

Funktionsapproximation mit Hilfe von künstlichen neuronalen Feedforward-Netzen

Martin Bernreuther

Informationsverarbeitung im Konstruktiven Ingenieurbau

Universität Stuttgart

Kurzfassung: Der vorliegende Beitrag beschäftigt sich mit den Grundlagen neuronaler Netze im Bereich der Funktionsapproximation. Hierbei wird speziell auf die Klasse der „Feedforward“-Netze eingegangen. Ein Schwerpunkt stellen Lernmethoden zur Bestimmung der Parameter dar. Hier stehen Gradientenverfahren, insbesondere die „Backpropagation“-Methode im Vordergrund. Eine Approximation einer statischen Berechnung durch ein neuronales Netz wird hinsichtlich der Lernparameter auf die erzielbaren Konvergenz hin untersucht und die Approximationsgüte bestimmt.

Das Vorbild für künstliche neuronaler Netze stammt aus dem Bereich der Biologie. Neuronale Netze bilden die Informationsverarbeitung bei Menschen und Tieren durch Simulation der Nervenzellen und ihres Zusammenwirkens im Gehirn nach. Das Gehirn empfängt über die Sinnesorgane Reize, deren Verarbeitung Reaktionen auslöst. Analog hierzu erhält eine Funktion Eingabewerte nach deren Verarbeitung sie einen Rückgabewert liefert. Neuronale Netze sind somit Automaten zur Berechnung von Funktionen.

1 Aufbau neuronaler Netzwerke

Ein neuronales Netzwerk bildet über eine Vielzahl einfacher, miteinander verbundener Verarbeitungseinheiten ein funktionales Gesamtsystem. Die Verarbeitungs- oder Berechnungseinheiten, im folgenden Neuronen genannt, erhalten über gerichtete Verbindungen einen Reiz, bzw. eine Eingabe. Diese Eingabe wird verarbeitet und erzeugt eine Ausgabe, die über Verbindungen weitergeleitet wird. Die Vorgänge dieses Prozesses werden im Gehirn massiv parallel ausgeführt und so die Leistungsfähigkeit des Gesamtsystems gesteigert. Die Anpassung der Funktionalität erfolgt über die Verbindungsstruktur, indem die Weiterleitung an andere Neuronen unterschiedlich stark erfolgt

und variiert werden kann. Ein neuronales Netz basiert somit auf einem gerichteten, gewichteten Graphen, bestehend aus Neuronen und Verbindungskanten.

1.1 Neuronen und Verbindungen

In Abbildung 1 ist ein Neuron schematisch dargestellt. Die Funktionsweise wird in drei Ebenen, bzw. Teilfunktionen aufgeteilt:

Die *Eingabe- bzw. Propagierungsfunktion* berechnet den Eingabewert. Die einzelnen Erregungen oder Eingaben werden hier aufsummiert:

$$net_j = f_{inp}(o_{con,ij}) = \sum_i o_{con,ij} \quad \text{mit} \quad o_{con,ij} = f_{con}(w_{ij}, o_i) = w_{ij} \cdot o_i$$

Der Verbindungskante wird hier eine eigene Funktionalität zugewiesen. Sie verstärkt oder schwächt das eingehende Ausgabesignal eines Neurons ab. Dies erfolgt multiplikativ mit der Wichtung. Üblicherweise wird dies in der Eingabefunktion mitberücksichtigt. Im nächsten Schritt berechnet man aus dem Eingabewert den Aktivierungszustand des Neurons mit Hilfe der *Aktivierungsfunktion*. Im Allgemeinen kann auch

der vorhergehende Zustand des Neurons mit einbezogen werden. Ansonsten ist kein Zeitparameter t notwendig. Aus der Aktivierung ergibt sich über die *Ausgabefunktion* die Ausgabe der Zelle. Für die Ausgabefunktion verwendet man üblicherweise die Identitätsfunktion, so daß dieser Schritt vernachlässigt werden kann. Bei den Eingabeneuronen sind keine eingehenden Verbindungskanten vorhanden. Ihnen werden die Eingabewerte bzw. Funktionsparameter übergeben. Dementsprechend besitzen Ausgabeneuronen keine ausgehenden Verbindungen und liefern das Ergebnis.

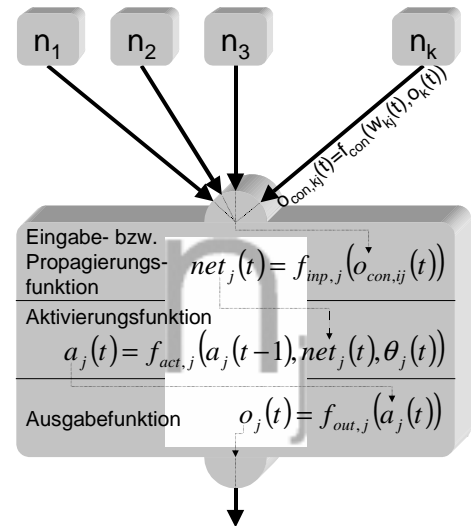


Abbildung 1: Neuron

1.2 Aktivierungsfunktionen

Für die Eingabeneuronen wird üblicherweise eine lineare Aktivierungsfunktion, die Identität, verwendet. Auch für Ausgabeneuronen kann, je nach Problem, eine solche Aktivierungsfunktion sinnvoll sein, ansonsten können solche Neuronen durch modifizierte Verbindungsgewichte ersetzt werden. Taylor-Reihenentwicklungen können mit Monomen und Fourierreihen mit Sinusfunktionen als Aktivierungsfunktion implementiert werden. Beim einfachen Perzeptron, einem speziellen Neuron, wird als Aktivierungs-

funktion die binäre Schwellwertfunktion verwendet: $f_{act}(net, \theta) = \begin{cases} 1 & \text{falls } net \geq \theta \\ 0 & \text{sonst} \end{cases}$. Pro-

blematisch ist die Unstetigkeitsstelle der Schwellwertfunktion. Über die Aktivierungsfunktion wird die Klasse der approximierbaren Funktionen, bzw. der Funktionalraum

vorgegeben. Unstetige Funktionen sind auch nicht differenzierbar, was für die Lernalgorithmen von Bedeutung ist.

Am häufigsten werden sigmoide Funktionen für die Aktivierung eingesetzt. Bei diesem Funktionstyp wird die Schwellwertfunktion mit stetigen, differenzierbaren Funktionen angenähert. Eine solche S-förmige Funktion ist der Tangens hyperbolicus oder die hier verwendete *logistische Aktivierungsfunktion*:

$$f_{\log}(x) = \frac{1}{1 - e^{-\frac{x}{T}}} \quad \text{mit}$$

$\frac{d f_{\log}(x)}{dx} = \frac{1}{T} \cdot f_{\log}(x) \cdot (1 - f_{\log}(x))$. Der Parameter T beeinflusst die Steilheit des Anstiegs. Für kleine T nähert man sich der Schwellwertfunktion. Die angegebenen Formeln, sowie die Darstellung in Abbildung 2 gehen von einem Schwellwert $\theta=0$ aus. Dies ist ausreichend, da der Schwellwert ausgelagert werden kann.

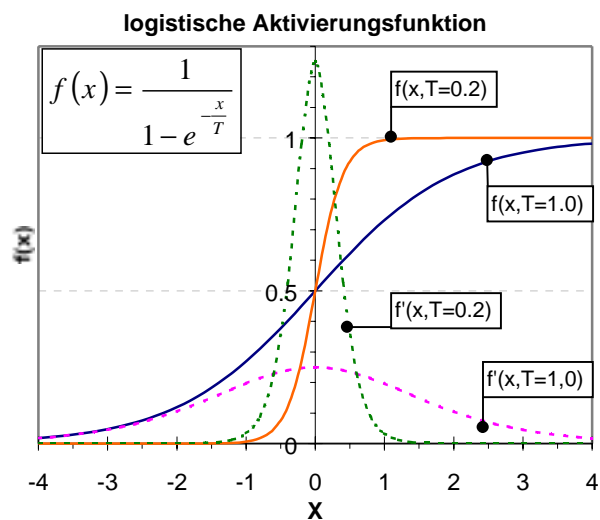


Abbildung 2: logistische Funktion

1.3 Auslagern des Schwellwerts

Die in 1.2 dargestellten Aktivierungsfunktionen können in der folgenden Weise transformiert werden: $f(x, \theta) \rightarrow f(x - \theta, 0)$. Durch die Subtraktion des Schwellwertes von dem Eingabewert x , wird die Funktion in x -Richtung verschoben. Ein sog. „On“-Neuron, das eine konstante Ausgabe von 1 liefert, wird mit jedem einzelnen Neuron, dessen Schwellwert geregelt werden soll, verbunden und den Gewichten jeweils der Wert $-\theta$ zugewiesen. Die Unbekannten des neuronalen Netzes sind somit ausschließlich Verbindungsgewichte.

1.4 Netztopologie

Die Neuronen werden ebenenweise angeordnet. Die unterste Schicht wird als Eingabeschicht und die oberste Schicht als Ausgabeschicht bezeichnet. Zwischen diesen Schichten können eine beliebige Anzahl versteckter Schichten angeordnet sein. Eine grobe Klassifikation unterscheidet Netze mit und ohne Rückkopplung (feedback).

Es sollen hier rückkopplungsfreie „Feedforward“-Netze (FF-Netze) besprochen werden, bei denen Neuronen einer Ebene nur mit Neuronen einer höheren Ebene verbunden sind. Wird hierbei eine Ebene oder mehrere Ebenen übersprungen, erhält man eine „shortcut connection“, und somit ein FF-Netz 2. Ordnung, da bei FF-Netzen 1. Ordnung nur Neuronen zweier aufeinanderfolgenden Schichten verbunden sein dürfen. Bei ei-

nem vollständig vernetzten FF-Netz 1. Ordnung wird jedes Neuron mit allen Neuronen der nächst höheren Schicht verbunden. Die Reihenfolge der Aktivierung kann durch eine topologische Sortierung des Graphen festgelegt werden. Die Neuronen einer Schicht sind hier synchron aktivierbar. Es sei a_i die Anzahl der Neuronen einer Schicht i eines vollständig vernetzten FF-Netzes mit n Schichten, wobei die Eingabeschicht mit $i=0$ und die Ausgabeschicht mit $i=n$ gekennzeichnet wird. Neben $\sum_{i=1}^n a_i$ Verbindungen zum „On“-Neuron existieren $\sum_{i=1}^n a_{i-1} \cdot a_i$ Verbindungen zwischen den Schichten. Es ergeben sich $\sum_{i=1}^n (a_{i-1} + 1) \cdot a_i$ Verbindungsgewichte.

1.5 Partielle Ableitungen

Für die durch das neuronale Netz approximierte Funktion existieren partielle Ableitungen nach den Eingabevariablen, falls die Aktivierungsfunktionen der Neuronen differenzierbar ist. Dies ist für die sigmoiden Funktionen der Fall. Zur Berechnung muß bei der Funktionskomposition, dem „Feedforward“, zusätzlich die Ableitung der Aktivierungsfunktion für die Eingabe des Neurons bestimmt und abgespeichert werden. Die Backpropagation durchläuft das Netz rückwärts und multipliziert in den Neuronen die Eingabe mit dem Ableitungswert. Ansonsten wird analog zum „Feedforward“ verfahren. Die Neuronen erhalten bei der „Backpropagation“ eine Eingabe von $\frac{\partial g(\cdot)}{\partial o_i}$ und eine Ausgabe von $\frac{\partial g(\cdot)}{\partial net_i} = \frac{\partial g(\cdot)}{\partial o_i} \cdot \frac{\partial o_i}{\partial net_i}$. Die Eingabeneuronen liefern die Ableitung nach den Eingabeparametern. Die Bildung der partiellen Ableitungen ist für eine serielle Schaltung von zwei Neuronen in Abbildung 4, sowie für eine parallele Schaltung in Abbildung 3 ge-

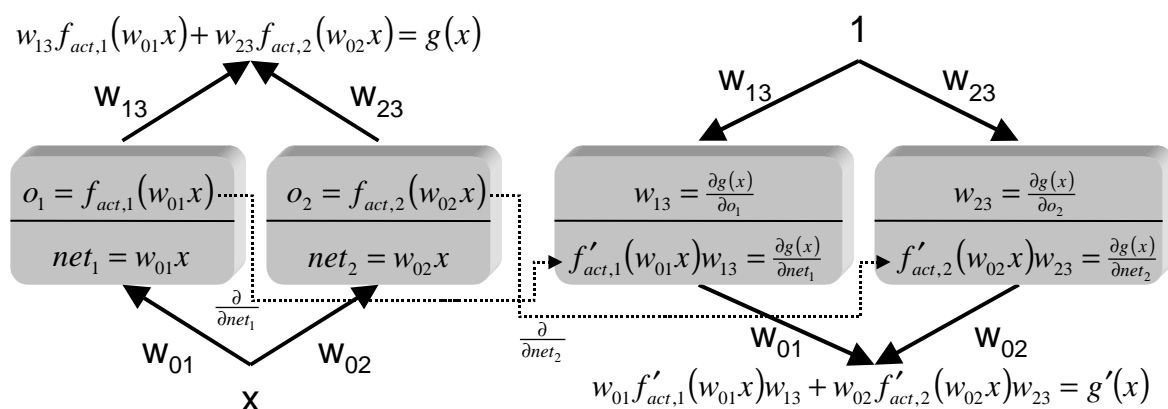


Abbildung 3: Parallele Schaltung von zwei Neuronen

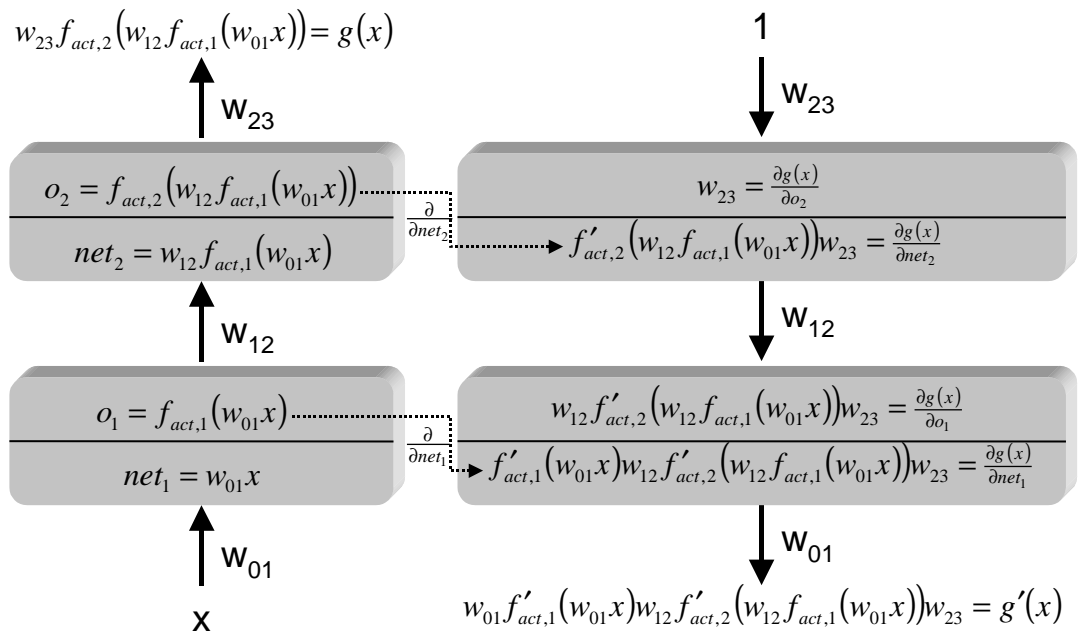


Abbildung 4: Serielle Schaltung von zwei Neuronen

zeigt. Auch die Ableitungen nach den Verbindungsgewichten w_i können analog bestimmt werden. Hierzu benötigt man die berechnete Aktivierung aus dem „Feedforward“-Schritt, denn es gilt

$$\frac{\partial g(\cdot)}{\partial w_{ij}} = \frac{\partial net_j}{\partial w_{ij}} \cdot \frac{\partial g(\cdot)}{\partial net_j} = o_i \cdot \frac{\partial g(\cdot)}{\partial net_j}.$$

2 Lernverfahren für „Feedforward“-Netze

Lernverfahren dienen zur Bestimmung der unbekanntenen Verbindungsgewichte. Beim überwachten Lernen benötigt man Eingabevektoren mit bekannten Ausgaben \vec{t}_p , die als Muster bezeichnet werden. Die vom neuronalen Netz berechnete Ausgabe \vec{o}_p eines Eingabevektors wird mit der vorgegebenen, erwünschten Soll-Ausgabe verglichen und ein Fehler bestimmt. Unter Verwendung der euklidische Vektornorm ergibt sich für eine

$$n\text{-dimensionale Ausgabe folgender Fehler: } E_p = \frac{1}{2} \cdot \|\vec{o}_p - \vec{t}_p\|^2 = \frac{1}{2} \cdot \sum_{q=1}^n (o_{q,p} - t_{q,p})^2$$

Das Vorgehen entspricht dem der Methode der kleinsten Quadrate. Die Bestimmung der Verbindungsgewichte mit dem Ziel einen minimalen Fehler zu erhalten, stellt ein Optimierungsproblem dar. Da das Netz für eine vorgegebene Menge von Mustern, eine sog. Epoche, die richtigen Ergebnisse liefern soll, werden die Fehler zum Gesamtfehler aufsummiert, d. h. $E = \sum_p E_p$. Das entspricht einer Multikriterienoptimierung. Durch die

geeignete Wahl der Neuronen-Aktivierungsfunktion ist die von den Verbindungsge-

wichtigen abhängige Fehlerfunktion glatt, so daß Gradientenverfahren angewendet werden können. Für eine Epoche mit m Muster erhält man:

$$\begin{aligned} \frac{\partial E(w_{ij})}{\partial w_{ij}} &= \frac{\partial}{\partial w_{ij}} \frac{1}{2} \cdot \sum_{p=1}^m \sum_{q=1}^n (o_{q,p} - t_{q,p})^2 \Big| = \frac{1}{2} \cdot \sum_{p=1}^m \sum_{q=1}^n \frac{\partial}{\partial w_{ij}} (o_{q,p} - t_{q,p})^2 \\ &= \frac{1}{2} \cdot \sum_{p=1}^m \sum_{q=1}^n 2 \cdot (o_{q,p} - t_{q,p}) \cdot \frac{\partial}{\partial w_{ij}} (o_{q,p} - t_{q,p}) = \sum_{p=1}^m \sum_{q=1}^n (o_{q,p} - t_{q,p}) \cdot \frac{\partial o_{q,p}}{\partial w_{ij}} \end{aligned}$$

Die Bestimmung der partiellen Ableitungen einer Ausgabe nach den Verbindungsgewichten wurde bereits in 1.5 erläutert. Mit

$$\delta_{j,p} = -\frac{\partial E_p}{\partial net_{j,p}} = \begin{cases} f'_{act}(net_{j,p}) \cdot (t_{j,p} - o_{j,p}) & \text{falls j Ausgabezelle} \\ f'_{act}(net_{j,p}) \cdot \sum_k \delta_{k,p} w_{jk} & \text{falls j verdeckte Zelle} \end{cases}$$

erhält man $\Delta_p w_{ij} = \eta \cdot o_{i,p} \cdot \delta_{j,p}$, wobei der Faktor η als Lernrate bezeichnet wird und zur Steuerung der Schrittweite dient. Dieses Gradientenverfahren ist als „Backpropagation“-Algorithmus bekannt. Werden die Gewichte geändert, nachdem alle Muster trainiert wurden, also $\Delta w_{ij} = \sum_p \Delta_p w_{ij}$, spricht man von „Batch“- oder „Offline Training“.

Alternativ hierzu kann beim „Online-Training“ nach jedem einzelnen Muster die ermittelte Gewichtsänderung angewendet werden. Zu Beginn werden alle Verbindungsgewichte mit zufälligen Werten initialisiert. Es ist vorteilhaft, den Grad der Eingangsverbindungen eines Neurons bei dieser Initialisierung zu berücksichtigen, um zu große Eingabewerte zu vermeiden. Sinnvoll ist eine Wahl von w_{ij} im Bereich $\pm \frac{1}{\text{indegree}(n_j)}$. Die Eingabepa-

rameter sind ebenfalls geeignet zu skalieren. Für große Beträge der Eingabewerte ist die geringe Steigung der Sigmoid-Funktionen problematisch. Es entsteht ein Plateau, auf dem ein Gradientenverfahren nur langsam fortschreitet. Wird der Gradient zu groß, sollte die Schrittweite gedrosselt werden. Generell kann nicht garantiert werden, daß das globale Minimum wirklich gefunden wird.

Allgemein wird nicht nur verlangt, daß das neuronale Netz die Ausgaben aller präsentierten Eingabemuster korrekt wiedergeben kann. Das trainierte Netz soll auch für neue Eingabemuster sinnvolle Ausgaben liefern. Dieser Punkt wird als Generalisierungsfähigkeit des Netzes bezeichnet.

3 Anwendung

Das in Abbildung 5 dargestellte Fachwerk ist mit den dargestellten Knotenlasten belastet. Bei einem E-Modul von 10^4 Ksi stehen folgende 6 Querschnitte zur Verfügung:

1.62 si 7.97 si 14.2 si

22.0 si 22.9 si 33.5 si

Für die möglichen 6^{10} Fachwerke soll das neuronale Netz die vertikale Durchbiegung des Knotens 2 bestimmen können. Hierzu werden 100 Testfachwerke erzeugt und berechnet, die als Grundlage der Lernens dienen. Die zehn Eingabeneuronen erhalten den Wert $\frac{1.62}{A_i} \cdot 2 - 1$. Für die Ausgabe wird die Verschiebung durch den Wert 25 geteilt und muß aufgrund der verwendeten logistischen Aktivierungsfunktion im Intervall $[0,1]$ liegen. Kontrolliert wird der mittlere quadratische Fehler (MSE). Das neuronale FF-Netz besteht aus einer versteckten Schicht mit 5 Neuronen. Zu Beginn werden die Verbindungsgewichte zufällig initialisiert und anschließend ein „Batch-Training“ durchgeführt. In Abbildung 6 ist der Einfluß der Lernrate ablesbar. Zu kleine

Lernraten wie $\eta=0.01$ im Beispiel, konvergieren sehr langsam. Wird die Schrittweite über einen gewissen Punkt gesteigert, wird das Verfahren instabil, man erhält Oszillationen wie hier für $\eta=0.5$, oder das Minimum wird in der vorgegebenen Schrittzahl nicht gefunden ($\eta=0.7$). Die optimale Lernrate für dieses Problem liegt bei $\eta=0.3$.

gibt die Verschiebung durch den Wert 25 geteilt und muß aufgrund der verwendeten logistischen Aktivierungsfunktion im Intervall $[0,1]$ liegen. Kontrolliert wird der mittlere quadratische Fehler (MSE). Das neuronale FF-Netz besteht aus einer versteckten Schicht mit 5 Neuronen. Zu Beginn werden die Verbindungsgewichte zufällig initialisiert und anschließend ein „Batch-Training“ durchgeführt. In Abbildung 6 ist der Einfluß der Lernrate ablesbar. Zu kleine

Lernraten wie $\eta=0.01$ im Beispiel, konvergieren sehr langsam. Wird die Schrittweite über einen gewissen Punkt gesteigert, wird das Verfahren instabil, man erhält Oszillationen wie hier für $\eta=0.5$, oder das Minimum wird in der vorgegebenen Schrittzahl nicht gefunden ($\eta=0.7$). Die optimale Lernrate für dieses Problem liegt bei $\eta=0.3$.

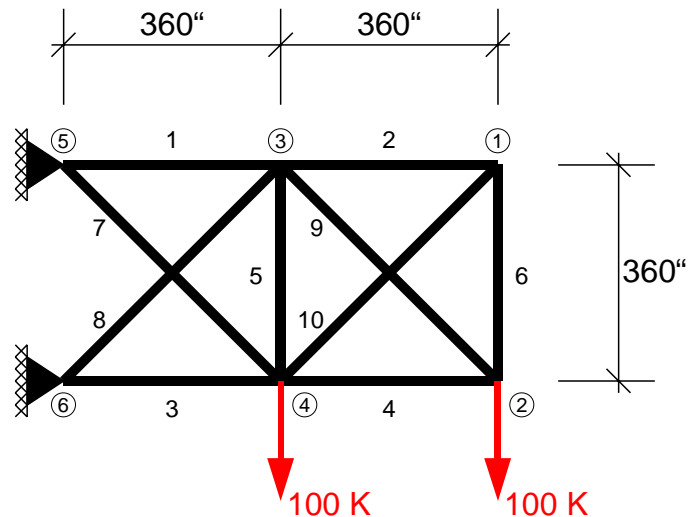


Abbildung 5: 10 Stäbe Fachwerk

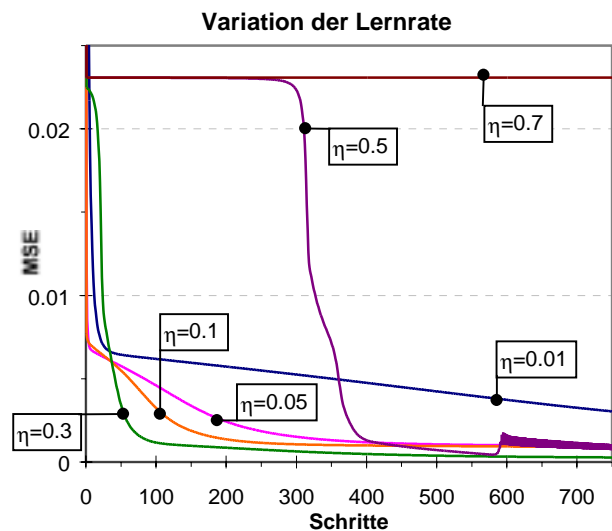


Abbildung 6: Lernrate

Es ist schwer die optimale Netztopologie a priori festzulegen. Werden mehr Neuronen verwendet kann genauer approximiert werden, es erhöht sich aber zugleich die Dimension des Optimierungsproblems. Die Netze mit mehr Neuronen lernen also langsamer. Diese Tendenzen sind deutlich in Abbildung 7 zu sehen. Ab 5 Neuronen sind die Verbesserungen nur noch gering.

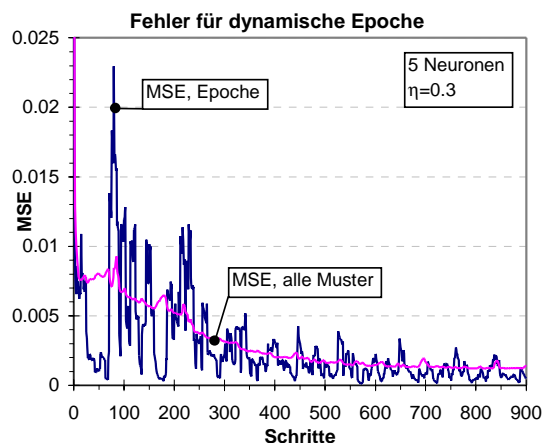


Abbildung 8: Dynamische Epoche

Epoche ist die Konvergenz für die Approximation des Problems erkennbar. Für einen mittleren quadratischen Fehler von $5.295 \cdot 10^{-4}$ war der maximale Fehler 0.014. Für 1000 neue Testbeispiele erreichte das Netz einen mittleren quadratischen Fehler von $6.032 \cdot 10^{-4}$ mit maximalem Fehler 0.022, was 0.55 mm entspricht.

Literatur

- [1] Bernreuther, Martin: „Diskrete Optimierung mit der Evolutionsstrategie auf parallelen Systemen“ in Fortschrittsberichte VDI Reihe 4 Nr. 135, Forum Bauinformatik, Junge Wissenschaftler forschen, Cottbus '96, S. 149 ff.; VDI-Verlag; Düsseldorf, 1996
- [2] Rojas, Raúl: Theorie der neuronalen Netze, Eine systematische Einführung; Springer; Berlin, Heidelberg, New York [u.a.], 1996
- [3] Zell, Andreas: Simulation Neuronaler Netze; Addison-Wesley; Bonn, Paris, Reading, Mass. [u.a.], 1994

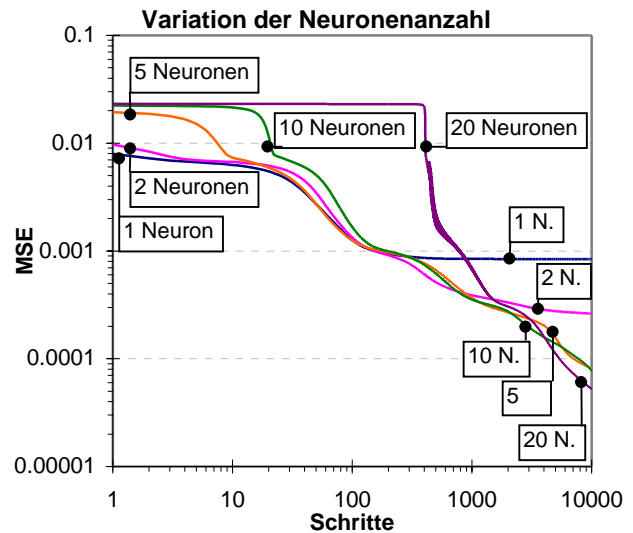


Abbildung 7: Anzahl Neuronen

Im folgender Versuch werden in einer Warteschlange für 100 Muster je Schritt ein Muster hinzugefügt und eines entfernt, so daß sich die Epoche dynamisch verändert. Für 900 Schritte wird der mittlere Fehler bezogen auf die jeweilige Epoche und auf die gesamten 1000 Muster in Abbildung 8 aufgetragen. Das Verfahren verbindet „Batch-“ und „Online-Training“, und trotz der Oszillationen des Fehlers für eine