Andreas Scherer
Überarbeitung: Hermann Helbig,
Wolfram Schiffmann

Kurs 01834

Künstliche Neuronale Netze

LESEPROBE

Fakultät für
Mathematik und Informatik

FernUniversität in Hagen

Kapitel 4

Die Grundlagen künstlicher Neuronaler Netze

Nachdem die Informationsverarbeitung in biologischen Systemen auf neuronaler Ebene behandelt wurde, wollen wir nun zur Beschreibung künstlicher Neuronaler Netze übergehen, die in enger Analogie zu natürlichen Nervensystemen entwickelt wurden. Zu diesem Zwecke betrachten wir Abbildung 4.1.

Das natürliche Neuron wird als Prozessorelement (Neuron, Unit) nachgebildet, das eine Neuronenfunktion realisiert, die n Eingaben $e_1, \ldots, e_n$ auf eine Ausgabe o (den Output) abbildet. Den Zuleitungen der Eingaben entsprechen die Dendriten des natürlichen Neurons, dem Prozessorelement entspricht der Zellkörper und der Outputableitung entspricht das Axon (vgl. Abb. 4.1a und 4.1b). Die Inputs werden nicht einfach unverändert auf das (künstliche) Neuron geschaltet, sondern jeweils mit einem entsprechenden Gewicht $w_i (1 \leq i \leq n)$ versehen. Dadurch wird die Wirkung der Synapsen bzw. deren Verbindungsstärke modelliert.

Die Nachbildung eines biologischen Neuronennetzes wird auf technischer Ebene dadurch erreicht, daß mehrere Neuronen zusammengeschaltet werden, indem die Outputs der einen Neuronen als Inputs weiterer Neuronen verwendet werden (vgl. Abb. 4.1c und 4.1d). Dabei bilden die Neuronen, die Inputs von außen erhalten, die Eingabeschicht und die Neuronen, die Outputs nach außen (d.h. nicht auf andere Neuronen) abgeben, die Ausgabeschicht. Alle anderen Neuronen gehören sogenannten verborgenen (d.h. nicht von außen direkt beobachtbaren bzw. beeinflußbaren) Schichten an.

In den beiden Teilen e) bzw. f) der Abbildung 4.1 ist die unterschiedliche materielle Basis für natürliche bzw. künstliche Neuronen angedeutet. Dabei ist darauf hinzuweisen, daß Neuronale Netze heute technisch meist nicht unmittelbar als elektronische Schaltungen realisiert werden, sondern vorwiegend programmtechnisch simuliert werden. Beide Vorgehensweisen setzen eine mathematische Modellierung voraus, die hier im Zentrum des Interesses stehen soll.

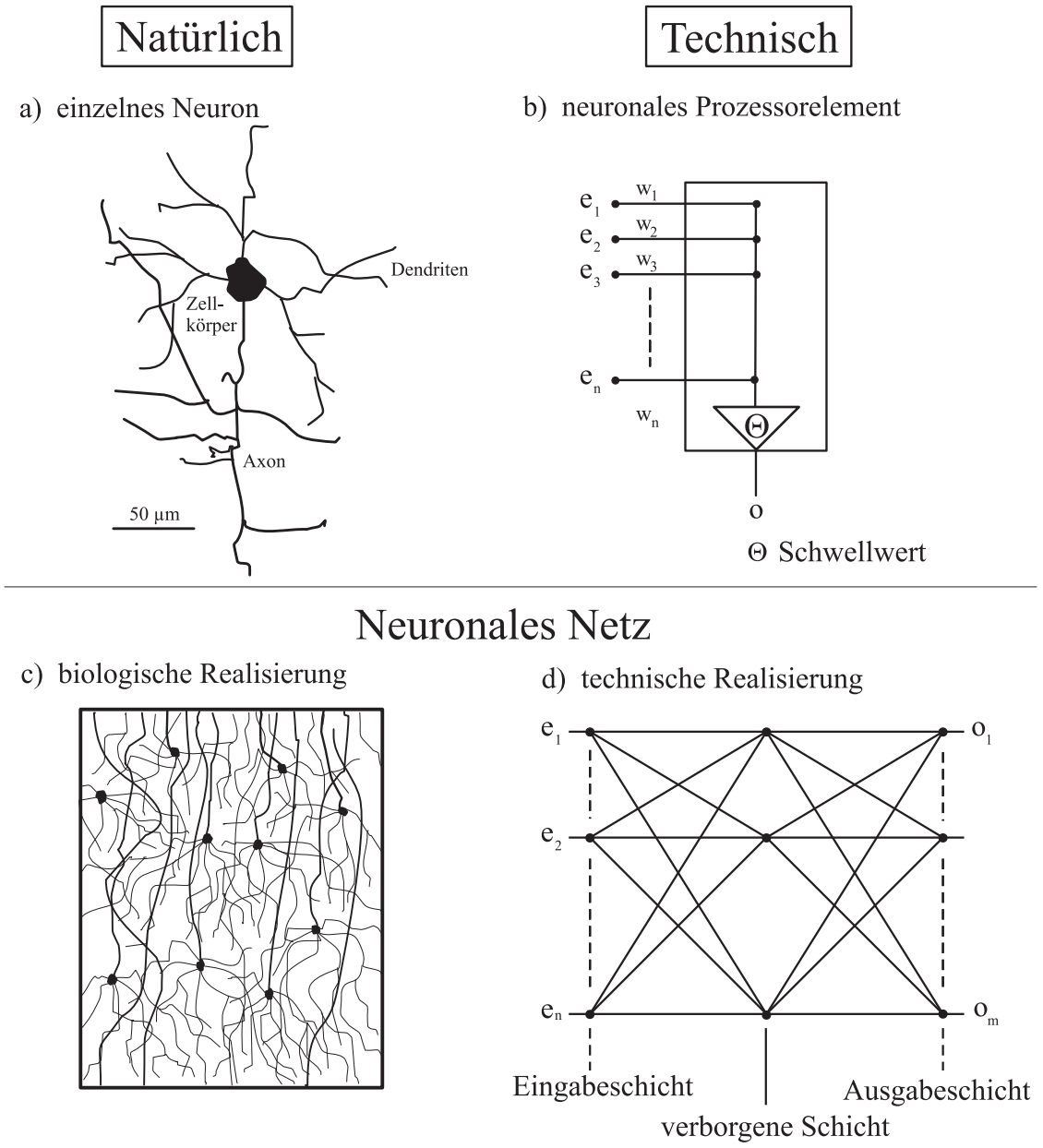


Abbildung 4.1: Analogien zwischen biologischen und künstlichen Neuronalen Netzen

4.1 Die charakteristischen Merkmale Neuronaler Netze

Neuronale Netze weisen bestimmte Gemeinsamkeiten auf, die man auch als die "building blocks" konnektionistischer Architekturen bezeichnet. Verschiedene Netztypen, die in den nachfolgenden Kapiteln behandelt werden, unterscheiden sich in der Regel in einem oder mehreren der nachstehenden Grundelemente.

1. **Das Neuron** (Zelle, Element, Unit): Das Neuron in einem künstlichen Neuronalen Netz ist die formalisierte Entsprechung der Nervenzelle in biologischen Systemen. Zur Beschreibung der Funktion eines Neurons werden in der Regel folgende Bestandteile unterschieden: Neuron

   a) **Die Inputfunktion oder Propagierungsfunktion** (propagation function): Sie beschreibt das Verhalten eines Neurons im Hinblick auf seine integrative Wirkung, indem sie alle Eingaben zu einem Wert, zum sogenannten *Nettoinput* zusammenfaßt. In der Regel werden die Eingaben der vorgeschalteten Neuronen entsprechend den Gewichtungen der Eingangsdaten aufsummiert. Nettoinput

   b) **Die Aktivierungsfunktion** (activation function): Sie bestimmt, wie sich der Aktivierungszustand zum Zeitpunkt $t + 1$ aus dem Aktivierungszustand im Zeitpunkt t und dem in diesem Zeitpunkt anliegenden Nettoinput errechnet. Aktivierungsfunktion

   c) **Die Aktivierungszustand** (activation state): Er widerspiegelt den aktuellen Erregungszustand eines Neurons (und entsprechend verallgemeinert auch eines ganzen Netzes). Er wird durch den Wert der Aktivierungsfunktion an einem Neuron (bzw. durch das Muster von Neuronen-Aktivierungen über ein ganzes Netz) bestimmt. Aktivierungszustand

   d) **Die Ausgabefunktion** (output function): Sie legt den tatsächlichen Ausgabewert eines Neurons in Abhängigkeit vom aktuellen Aktivierungszustand fest. Ausgabefunktion

2. **Der Netzwerkgraph** (topology): Ein Neuronales Netz kann als gerichteter Graph aufgefaßt werden, dessen Kanten die gewichteten Verbindungen zwischen einzelnen Neuronen darstellen. Der Netzwerkgraph wird wesentlich durch die Matrix $W = \{w_{ij}\}$ der Verbindungsstärken zwischen i-tem und j-tem Neuron bestimmt (pattern of connectivity). Der Netzwerkgraph beschreibt die Topologie des Netzes. Die Knoten des Graphen werden typischerweise in Schichten angeordnet, die durch bestimmte Restriktionen hinsichtlich der Verbindung zwischen den Knoten Netzwerkgraph Netztopologie

jeder Schicht charakterisiert sind. Die Knoten, an die von außen Eingaben übermittelt werden, bilden die Eingabeschicht. Die Knoten, die nach außen Ergebnisse (Outputs) abgeben, bilden die Ausgabeschicht (alle anderen Schíchten werden "innere" oder "verdeckte" Schichten gennannt).

3. **Die Lernregel** (learning rule): Diese legt fest, wie bestimmte Netzparameter (in den meisten Fällen sind dies die Gewichte der Neuronenverbindungen) verändert werden sollen, damit das Netz eine gewünschte Leistung (Klassifizierung, Ausbildung bestimmter Assoziationen usw.) hervorbringt.

Lernregel

Grundlage der Darstellungen sind im folgenden Zell [Zel94] und Rumelhart und McClelland [RM86].

4.2 Das Neuron

Die technische Realisierung eines Neurons wird wie in Abbildung 4.2 dargestellt beschrieben.

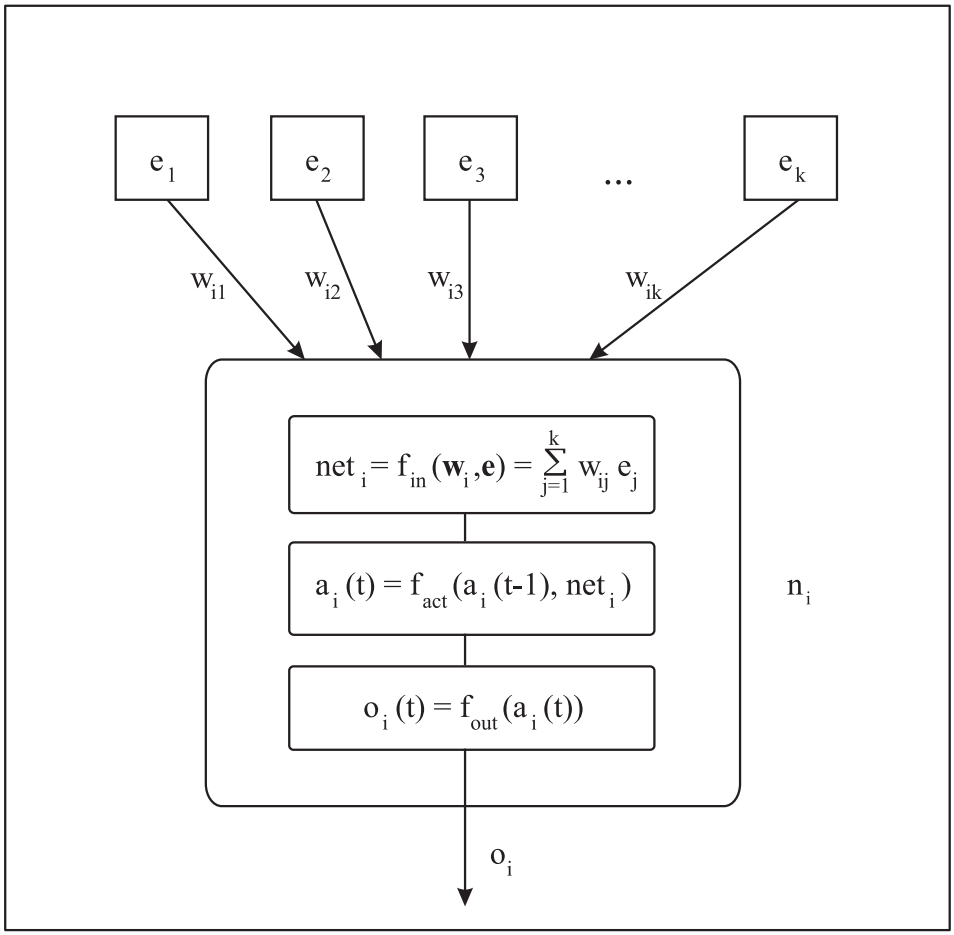


Abbildung 4.2: Aufbau eines Neurons

Als Konventionen zur Beschreibung der Bestimmungsstücke eines Neuronalen Netzes werden die folgenden verwendet:

i	Index/Nummer der neuronalen Einheit	
e_j	Eingabewert (*Input*), der entweder von außen vorgegeben wird oder der Output o_j eines anderen Neurons (des Unit mit der Nummer j) in einem Neuronalen Netz ist	
w_{ij}	Faktor, mit dem die j-te Eingabe des i-ten Neurons gewichtet wird (im folgenden kurz *Gewicht* genannt)	Gewicht
θ	*Schwellwert*; dieser kann ohne Einschränkung der Allgemeinheit weggelassen und durch einen zusätzlichen festen Input $e_o = -1$ (den sogenannten Bias) und ein entsprechendes Gewicht w_{io} ersetzt werden	Schwellwert
o_i	*Output* des i-ten Elements; $f_{out}(a_i)$	
$\sum_{j=1}^{k} w_{ij}e_j = \text{net}_i$	Nettoinput des i-ten Elements (*Inputfunktion* = $f_{in}(\mathbf{w}_i, e)$)	Inputfunktion
$a_i(t) = f_{act}(a_i(t-1), net_i)$	Aktivierung des i-ten Elements; $a_i(t)$ wird auch als *Aktivierungsfunktion* oder als *Transferfunktion* bezeichnet ($\rightsquigarrow f_{act}$). Oft wird für f_{act} nur eine Schwellwertfunktion angesetzt: $a_i = f_{act}(net_i) = s(net_i)$.	Transferfunktion

Die Inputs eines Neurons oder der Inputschicht werden oft zu einem Vektor $\mathbf{e} = \{e_1, \ldots, e_k\}$ zusammengefaßt. Analoges gilt für den Output $\mathbf{o} = \{o_1, \ldots, o_m\}$ einer Gruppe von m Neuronen (z.B. aller Neuronen der Ausgabeschicht) oder für die Gewichte $\mathbf{w}_j = \{w_{j1}, w_{j2}, \ldots, w_{jk}\}$ aller Eingaben zum Neuron mit der Nummer j. Die Matrix aller Gewichte wird mit $\mathbf{W} = \{w_{ij}\}$ bezeichnet. Vektoren und Matrizen werden im folgenden durch Fettdruck hervorgehoben.

Die Verarbeitungseinheit 'Neuron' kann durch die Angabe des Aktivierungszustandes, der verwendeten Propagierungsfunktion, der Aktivierungsfunktion und der Ausgabefunktion vollständig beschrieben werden. Das Neuron n_i steht in Verbindung zu seinen Vorgängerneuronen $n_1, \ldots, n_k$. Den Verbindungen von den $n_1, \ldots, n_k$ zu n_i sind Gewichte w_{ij} zugeordnet. Die Aktivierung des Neurons n_i wird nach entsprechender Zwischenverarbeitung durch

dieses Neuron an andere Neuronen oder an die Gesamtausgabe des Netzes weitergeleitet. Im weiteren soll untersucht werden, welche Möglichkeiten es zur Modellierung des "Inneren" des Neurons gibt.

4.2.1 Die Inputfunktion oder Propagierungsfunktion

Die Inputfunktion f_{in} gibt an, wie sich der Nettoinput net_i des Neurons mit der Nummer i aus den Eingaben $\mathbf{e} = \{e_1, \ldots, e_k\}$ zu diesem Neuron errechnet. Der Vektor $\mathbf{e}$ spezifiziert dabei entweder Eingaben, die von außen an das Neuron herangeführt werden, oder die e_j $(1 \leq j \leq k)$ entsprechen jeweils Ausgaben o_j von anderen Neuronen mit der Nummer j $(1 \leq j \leq k)$. Folgende Input- bzw. Propagierungsfunktionen werden für die Beschreibung Neuronaler Netze verwendet:

Summation der gewichteten Eingaben

$$\text{net}_i = \sum_{j=1}^{k} w_{ij} \cdot e_j \tag{4.1}$$

Maximalwert der gewichteten Eingaben

$$\text{net}_i = \max_j (w_{ij} \cdot e_j) \tag{4.2}$$

Produkt der gewichteten Eingaben

$$\text{net}_i = \prod_{j=1}^{k} w_{ij} \cdot e_j \tag{4.3}$$

Minimalwert der gewichteten Eingaben

$$\text{net}_i = \min_j (w_{ij} \cdot e_j) \tag{4.4}$$

Die Eingaben e_j an das Neuron n_i werden in allen hier angegebenen Propagierungsfunktionen mit einem entsprechenden Gewicht multipliziert. Anschließend erfolgt eine Aufaddierung, Multiplikation oder Minimum- bzw. Maximumbildung. Neuronen mit Inputfunktion (4.1) werden auch als *Sigma-Units* (Σ-Units) und solche mit Inputfunktion (4.3) als *Pi-Units* (Π-Units) bezeichnet. Eine Kombination von beiden heißt dementsprechend *Sigma-Pi-Unit*. Als Definitions- und Wertebereiche für die beschriebenen Inputfunktionen kommen sowohl diskrete als auch kontinuierliche Wertemengen (z.B. die Menge aller reellen Zahlen) in Frage. Die Propagierungsfunktion (4.1) ist eine der am häufigsten verwendeten Vorschriften zur Berechnung des Nettoinputs bei Neuronalen Netzen. Sie kann unter Verwendung der Vektordarstellung $\mathbf{e} = \{e_1, \ldots, e_k\}$ für den Eingabevektor und $\mathbf{w}_i = \{w_{i1}, w_{i2}, \ldots, w_{ik}\}$ für den Gewichtsvektor des i-ten Neurons auch in Form eines Skalarprodukts geschrieben werden:

Sigma-Units
Pi-Units
Sigma-Pi-Unit

$$\text{net}_i = \sum_{j=1}^{k} w_{ij} e_j = \mathbf{w}_i \cdot \mathbf{e} \tag{4.5}$$

4.2.2 Aktivierungsfunktion und Aktivierungszustand

Allgemeine Aktivierungsfunktion

Der Aktivierungszustand $a_i(t)$ beschreibt den Zustand eines Neurons n_i zum Zeitpunkt t. Man kann $a_i(t)$ auch als inneren Zustand oder Erregung zum Zeitpunkt t auffassen. Die Aktivierungsfunktion f_{act} (auch Transferfunktion, activation function genannt) gibt an, wie sich aus dem vorherigen Zustand und dem aktuellen Input der Zustand des Neurons zum Zeitpunkt $t+1$ berechnet.

$$a_i(t+1) = f_{act}(a_i(t), \text{net}_i(t)) \tag{4.6}$$

Die Berücksichtigung des zeitlich vorangehenden Aktivierungszustandes stellt eine Modellierungsoption dar, die häufig **nicht** eingesetzt wird. Praktisch relevant sind folgende Klassen von Aktivierungsfunktionen:

- lineare Aktivierungsfunktionen,
- Schwellwertfunktionen,
- sigmoide Funktionen (Sigmoide)

Die Wahl der Aktivierungsfunktion hängt vom konkret vorliegenden Anwendungsproblem ab und bestimmt wesentlich den Netztyp.

Lineare Aktivierungsfunktion

Lineare Aktivierungsfunktion. Im einfachsten Falle werden lineare Funktionen zur Berechnung der Aktivierung verwendet. Der Hauptvorteil besteht darin, daß sich solche Funktionen leicht berechnen und schnell implementieren lassen. Lineare Aktivierungsfunktionen werden nur dann eingesetzt, wenn das Netz keine verdeckten Schichten besitzt (d. h. nur aus einer Ein- und Ausgabeschicht besteht) und die Korrelation zwischen Ein- und Ausgabedaten linear ist bzw. hinreichend genau linear approximiert werden kann.

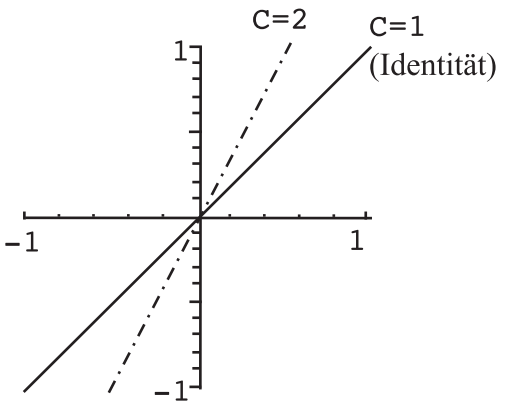


Abbildung 4.3: Identität als Aktivierungsfunktion

Lineare Aktivierungsfunktionen werden durch Gleichung (4.7) beschrieben, ein Beispiel für eine solche Funktion ist die Identität (vgl. Abb. 4.3). Es kommen aber auch abschnittsweise lineare Funktionen zum Einsatz (vgl. Gleichung (4.8) und Abb. 4.4).

$$o_i = c_i \cdot \text{net}_i \qquad \text{(lineare Outputfunktion)} \tag{4.7}$$

$$o_i = \begin{cases} -1 & \text{wenn } \text{net}_i \leq -1 \\ 1 & \text{wenn } \text{net}_i \geq 1 \\ \text{net}_i & \text{sonst} \end{cases} \qquad \begin{array}{l}\text{abschnittsweise} \\ \text{lineare Funktion}\end{array} \tag{4.8}$$

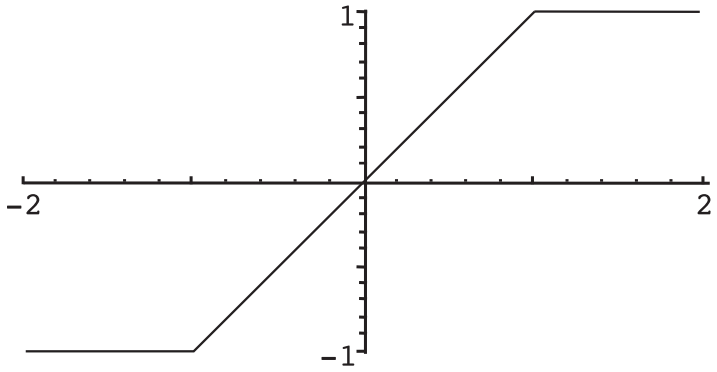
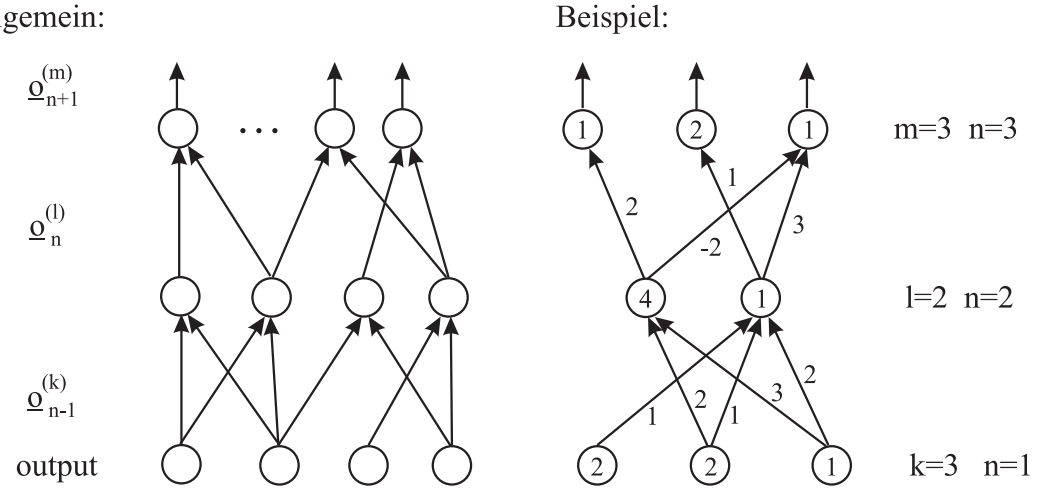


Abbildung 4.4: Abschnittsweise lineare Aktivierungsfunktion

Die Verwendung von linearen Aktivierungsfunktionen ist – wie bereits angedeutet – nur sinnvoll, wenn das Neuronale Netz aus zwei Schichten (der Eingabe- und der Ausgabeschicht) besteht. Ein mehrschichtiges Netz mit $n > 2$ Schichten kann mit linearen Aktivierungsfunktionen nicht mehr leisten als ersteres. Es kann immer auf ein zweilagiges Netz reduziert werden, wie nachfolgende Rechnung zeigt (vgl. Abb. 4.5).

Sei $\mathbf{o}_n^{(l)}$ der Outputvektor der n-ten Schicht, wobei l die Anzahl der Komponenten des Vektors, d.h. die Zahl der Neuronen in dieser Schicht, angibt. Sei weiter $\mathbf{W}_{n,(n-1)}^{(l,k)}$ die Matrix der Gewichtsfaktoren vom Typ (l,k), d.h. mit l Zeilen und k Spalten, die die Gewichte zwischen Neuronen der $(n-1)$-ten und der n-ten Schicht zusammenfaßt. Ein Matrixelement w_{ij} dieser Matrix bezeichnet das Gewicht der Verbindung vom j-ten Neuron der $(n-1)$-ten Schicht $(1 \leq j \leq k)$ zum i-ten Neuron der n-ten Schicht $(1 \leq i \leq l)$. Bei der Verwendung einer linearen Aktivierungsfunktion und Gleichsetzen von Aktivierung und Output gilt für das j-te Neuron $o_j = a_j = c_j \cdot \text{net}_j$. Schreibt man noch die Linearfaktoren der n-ten Schicht aus Gleichung (4.7) als Diagonalmatrix $\mathbf{C}_n^{(l,l)}$ vom Typ (l,l) (diese sind in Abb. 4.5 in den Kreisen angegeben) und repräsentiert die Nettoinputs jeder Schicht analog zu den Outputs in Vektorschreibweise, dann gelten mit den Bezeichnungen aus Abbildung 4.5 folgende



[unterer Index: Nummer der Schicht; oberer Index: Zahl der Neuronen]

Abbildung 4.5: Musternetz zur Schichtenreduktion

Gleichungen (wie die oberen Indizes, d.h. die Typangaben zeigen, ist in den Formeln auch die korrekte Verkettung der Matrizen und Vektoren gesichert).

$$\begin{aligned}\mathbf{o}_{n+1}^{(m)} &= \mathbf{C}_{n+1}^{(m,m)} \cdot \mathbf{net}_{n+1}^{(m)} = \mathbf{C}_{n+1}^{(m,m)} \cdot \mathbf{W}_{(n+1),\,n}^{(m,l)} \cdot \mathbf{o}_n^{(l)} \\ &= \mathbf{C}_{n+1}^{(m,m)} \cdot \mathbf{W}_{(n+1),\,n}^{(m,l)} \cdot \mathbf{C}_n^{(l,l)} \cdot \mathbf{net}_n^{(l)} \\ &= \mathbf{C}_{n+1}^{(m,m)} \cdot \mathbf{W}_{(n+1),\,n}^{(m,l)} \cdot \mathbf{C}_n^{(l,l)} \cdot \mathbf{W}_{n,\,(n-1)}^{(l,k)} \cdot \mathbf{o}_{n-1}^{(k)}\end{aligned} \tag{4.9}$$

setzt man noch:

$$\overline{\mathbf{W}}_{(n+1),\,(n-1)}^{(m,k)} = \mathbf{W}_{(n+1),\,n}^{(m,l)} \cdot \mathbf{C}_n^{(l,l)} \cdot \mathbf{W}_{n,\,(n-1)}^{(l,k)} \tag{4.10}$$

so erhält man:

$$\mathbf{o}_{n+1}^{(m)} = \mathbf{C}_{n+1}^{(m,m)} \cdot \overline{\mathbf{W}}_{(n+1),\,(n-1)}^{(m,k)} \cdot \mathbf{o}_{n-1}^{(k)} \tag{4.11}$$

Diese Gleichung drückt genau den behaupteten linearen Zusammenhang zwischen $(n-1)$-ter und $(n+1)$-ter Neuronenschicht nur unter Verwendung einer modifizierten Gewichtsmatrix aus. Durch wiederholte Anwendung dieses Reduktionsverfahrens läßt sich jedes so aufgebaute mehrschichtige Netz auf zwei Schichten reduzieren.

Für das Beispiel aus Abbildung 4.5 ergibt sich durch Anwendung der Reduktionsgleichung (4.10) das in Abbildung 4.6 dargestellte äquivalente Neuro-

nale Netz aufgrund nachstehender Berechnung:

$$\begin{aligned}\overline{W}_{31}^{(3,3)} &= W_{32}^{(3,2)} \cdot C_2^{(2,2)} \cdot W_{21}^{(2,3)} \\ &= \begin{pmatrix} 2 & 0 \\ 0 & 1 \\ -2 & 3 \end{pmatrix} \cdot \begin{pmatrix} 4 & 0 \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 0 & 2 & 3 \\ 1 & 1 & 2 \end{pmatrix} \\ &= \begin{pmatrix} 0 & 16 & 24 \\ 1 & 1 & 2 \\ 3 & -13 & -18 \end{pmatrix}\end{aligned}$$

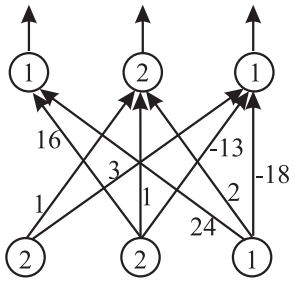


Abbildung 4.6: Reduziertes Netz (äquivalent zu Beispielnetz aus Abb. 4.5)

Schwellwert-funktion

Schwellwertfunktion. Eine einfache Variante, den Aktivierungszustand eines Neurons zu berechnen, stellt die sogenannte Schwellwertfunktion dar (vgl. Gleichung (4.12) und Abb. 4.7). Sie wird vorwiegend in den klassischen Netztypen (z. B. im Perzeptron, s. Kap. 5) verwendet.

$$o_i = \begin{cases} 1 & \text{falls } \text{net}_i \geq \theta_i \\ 0 & \text{sonst} \end{cases} \tag{4.12}$$

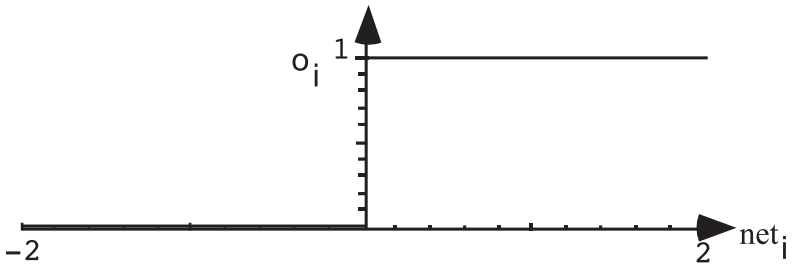


Abbildung 4.7: Schwellwertfunktion

Diese Funktion kann besonders effizient ausgewertet werden und ist auch leicht zu implementieren. In neueren Netzwerktypen wird jedoch häufig die

Differenzierbarkeit der Aktivierungsfunktion gefordert (z. B. in mehrschichtigen Netzen, s. Kap. 5). Da Schwellwertfunktionen an der Sprungstelle nicht differenzierbar sind, verbietet sich ihre Anwendung innerhalb dieser Verfahren.

Sigmoide

Sigmoide. Sigmoide Aktivierungsfunktionen (das sind Funktionen, deren graphische Darstellung einen S-förmigen Verlauf zeigt) werden in einer Reihe von Netzwerktypen angewendet. Sie spielen eine große Rolle bei der Modellierung nichtlinearer Beziehungen zwischen Ein- und Ausgabedaten. Zudem erfüllen sie folgende wesentliche Anforderungen:

- Stetigkeit,
- Differenzierbarkeit,
- Monotonie.

Hier seien nur zwei sigmoide Aktivierungsfunktionen und deren Ableitungen beschrieben:

Fermi-Funktion

1) Die Fermi-Funktion

Die nach dem Physiker Fermi benannte Funktion, die auch eine wichtige Rolle in der Thermodynamik spielt (und zwar in der Fermi-Statistik); sie wird mitunter auch als '*logistische Funktion*' bezeichnet und hat die Form:

logistische Funktion

$$s_F(x) = \frac{1}{1 + e^{-x/T}} \tag{4.13}$$

Ihr Wertebereich ist das offene Intervall $]0, 1[$ (vgl. Abb. 4.8); sie hat die besondere Eigenschaft, daß sich ihre erste Ableitung durch die Funktion selbst ausdrücken läßt:

$$\frac{d}{dx} s_F(x) = s_F(x)(1 - s_F(x)) \cdot \frac{1}{T} \tag{4.14}$$

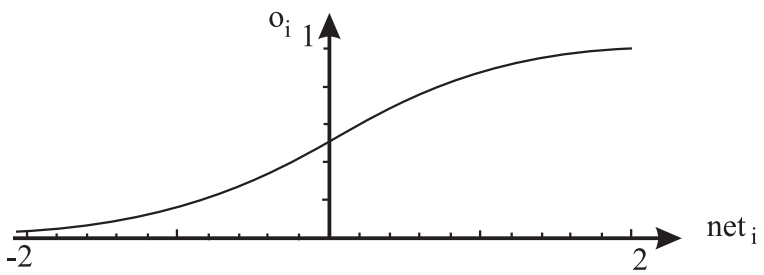


Abbildung 4.8: Fermi-Funktion für $T = 0.5$

In Analogie zur Physik wird T auch als Temperaturparameter bezeichnet, der nach (4.14) den Anstieg der Fermi-Funktion bestimmt (kleines T bedeutet starken Anstieg in der Umgebung von $x = 0$).

Tangens hyperbolicus

2) Der Tangens hyperbolicus

Auch der Tangens hyperbolicus (Symbol: tanh) wird als Transferfunktion verwendet. Die entsprechende Funktionsgleichung ist definiert durch:

$$s_{\tanh}(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{4.15}$$

Der Wertebereich ist das offene Intervall $]-1;1[$ (vgl. Abb. 4.9). Die Ableitung des Tangens hyperbolicus berechnet sich wie folgt:

$$\frac{d}{dx} s_{\tanh}(x) = 1 - s^2_{\tanh}(x) \tag{4.16}$$

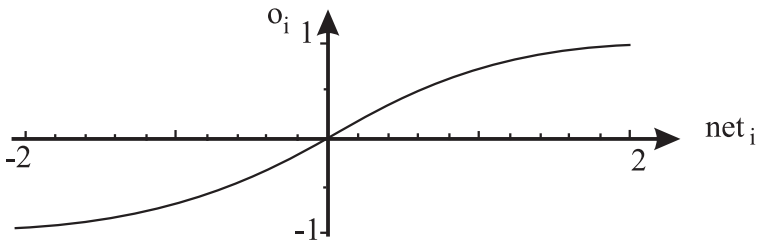


Abbildung 4.9: Tangens hyperbolicus

4.2.3 Die Ausgabefunktion

Die Ausgabefunktion bildet den aktuellen Zustand des Neurons auf einen gewünschten Wertebereich ab. Viele Autoren führen die Ausgabefunktion nicht gesondert auf, sondern verstehen diese als Bestandteil der Aktivierungsfunktion. Bei hybriden Netzen mit verschiedenen Aktivierungsfunktionen, die unterschiedliche Definitions- und Wertebereiche haben, muß jedoch eine explizite Ausgabefunktion eingesetzt werden. An dieser Stelle führen wir daher formal die Ausgabefunktion f_{out} ein, für die allerdings – wie bereits bemerkt – häufig die Identitätsfunktion verwendet wird.

Allgemeine Ausgabefunktion

$$o_i(t+1) = f_{out}(a_i(t+1)) = f_{out}(f_{act}(a_i(t), \mathrm{net}_i(t))) \tag{4.17}$$

4.3 Der Netzwerkgraph

Die Leistungsfähigkeit konnektionistischer Verfahren beruht allgemein auf der Verbindung vieler einfacher Prozessorelemente zu einer netzartigen aktiven

Struktur, die in der Lage ist, komplexere Problemstellungen zu lösen. Man kann diese Struktur als einen endlichen Graphen darstellen. In der Fachsprache existieren einige Begriffe, die unterschiedliche Topologievarianten benennen. Diese sollen im folgenden kurz eingeführt werden.

Neuronales Netz als Graph

Definition 4.1 Mathematisch ist ein Neuronales Netz ein gerichteter Graph, der aus einer Menge von Knoten, den Neuronen $N = \{n_1, \ldots, n_m\}$ und einer Menge von gewichteten Verbindungen, das sind markierte Kanten $A = \{a_1, \ldots, a_p\}$ besteht, mit $A \subseteq N \times N$ und einer Markierungsfunktion $\mu(a) = w$ (mit $a \in A$ und $w \in W$, dem Wertebereich der Gewichte). □

In vielen Neuronalen Netzen läßt sich die Gesamtheit N der Neuronen vollständig in Partitionen $S_n (1 \leq n \leq k)$ zerlegen, wobei $S_n \subseteq N$ und $\bigcup_{i=1}^{k} S_i = N$ und $S_i \cap S_j = \oslash$ für $i \neq j$. Dieser Zerlegung liegt die Vorstellung zugrunde, daß die Neuronen einer Partition – diese wird auch *Schicht* genannt – eine in paralleler Arbeit realisierte, gemeinsame Funktion im Gesamtverband der Neuronen besitzen. Dem Begriff der Schicht liegt also das Konzept von Funktionseinheiten bzw. der Modularisierung von Netzen zugrunde.

Schichten

Folgende wichtige Netztopologien lassen sich unterscheiden:

Netztopologie

- Feedforward-Netze (FF-Netze)
  - 1. Ordnung,
  - 2. Ordnung.
- Feedback-Netze (FB-Netze)

  Dazu gehören Feedback-Netze mit:
  - indirektem Feedback,
  - Lateralverbindung,
  - vollständiger Vermaschung.

4.3.1 FF-Netze

Bei dieser Topologievariante kann man die Schichten des entsprechenden Netzes ordnen. Seien also die $S_n \, (1 \leq n \leq k)$ zu einem Netz gegeben.

FF-Netz 1. Ordnung

Man spricht von einem FF-Netz 1. Ordnung genau dann, wenn es nur gerichtete Verbindungen zwischen Schicht S_n und Schicht S_{n+1} gibt. Man sagt auch, das Netz ist schichtweise verbunden (vgl. Abb. 4.10a).

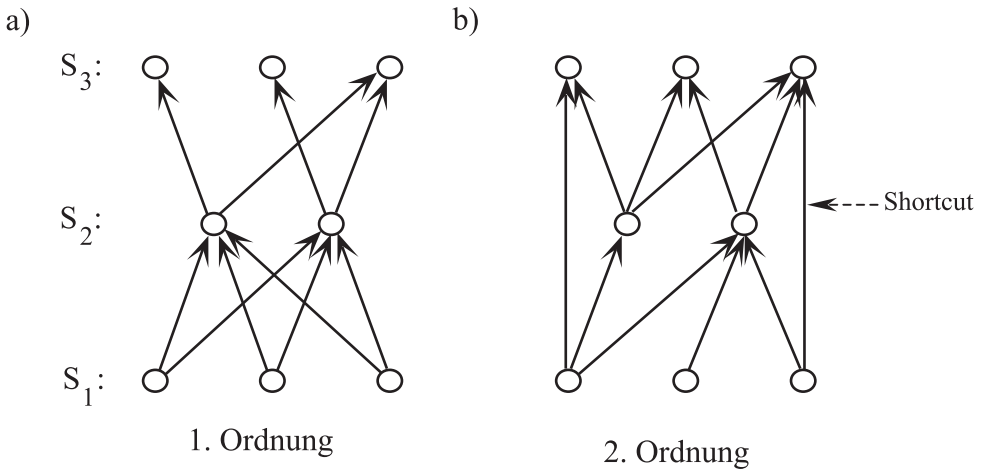


Abbildung 4.10: Feedforward-Netze

FF-Netz 2. Ordnung

Ein Netz heißt FF-Netz 2. Ordnung genau dann, wenn Neuronen einer Schicht S_n auch gerichtete Verbindungen zu Neuronen aus höheren Schichten S_{n+m} mit $m > 1$ aufweisen. Man nennt Verbindungen in FF-Netzen, die von Schicht S_n nach S_{n+m} mit $m > 1$ verlaufen, *shortcut connections*. (vgl. Abb.4.10b).

Ein Neuronales Netz NN ist genau dann ein Feedforward-Netz, wenn der entsprechende gerichtete Graph zyklenfrei ist.

4.3.2 FB-Netze

Bei den rückgekoppelten Netzen verlaufen gerichtete Verbindungen von Neuronen höherer Schichten zu Neuronen niedrigerer Schichten bzw. zu Neuronen der gleichen Schicht. Diese Art von Rückkopplung kann bei einer Reihe von Problemen mit komplexer zeitlicher Dynamik sehr wertvoll sein (vgl. Abb. 4.11).

direkte Rückkopplung

Ein Feedback-Netz weist direkte Rückkopplungen auf, wenn ein Neuron $n \in N$ seine Ausgabe direkt wieder als Eingabe erhält und bei dem nächsten Bearbeitungsschritt in die Berechnung seines neuen Aktivierungszustandes einfließen läßt (vgl. Abb. 4.11a).

indirekte Rückkopplung

Ein Feedback-Netz mit indirekter Rückkopplung weist Verbindungen von Schichten S_{n+k} zu Schichten S_n auf, mit $(k \geq 1)$, und es gibt gleichzeitig gerichtete direkte oder indirekte Verbindungen der Neuronen von Schicht S_n zu Schichten S_{n+k}. Rückkopplungen werden häufig dazu verwendet, bestimmte Bereiche der Eingabe als besonders relevant hervorzuheben (vgl. Abb. 4.11b).

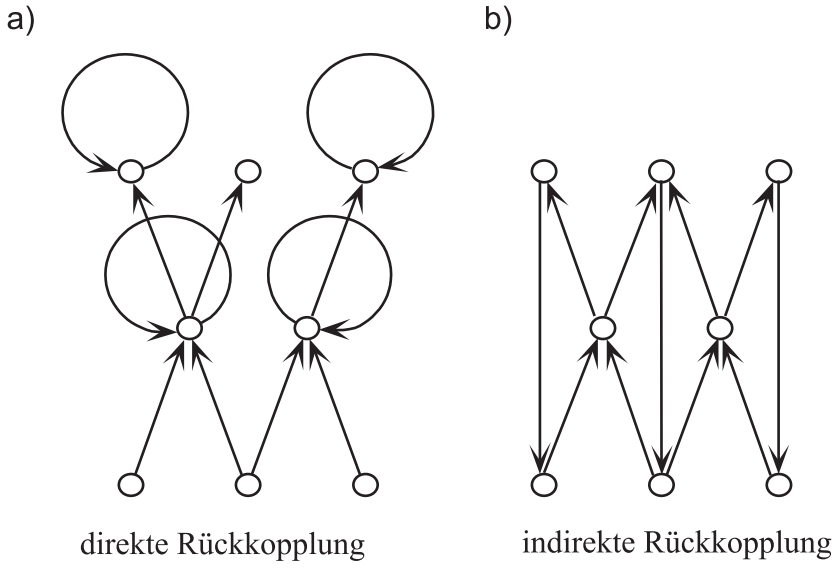


Abbildung 4.11: Rückkopplung in Neuronalen Netzen

Netze mit Lateralverbindung (vgl. Abb 4.12a) weisen Kanten zwischen Neuronen der gleichen Schicht auf. Häufig wird diese Art von Verbindung dazu benutzt, die Aktivität eines Neurons hervorzuheben, während gleichzeitig der Einfluß benachbarter Neuronen gedämpft wird (vgl. hierzu die Ausführungen zur lateralen Hemmung in Abschnitt 3.4 und zu Kohonen-Netzen in Kurseinheit 4).

Lateralverbindung

Ein Netz heißt vollständig vermascht, wenn jedes Neuron $n_i \in N$ zu jedem Neuron $n_j \in N$ $(i \neq j)$ eine gerichtete Verbindung aufweist (vgl. Abb. 4.12b). Diese Topologievariante wird bei sogenannten Hopfield-Netzen verwendet.

vollständige Vermaschung

4.4 Die Lernregel

Eine der zentralen Eigenschaften Neuronaler Netze ist ihre Lernfähigkeit. Diese Fähigkeit impliziert im elementarsten Fall, daß ein Netz früher vorgegebene Paare von Ein- und Ausgabe-Daten bei Vorgabe des Inputs einfach reproduzieren kann (analog zum "Auswendiglernen" beim Menschen). Diese Leistung allein würde aber für interessante Anwendungen noch nicht genügen. Was man tatsächlich anstrebt, ist ein generalisiertes Lernen. Hierzu muß zunächst eine geeignete netzinterne Repräsentation der Eingabedaten ausgebildet werden. Die Lernregel legt dabei fest, nach welcher Strategie das Netz zu verändern ist, um diese interne Repräsentation zu erzielen. Es gibt grundsätzlich mehrere Möglichkeiten, ein solches Lernverfahren zu realisieren:

Lernverfahren

1. Aufbauen und Löschen von Verbindungen,
2. Hinzufügen und Löschen von Neuronen,

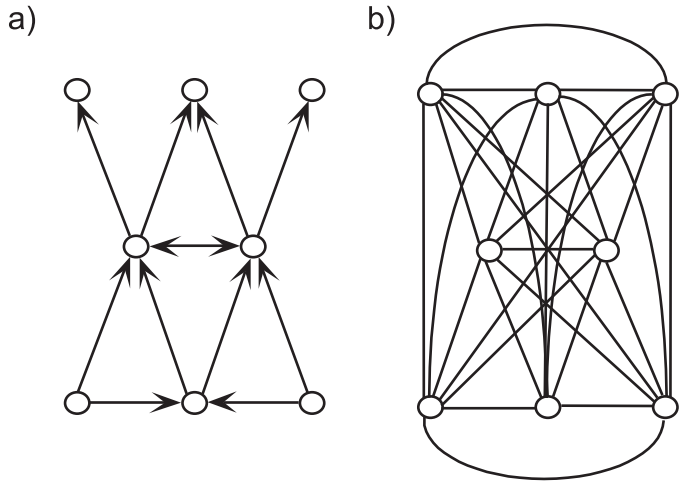


Abbildung 4.12: Lateralverbindung und vollständige Vermaschung

3. Modifikation der Gewichte von Verbindungen,

4. Modifikation von Parametern innerhalb des Neurons (Variation von internen Parametern der Aktivierungsfunktion, Schwellwertänderungen etc.).

Variante 3 stellt eine der am häufigsten eingesetzten Lernstrategien dar. Insbesondere kann man mit dieser Methode Variante 1 simulieren, indem nicht existierende Verbindungen zwischen zwei Neuronen einfach mit 0 gewichtet werden. In den folgenden Kurseinheiten werden verschiedene Lernverfahren detailliert behandelt, die eine oder mehrere der oben angeführten Strategien einsetzen. Zunächst sollen aber einige der bekanntesten Lernregeln unabhängig von ihrem Einsatz in bestimmten Netzen eingeführt werden.

4.4.1 Hebbsche Lernregel

Bereits Ende der 40er Jahre stellte Donald Hebb (Hebb, 1949) die Theorie auf, daß assoziatives Speichern in biologischen Systemen auf Prozessen beruht, die die Verbindung zwischen Nervenzellen modifizieren.

Hebbsche Lernregel

In der nach ihm benannten Hebbschen Regel führt er Lernen auf das folgende einfache Prinzip zurück. Wenn zwei Neuronen n_i und n_j zur gleichen Zeit aktiv sind **und** eine Verbindung zwischen beiden Neuronen existiert, so ist das Gewicht dieser Verbindung in definierter Weise zu verstärken:

$$\Delta w_{ij} = \eta \cdot a_i \cdot o_j \tag{4.18}$$

Hierbei ist Δw_{ij} die Veränderung des Verbindungsgewichtes von Neuron n_j zum Neuron n_i, a_i die Aktivierung von Neuron n_i, o_j die Ausgabe von Neuron n_j und η eine geeignet zu wählende Lernrate.

In technischen Systemen, die sich an die Hebbsche Regel anlehnen, legt die *Lernrate* η fest, wie groß die Veränderung eines Verbindungsgewichtes innerhalb eines Lernschritts sein soll. Sie wird meist als Konstante angegeben und ist ein kritischer Parameter, der sorgfältig ausgewählt werden sollte. In Kap. 5, Backpropagation, wird ein Verfahren vorgestellt, das diese Lernrate in der Lernphase modifiziert.

Lernrate

4.4.2 Delta-Regel

Eine weitere wichtige Lernregel ist die von Widrow und Hoff entwickelte Delta-Regel [WH60]. Sie ist für sogenannte einstufige FF-Netze geeignet. Ihr liegt folgende Überlegung zugrunde.

Solange eine Abweichung zwischen der Zielausgabe und Sollausgabe eines Netzes festgestellt wird, verändert man die internen Gewichte der Verbindung zwischen Ein- und Ausgabeschicht in definierter Weise, und zwar genau nach Formel (4.19). Übergeordnetes Ziel ist es, durch iterative Anwendung der Delta-Regel die Ist-Ausgabe des Netzes in Richtung der gewünschten Soll-Ausgabe zu verschieben.

Delta-Regel

$$\Delta w_{ij} = \eta \cdot o_j \cdot (\overline{o}_i - o_i) \tag{4.19}$$

Hierbei ist Δw_{ij} wieder die Veränderung des Verbindungsgewichtes von Neuron n_j zu Neuron n_i. Der Term $\delta_i = (\overline{o}_i - o_i)$ beschreibt die Differenz zwischen der Sollausgabe $\overline{o}_i$ und der Ist-Ausgabe o_i für das Neuron n_i, o_j ist die Ausgabe des Vorgängerneurons n_j. Wenn die Differenz δ_i gegen Null geht, d.h. Soll- und Ist-Ausgabe für das betreffende Neuron bereits nahe beieinander liegen, dann fällt die entsprechende Änderung Δw_{ij} für die Neuron n_j mit Neuron n_i verbindende Kante entsprechend klein aus.

4.4.3 Erweiterte Delta-Regel

Die einfache Delta-Regel hat den Nachteil, daß sie als Lernregel für mehrschichtige Neuronale Netze nicht ohne weiteres anwendbar ist, da man den Parameter δ_i für die inneren (verdeckten) Neuronen nicht unmittelbar bestimmen kann. Deshalb wurde eine erweiterte Delta-Regel entwickelt, die für mehrschichtige FF-Netze höherer Ordnung einsetzbar ist. Wie wir noch sehen werden, hat dies sehr weitreichende Konsequenzen im Hinblick auf die mit solchen Neuronalen Netzen lösbaren Problemklassen. Die erweiterte Delta-Regel ist definiert durch:

Erweiterte Delta-Regel

$$\Delta w_{ij} = \eta \cdot o_j \cdot \delta_i$$

$$\delta_i = \begin{cases} s'_F(\mathrm{net}_i)(\overline{o}_i - o_i) & \text{falls } n_i \text{ ein Ausgabeneuron ist} \\ s'_F(\mathrm{net}_i) \sum_k (\delta_k \cdot w_{ki}) & \text{sonst} \end{cases} \tag{4.20}$$

Der Index k läuft dabei über alle Nachfolgerneuronen des Neurons n_i, d.h. über alle Neuronen, die näher zur Outputschicht liegen. Die Funktion s'_F ist die erste Ableitung der in dem Neuron als Aktivierungsfunktion verwendeten Fermifunktion. Die ausführliche Herleitung der erweiterten Delta-Regel wird in Kap. 6 (KE 3) angegeben.

4.5 Die Arbeitsphasen Neuronaler Netze

In der Arbeit Neuronaler Netze unterscheidet man zwei verschiedene Phasen oder Modi:

Lernphase

- die *Lernphase* oder *Trainingsphase*, in der in Abhängigkeit vom Verhalten des Netzes zielgerichtet Parameterveränderungen vorgenommen werden,

Ausführungsphase

- die *Ausführungsphase* (Recall Mode), in der das Netz die eingelernten Leistungen (Klassifizieren, Assoziieren usw.) erbringt.

überwachtes Lernen

Es gibt zwei verschiedene Arten von Lernen. Beim *überwachten Lernen* (assoziativen Lernen) werden dem System bestimmte Vektoren von Eingabewerten $\mathbf{e} = \{e_1, \ldots, e_n\}$ und die zu diesen gehörenden, d.h. die gewünschten Ausgabedaten $\overline{\mathbf{o}} = \{\overline{o}_1, \overline{o}_2, \ldots, \overline{o}_m\}$ vorgegeben. Durch geeignete Änderung der Gewichte wird dann versucht, genau die gewünschte Input/Output-Relation ($\mathbf{e}$, $\overline{\mathbf{o}}$) mit dem Netz zu realisieren.

Beim *nicht-überwachtem Lernen* (entdeckenden Lernen) erhält das System nur die Eingabedaten und muß sich dann gewissermaßen von selbst einstellen.

Autoadaptation

Diesen Vorgang nennt man *Autoadaptation*.

Bei Netzen, die diese Art des Lernens zeigen, besteht die Leistung im Recall Mode darin, daß bei Eingaben, die sich ähneln (zur gleichen Klasse gehören), immer auch ähnliche Erregungsmuster der Neuronen ausgebildet werden. Bei diesen Netzen werden die in den Daten enthaltenen Ähnlichkeitsklassen erst im Laufe des Lernens entdeckt (entdeckendes Lernen).

4.6 Datenräume

Partitionierung des Datenraums

Stellt man sich die Ein- und Ausgaben bzw. Aktivierungen eines Neuronalen Netzes als Punkte in einem mehrdimensionalen Datenraum vor, so definiert ein entsprechend trainiertes Netz die Partitionierung dieses Datenraums in Teilräume. In den nachfolgenden Bildern ist jeweils die Aktivierung eines (einzigen) Outputneurons über beiden Eingaben in das Neuronale Netz aufgetragen, dies

ergibt eine dreidimensionale Darstellung: je eine Dimension für die beiden Inputs (die beiden waagrechten Achsen) und für den Output (senkrechte Achse).

Den einzelnen Partitionen des Datenraums entsprechen Klassen von Eingaben und jeder Klasse entspricht bei geeignetem Training des Netzes wiederum ein bestimmter Ausgabevektor. Komplexe Probleme zeichnen sich üblicherweise dadurch aus, daß die Klassengrenzen nichtlinearer Natur sind. Das trainierte Netz fungiert demnach als Klassifikator, der Eingabevektoren den entsprechenden Klassen zuordnet. Um das für eine bestimmte Anwendung adäquate Klassifikationsverhalten zu erzielen, müssen vom Netz möglicherweise sehr kompliziert geformte *Entscheidungsflächen* im Datenraum ausgebildet werden, die die Klassen separieren.

Klassifikation und Entscheidungsflächen

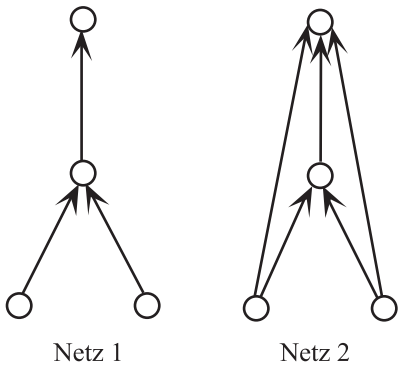


Abbildung 4.13: Beispielnetze

4.6.1 Einfluß der Aktivierungsfunktion auf die Entscheidungsfläche

Zum Abschluß soll veranschaulicht werden, welchen Einfluß die Aktivierungs- und Propagierungsfunktionen auf die Gestalt der Entscheidungsflächen haben. Für ein einfaches Netz (vgl. Abb. 4.13) werden unter Variation der Aktivierungsfunktion, der Propagierungsfunktion und gewisser Netzverbindungen die resultierenden Entscheidungsflächen für das Ausgabeneuron dargestellt.

Die Entscheidungsebene aus Abbildung 4.14 veranschaulicht nochmals graphisch die Ausführungen in Kapitel 4.2.2 zur Wirkung linearer Aktivierungsfunktionen. Dort wurde gezeigt, daß von diesen bei zwei Eingängen auch in mehrschichtigen Netzen nur eine zweidimensionale Entscheidungsebene erzeugt werden kann.

In Abbildung 4.15 wurde für Netz 1 die gewichtete Summe der Eingangsaktivitäten (Gleichung 4.1) als Propagierungsfunktion verwendet. Als Aktivierungsfunktion ist der einfache *tangens hyperbolicus* eingesetzt worden. Als Re-

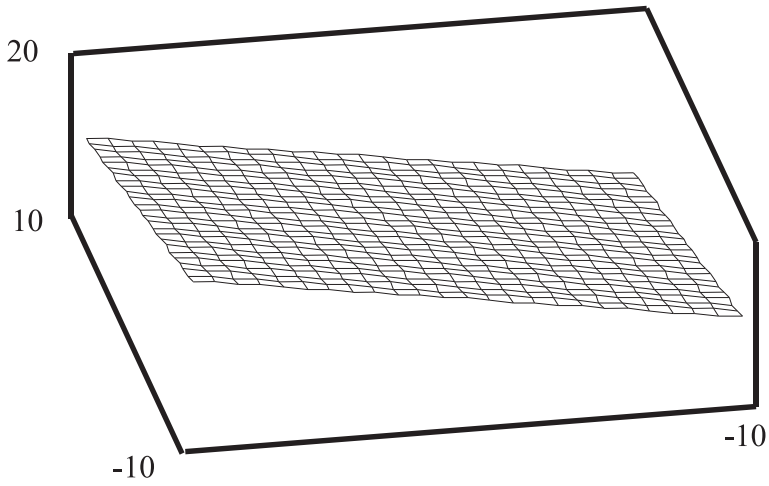


Abbildung 4.14: Verwendung linearer Aktivierungsfunktionen (Netz 1)

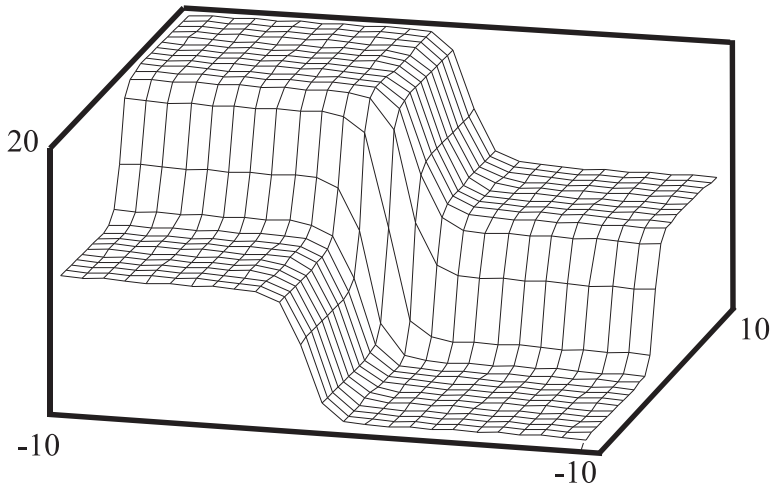


Abbildung 4.15: Verwendung des *tangens hyperbolicus* (Netz 1)

sultat erhält man eine im Raum gekrümmte Entscheidungsfläche. Diese erlaubt im Vergleich zu Abbildung 4.14 bereits eine komplexere Klassifikationsleistung.

In den Abbildungen 4.16 und 4.17 zeigt sich schließlich der Effekt des Einsatzes von Sigma-Pi-Units (vgl. Kap. 4.2.1, Formeln (4.1) bzw. (4.3)) und von Veränderungen der Netztopologie. Man erkennt, daß mit diesen Maßnahmen weitere Faltungen der Entscheidungsfläche erzielbar sind.

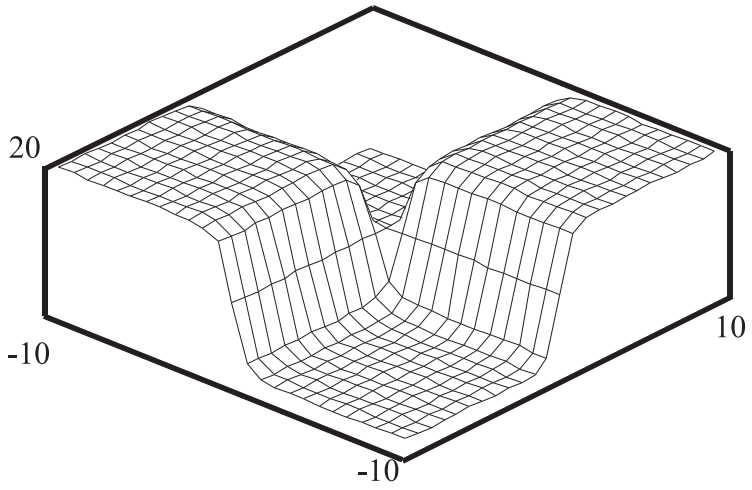


Abbildung 4.16: Verwendung des *tangens hyperbolicus* mit SigmaPi-Units (Netz 1)

4.6.2 Einfluß der verdeckten Neuronen auf die Entscheidungsfläche

Entsprechend den Ausführungen aus Kapitel 2 zum Thema Entscheidungsflächen kann ein Neuronales Netz als ein Funktionsapproximator angesehen werden, der den Verlauf einer gewünschten Entscheidungsfläche bzw. eine Näherung derselben berechnet. Dabei spielt die Konfiguration des Netzes eine entscheidende Rolle. Abbildung 4.18 zeigt eine exemplarische Trainingsmenge. Die Gesamtmenge folgt in ihrem Grundverlauf einer kubischen Funktion $f(x)$. Aufgrund leichter stochastischer Störungen zeigen jedoch die Einzelfälle mehr oder weniger starke Abweichungen von dieser kubischen Grundfunktion.

Angenommen, man würde die den Trainingsdaten zugrundeliegende Funktion genau kennen, dann stellte natürlich ein Polynom dritten Grades die beste Approximation dar. Durch geeignete Wahl der Koeffizienten kann dann eine optimale Approximation der Grundfunktion und der verrauschten Beispieldaten bei minimaler mittlerer Abweichung (s. Abb. 4.20) erreicht werden. Diese Funktion würde die Trainingsdaten am besten generalisieren.

Da die Grundfunktion nicht bekannt ist, kann man versuchen, diese sukzessive durch Polynome unterschiedlichen Grades anzunähern. Beginnt man die Daten durch eine quadratische Funktion zu approximieren, so stellt diese nur eine schlechte Annäherung an die tatsächlichen Gegebenheiten dar (s. Abb. 4.19).

Wählt man andererseits ein Polynom höheren als dritten Grades, so optimiert man immer mehr den Reproduktionsfehler auf der Trainingsmenge, verliert aber gleichzeitig den funktionellen Zusammenhang. Abbildung 4.21

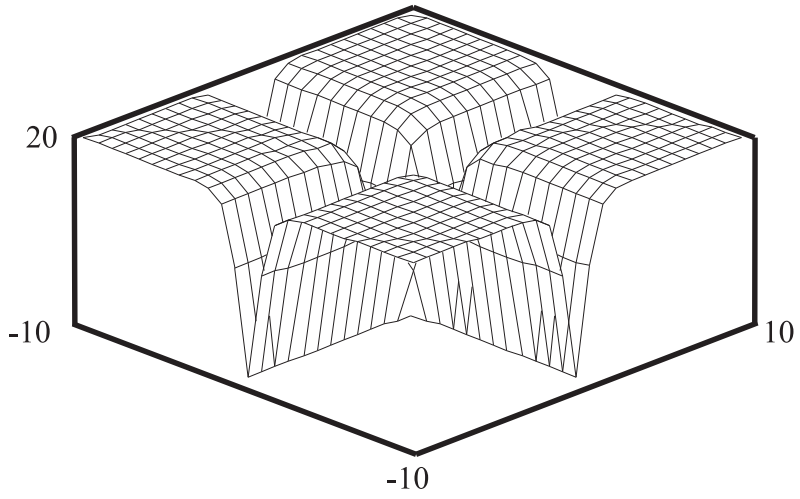


Abbildung 4.17: Verwendung des *tangens hyperbolicus* mit SigmaPi-Units und veränderter Netztopologie (Netz 2)

zeigt dies besonders deutlich. Hier wird letztlich das stochastische Rauschen im Detail beschrieben, und die Oszillationseffekte bestimmen den Funktionsverlauf zwischen den Datenpunkten. Der gewünschte Generalisierungseffekt würde aber i.a. verlorengehen, da die so berechnete Funktion zwar die Trainingsdaten "übergenau" wiedergibt, aber im weiteren Verlauf ein ganz anderes Verhalten zeigt als die intendierte Grundfunktion (sogenanntes "*Overfitting*").

Overfitting

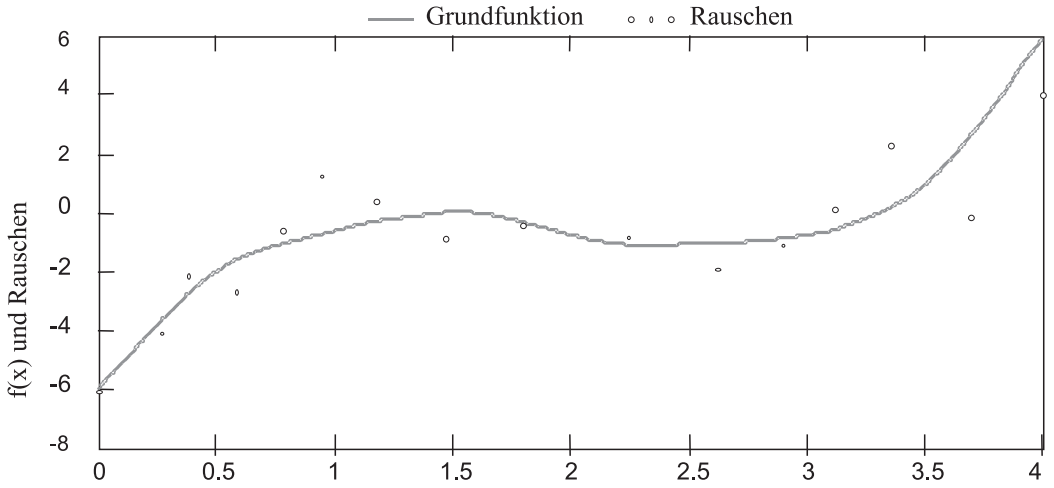


Abbildung 4.18: Eine kubische, stochastisch gestörte Funktion f (nach [Was93])

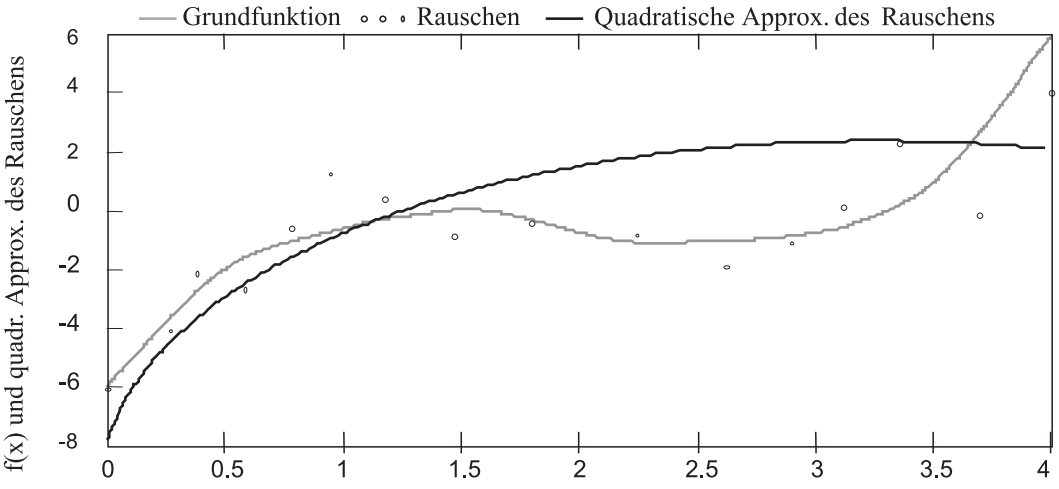


Abbildung 4.19: Eine quadratische Approximation zu f (nach [Was93])

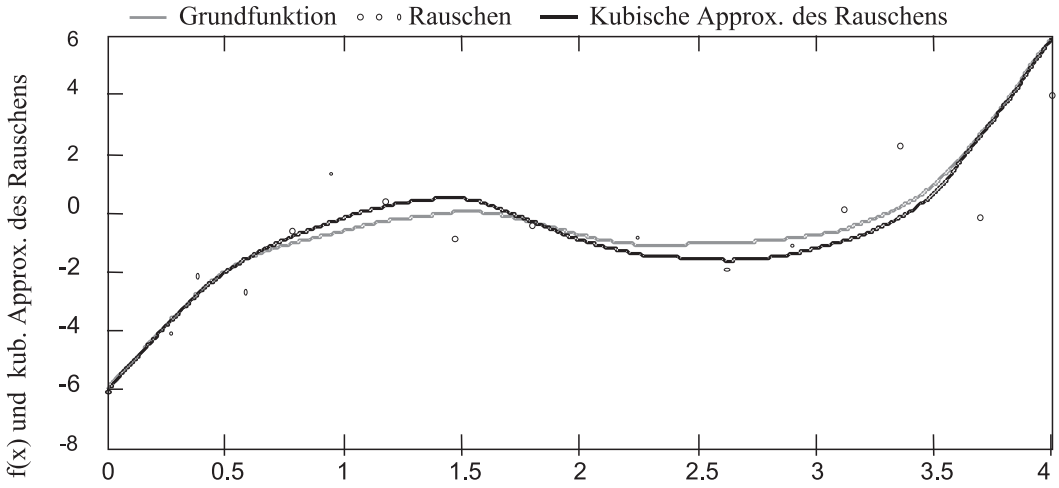


Abbildung 4.20: Eine kubische Approximation zu f (nach [Was93])

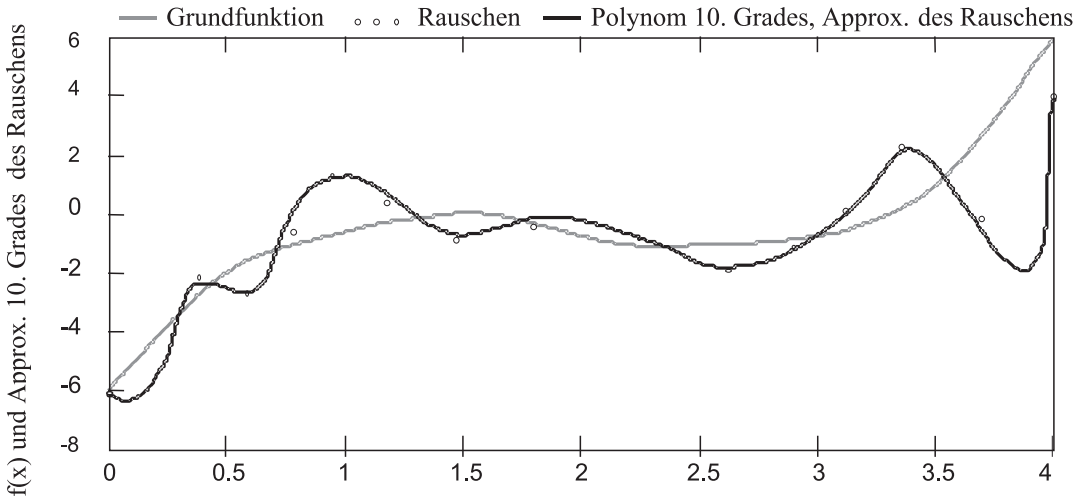


Abbildung 4.21: Eine Approximation zehnten Grades zu f (nach [Was93])