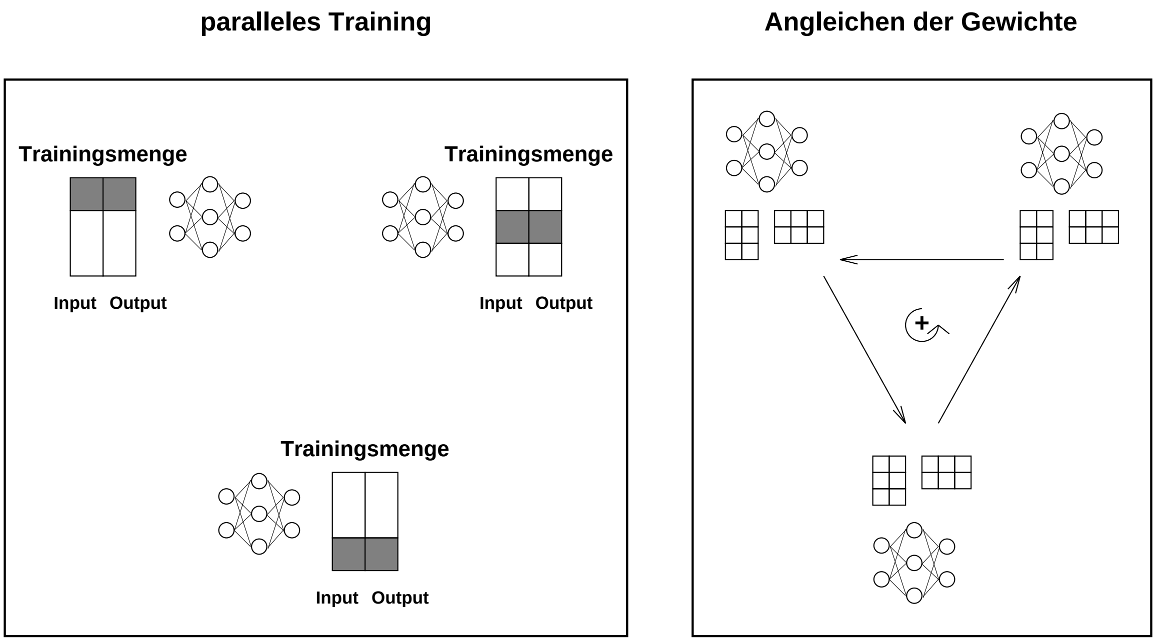

Abbildung 3: Paralleles Batch-Learning

3.2. On-Line-Training

Beim On-Line-Training wird das Neuronale Netz mit der gesamten Trainingsmenge trainiert. Die Parallelisierung verfolgt hier den Ansatz, die aufwendigen Matrix-Vektor-Operationen während der Forward- und Backward-Phase aufzuteilen. Dazu wird die Zuständigkeit für die Neuronen jeder Schicht unter den Prozessoren verteilt.

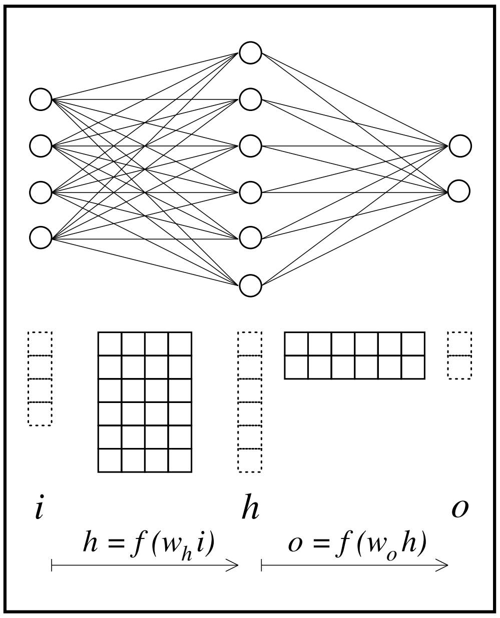

Abbildung 4: Matrix-Vektor-Operationen in der Forward-Phase

Jeder Prozessor bestimmt für die ihm zugeordneten Neuronen die neuen Aktivierungen anhand der Aktivierungen der vorherigen Schicht und der lokal verteilten Gewichtsvektoren. Das Ergebnis dieser Operation wird durch eine *sigmoide* Transferfunktion f in eine zulässige Aktivierung überführt (s. Abb. 4). Bevor die Aktivierungen der folgenden Schichten bestimmt werden können, müssen die neu berechneten Aktivierungen der Neuronen einer Schicht global verteilt werden (s. Abb. 5).

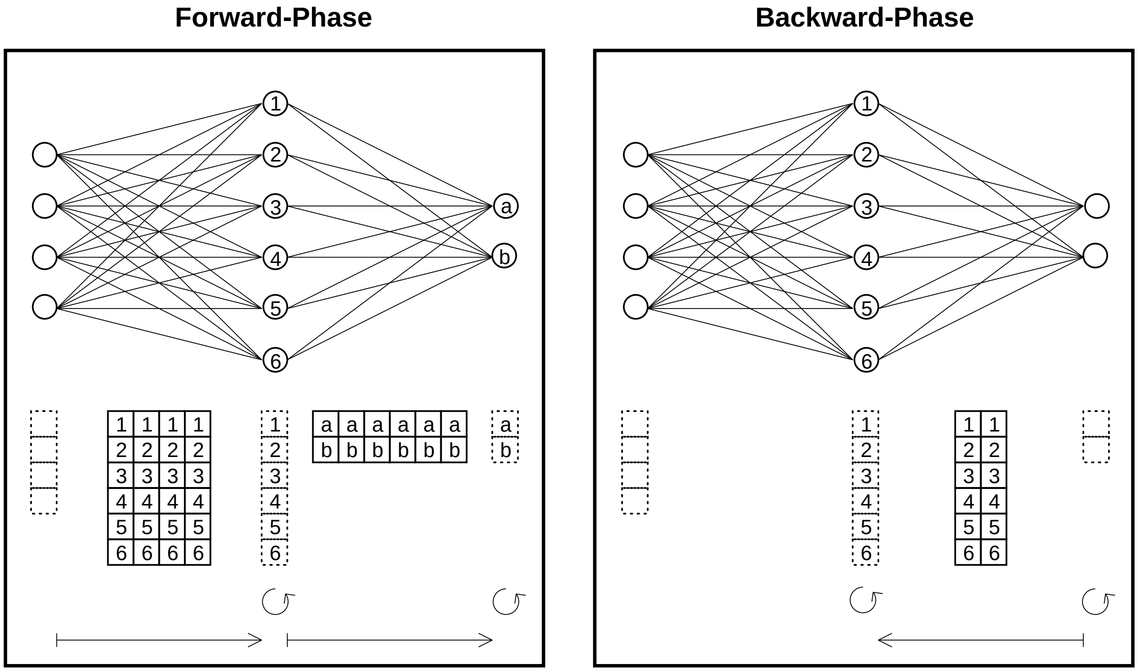

Abbildung 5: Parallele Forward- und Backward-Phase

In der Backward-Phase werden ausgehend von dem Fehlervektor in der Output-Schicht die Fehler rückwärts propagiert. Zur Verringerung des Kommunikationsaufwands werden neben den rezeptiven auch die projektiven Gewichte eines Neurons auf dem jeweiligen Prozessor gespeichert [6]. Ausgehend von der Output-Schicht wird für jede Hidden-Schicht der jeweils dort auftretende Fehler bestimmt. Die Matrix-Vektor-Operationen finden wie in der Forward-Phase parallel statt (s. Abb. 5).

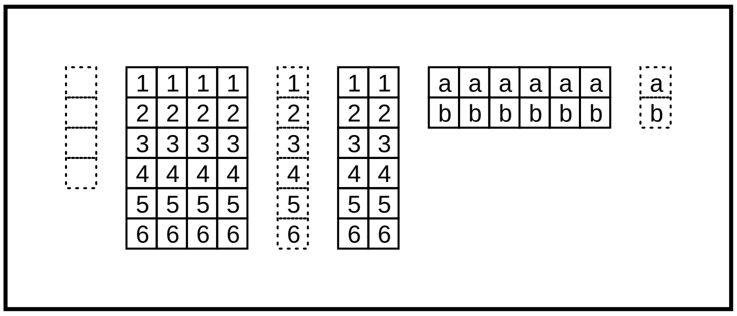

Abbildung 6: Aktualisierung der Gewichtsmatrizen

Die Gewichtsmatrizen werden parallel aktualisiert, gemäß der Zuordnung der Neuronen auf die Prozessoren. Diese sind in der Abbildung 6 mit a und b bzw. 1 bis 6 bezeichnet.

Ein Problem bei der Verfolgung dieser feinkörnigen Parallelisierungsidee ist die Behandlung des Neuronen-Splittings. Durch das Aufspalten der Neuronen und das daraus resultierende Wachsen der Hidden-Schichten ist es notwendig, dynamisch die neuen Neuronen mit den Zeilen bzw. Spalten der Gewichtsmatrix zu verteilen.

4. Implementation

Die Implementation der parallelen Algorithmen erfolgt sowohl auf einem Parallelrechner als auch auf einem Workstation-Cluster. Zur Verfügung stehen ein Transputernetzwerk mit 64 T800 mit der Laufzeitumgebung Parix und ein Cluster von SUN SPARCs mit PVM. Als interessante Vergleichsmöglichkeit bietet sich auch PVM auf Parix für die Transputer an. Als Programmiersprache steht in allen Fällen ANSI-C zur Verfügung.

5. Ergebnisse

Um die Güte unseres Backpropagation-Algorithmus mit der modifizierten Lernregel zu testen, wird als Zeitreihe die chaotische Entwicklung der VERHULST-Gleichung [7] gewählt mit $k = 2,8$.

$$\boxed{p_{n+1} = p_n \cdot (1 + k \cdot (1 - p_n))} \qquad \text{(VERHULST-Gleichung)}$$

Der Verlauf der Folgenwerte ist in den Abbildungen 7 bis 9 zusammen mit den Abbildungsergebnissen unterschiedlicher Netzkonfigurationen und den benötigten Trainingszeiten dargestellt.

5.1. Meßwerte

In den folgenden drei Abbildungen werden neben einem Ausschnitt der Trainingsmenge (links des Trennstrichs) auch einige nicht trainierte, d.h. prognostizierte Werte dargestellt.

Um die Approximationsfähigkeit des FMP-Netzwerkes zu testen, werden zwei Ansätze gewählt. Zum einen wird aus drei aufeinanderfolgenden Gliedern der nächste Wert bestimmt. Dabei wird ein kleines Netz mit nur einer Hidden-Schicht mit sechs Neuronen trainiert (s. Abb. 7).

Der zweite Ansatz verfolgt das Ziel, mehr Informationen über die Vergangenheit einfließen zu lassen. Dabei werden die Werte von 30 Input-Neuronen durch zwei Hidden-Schichten mit zehn bzw. 15 Neuronen propagiert (s. Abb. 8).

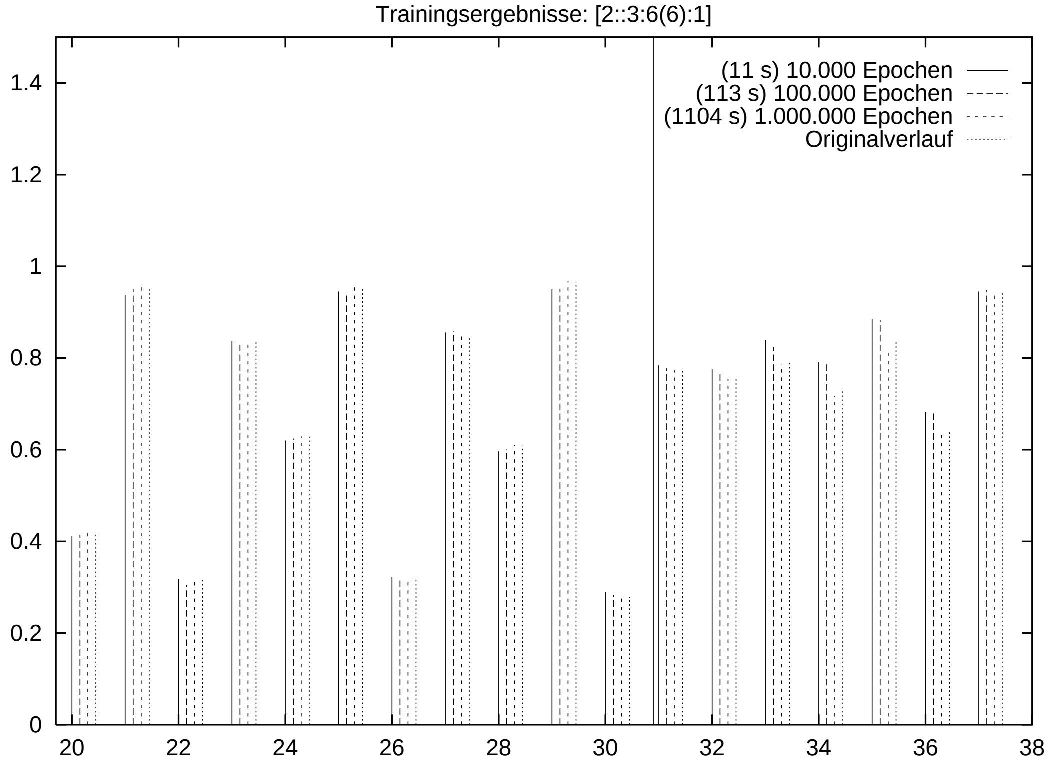

Abbildung 7: Trainingsergebnisse eines Netzes mit sechs verborgenen Neuronen

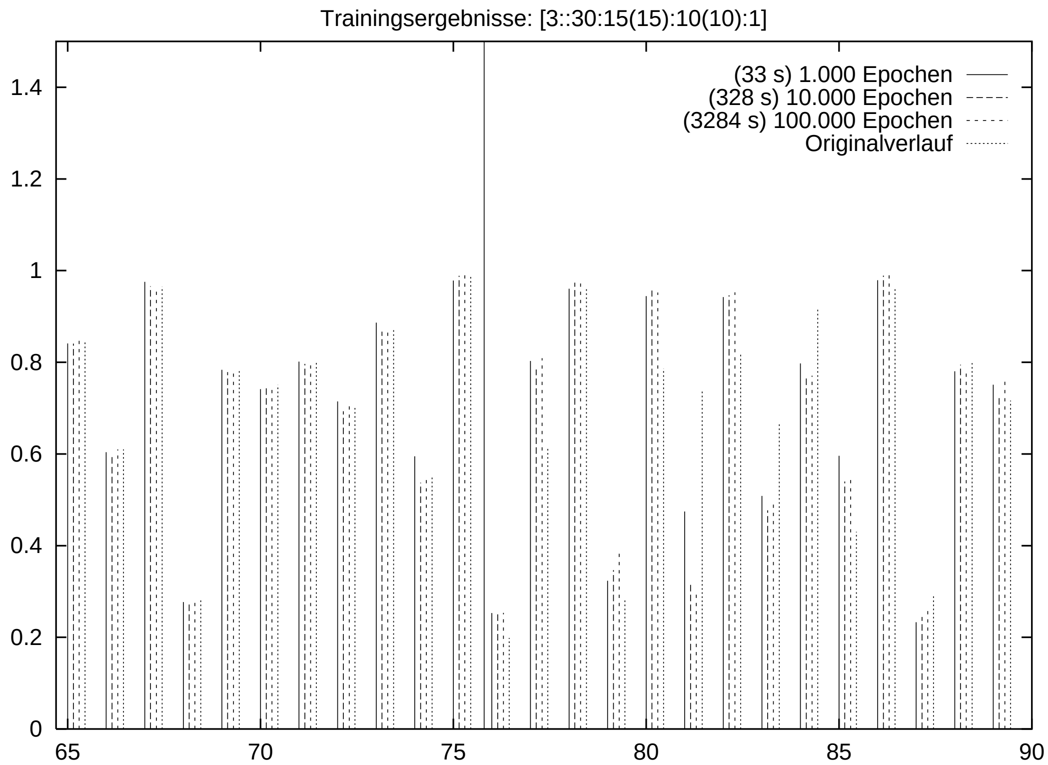

Abbildung 8: Trainingsergebnisse eines Netzes mit 15 und 10 verborgenen Neuronen

Der Vergleich der Abbildungen 7 und 8 zeigt deutlich eine bekannte Problematik: Die Adaptionsfähigkeit des großen Netzes ist nicht höher als die des kleineren; es lernt lediglich die Trainingsmenge auswendig.

Als neue Variante im Vergleich zur herkömmlichen Methode wird die Zahl der Neuronen in der Hidden-Schicht gemäß der modifizierten Lernregel aufgespalten. Das Netz beginnt mit sechs Neuronen in der Hidden-Schicht und wächst bis auf 24 Neuronen. Dieses wird in Abbildung 9 verglichen mit zwei Netzen mit konstant sechs bzw. 24 Neuronen in der Hidden-Schicht. Das Trainingsset und die Abfrage entsprechen dem ersten Ansatz.

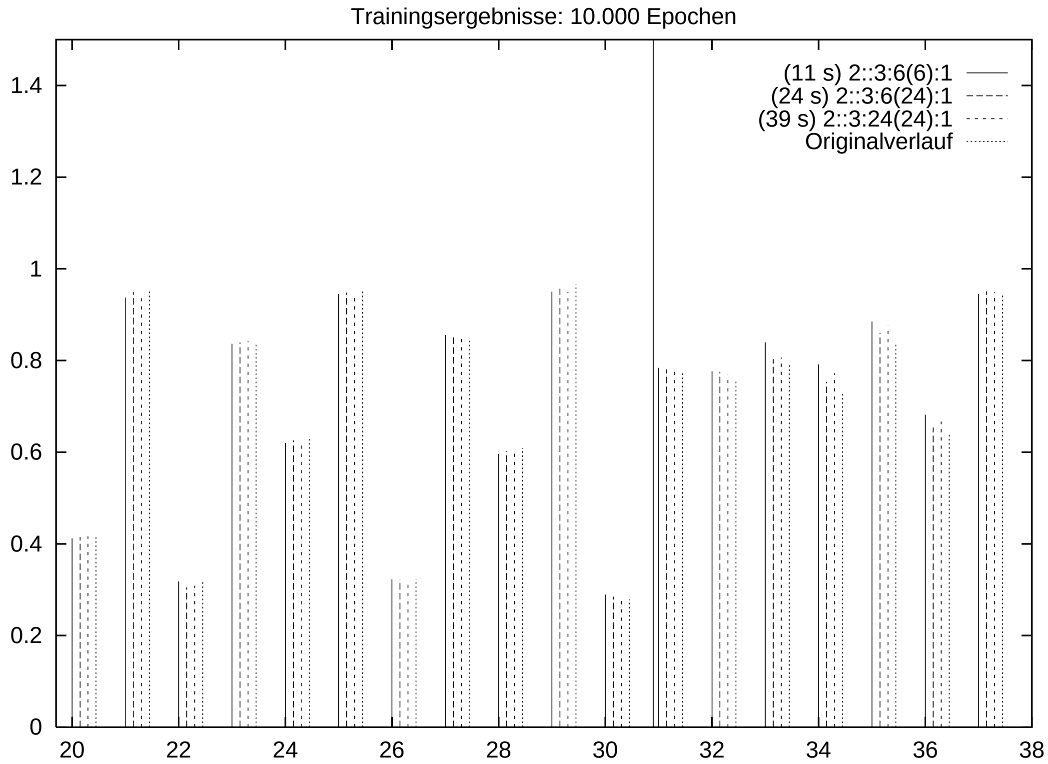

Abbildung 9: Vergleich der Trainingsergebnisse mit und ohne Neuronenaufspaltung

5.2. Auswertung

Die Meßergebnisse in Abbildung 9 zeigen, daß das kleinere Netz Probleme hat, nicht nur den Fehler in der Trainingsmenge zu minimieren, sondern insbesondere auch nicht in der Lage ist, unbekannte Werte zu prognostizieren. Das größere Netz minimiert zwar den Fehler auf der Trainingsmenge sehr gut, versagt aber auf den Testdaten, da es lediglich auswendig gelernt hat. Das Netz, welches unter Neuronenaufspaltung mit der modifizierten Lernregel trainiert wurde, weist sowohl einen geringeren Fehler auf den Trainings- als auch auf den Testdaten auf und lernt aufgrund der lange Zeit kleinen Neuronenzahl auch schneller.

6. Zusammenfassung

Für die Parallelisierung des zeitaufwendigen Trainings Neuronaler Netze mit der Backpropagation-Lernregel bieten sich zwei Verfahren an: Die Aufteilung des Trainingssets beim parallelen Batch-Learning und die Parallelisierung der notwendigen Matrix-Vektor-Operationen beim On-Line-Training. Beide Verfahren lassen sich in der Güte und der Geschwindigkeit der Konvergenz durch die modifizierte Lernregel noch steigern. Dabei wächst die Zahl der Neuronen in den Hidden-Schichten durch Aufspaltung schlecht trainierter Neuronen.

Die Kombination der vorgestellten Verfahren führt dazu, gut adaptierende Neuronale Netze schnell zu trainieren.

Literatur

[1] U. Ahle. *Künstliche Neuronale Netze für die Prognose des Fernwärmebedarfs.* Vorläufiger Tagungsband der 4. Dortmunder Fuzzy-Tage, pp. 263-273, 1994.

[2] H. Schreiber, S. Heine. *Einsatz von Neuro-Fuzzy-Technologien für die Prognose des Elektroenergieverbrauchs an „besonderen" Tagen.* Vorläufiger Tagungsband der 4. Dortmunder Fuzzy-Tage, pp. 274-281, 1994.

[3] B. Widrow, D.E. Rumelhart, M.A. Lehr. *Neural Networks: Applications in Industry, Business and Science.* CACM March 1994, Vol. 37, No. 3, pp. 93-105, 1994.

[4] D.E. Rumelhart, G.E. Hinton, R.J. Williams. *Learning internal representations by error propagation.* In D.E. Rumelhart and J.L. McClelland (Eds.), Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1, pp. 318-362, MIT 1987.

[5] I. Glöckner. *Monotonic Incrementation of Backpropagation Modules and Network Architectures.* Studienarbeit, Universität Osnabrück, Computerlinguistik und künstliche Intelligenz, 1994.

[6] H. Yoon, J.H. Nang, S.R. Maeng. *A distributed backpropagation algorithm of neural networks on distributed-memory multiprocessors.* Proceedings of the 3rd symposium on the Frontiers of Massively Parallel Computation, pp. 358-363, IEEE 1990.

[7] H.-O. Peitgen, H. Jürgens, D. Saupe. *Bausteine des Chaos - Fraktale.* Klett-Cotta/Springer-Verlag, 1992.