Universität Hamburg

Fachbereich Wirtschaftswissenschaften

Institut für Wirtschaftsinformatik

Hausarbeit zum Thema 10

Grundlagen und Funktionsweise von Künstlichen Neuronalen Netzen

Prof. Dr. D. B. Preßmar

Seminar zur betrieblichen Datenverarbeitung

SS 2002

Computergestützte Systeme für die betriebliche Planung

Betreuer: Sven F. Crone

Tobias Baumgärtel

Am Sood 40
22848 Norderstedt
Telefon: 0 40 / 5 50 85 40

Matr.Nr.: 134 03 64

Fachrichtung:
Wirtschaftsingenieurwesen

Fachsemester: 20

Datum: 19.4.2002
Abgabetermin: 22.4.2002

Inhaltsverzeichnis

1 Einleitung **3**

2 Biologischer Hintergrund **3**

3 Bestandteile Künstlicher Neuronaler Netze **4**

3.1 Definition . . . 4
3.2 Das künstliche Neuron . . . 5
3.2.1 Bestandteile . . . 5
3.2.2 Eingangsfunktion . . . 5
3.2.3 Transferfunktion . . . 6
3.3 Vernetzung/Topologien . . . 6
3.3.1 Schichten . . . 6
3.3.2 Rückkopplung . . . 7
3.4 Lernen . . . 8
3.4.1 Allgemein . . . 8
3.4.2 Überwachtes Lernen . . . 10
3.4.2.1 Hebb'sche Lernregel . . . 10
3.4.2.2 Delta- oder Widrow-Hoff-Lernregel . . . 10
3.4.2.3 Backpropagation-Lernregel . . . 11
3.4.3 Unüberwachtes Lernen . . . 12

4 Populäre Netzwertopologien **12**

4.1 Allgemein . . . 12
4.2 Adaline . . . 13
4.3 Perzeptron . . . 13
4.4 Madaline . . . 13
4.5 Multi-Layer-Perzeptron . . . 13

5 Beispiel: Das XOR-Problem **15**

Abbildungsverzeichnis

1 Nervenzelle . . . 3
2 Neuronales Netz mit zwei verdeckten Schichten ohne Rückkopplung . . . 4
3 Schema eines künstlichen Neurons . . . 5
4 Schwellenwertfunktion, Sigmoide Funktionen . . . 7
5 Topologie des Adaline . . . 13
6 Rechenbeispiel: Einschichtiges Netz . . . 15
7 Rechenbeispiel: Zweischichtiges Netz . . . 16

Tabellenverzeichnis

1 XOR-Funktion . . . 15
2 Berechnung im einschichtigen Netz . . . 15
3 Berechnung im zweischichtigen Netz . . . 16

Abkürzungsverzeichnis

SOM	Self Organizing Maps

1 Einleitung

Neuronale Netze finden ihren Einsatz bei Problemen, die weder algorithmisch exakt oder mit ökonomischem Aufwand berechnet werden können noch chaotischer Natur sind. Ihre Stärken liegen in ihrer Generalisierungsfähigkeit und Fehlertoleranz.[1]

Sie spielen heute eine wichtige Rolle in bestimmten betriebswirtschaftlichen Planungs- und Entscheidungsprozessen, wie z.B. Kreditprüfung, Immobilienanalyse, Aktienkursprognose, Abschätzung der Insolvenzwahrscheinlichkeit von Unternehmen uvm.[2]

In dieser Arbeit werden die Grundlagen und Mechanismen erläutert, die allen gebräuchlichen neuronalen Netzen gemein sind.

2 Biologischer Hintergrund

Künstliche neuronale Netze können als technische Umsetzung der Gehirnfunktionen verstanden werden.[3] Neuronale Systeme sind Systeme aus zu Netzwerken verknüpften Nervenzellen (Neuronen). Künstliche Neuronen sollen die Funktionen biologischer Nervenzellen modellieren. [4]

Eine Nervenzelle (s. Abb. 1) besteht im wesentlichen aus den kurzen stark verästelten Dendriten, dem Zellkörper und einer (beim Menschen über einen Meter langen) Nervenfaser, dem Neurit oder Axon, welches durch hemmende oder erregende Synapsen andere Nervenzellen beeinflußt. An einer Nervenzelle enden viele Axome fremder Neurone und wird eine größere Anzahl der erregenden Synapsen gleichzeitig aktiviert, summieren sich ihre Wirkungen. Wird ein bestimmter Schwellenwert erreicht, feuert das Neuron seinerseits ein Aktionspotential ab und erregt oder hemmt damit andere Nervenzellen.

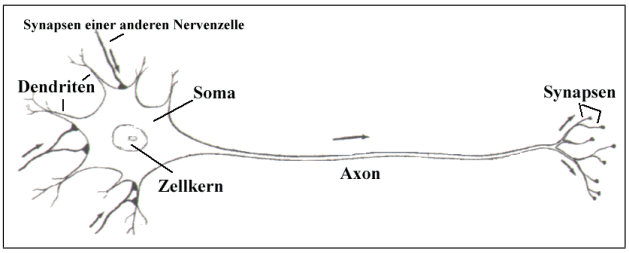

Abbildung 1: Nervenzelle

Quelle: Online unter

http://www.uni-ulm.de/ s_ykrist/Arbeit/gradient/nernenzelle.gif

Verbindet man nun mehrere Neuronen miteinander, so erhält man ein neuronales Netz.[5]

[1] Vgl. Kinnebrock, Werner: Neuronale Netze, Grundlagen, Anwendungen, Beispiele, 2. Aufl., München 1994, S. 10

[2] Vgl. Mertens, Peter: Lexikon der Wirtschaftsinformatik, 3. Aufl., Berlin 1997, S. 281

[3] Hoffmann, Norbert: Neuronale Netze, Anwendungsorientiertes Wissen zum Lernen und Nachschlagen, Braunschweig 1993, S. 1

[4] Klein, Thorsten, Diplomarbeit: Untersuchungen zur Struktur- Eigenschaftsbeziehung reiner Stoffe nach inkrementellen und nicht-inkrementellen Verfahren, Universität-Gesamthochschule Paderborn, Online im Internet: http://ac16.uni-paderborn.de/arbeitsgebiete/diplomarbeiten/thorsten_klein/diplom.html, Abruf 27.3.2002

[5] Vgl. Linder, Hermann: Biologie, Lehrbuch für die Oberstufe, 20. Aufl., Stuttgart 1989, S. 206 ff.

3 Bestandteile Künstlicher Neuronaler Netze

3.1 Definition

Ein neuronales Netz ist ein Paar (N, V) mit einer Menge N Neuronen und einer Menge V von Verbindungen. Es besitzt die Struktur eines gerichteten Graphen, für den die folgenden Einschränkungen und Zusätze gelten:

i. Die Knoten des Graphen heißen Neuronen.

ii. Die Kanten heißen Verbindungen.

iii. Jedes Neuron kann eine beliebige Menge von Verbindungen empfangen, über die es seine Eingabe erhält.

iv. Jedes Neuron kann genau eine Ausgabe über eine beliebige Menge von Verbindungen aussenden.

v. Das Neuronale Netz erhält aus Verbindungen, die der „Außenwelt“ entspringen Eingaben und gibt seine Ausgabenüber in der Außenwelt endende Verbindungen ab.[6]

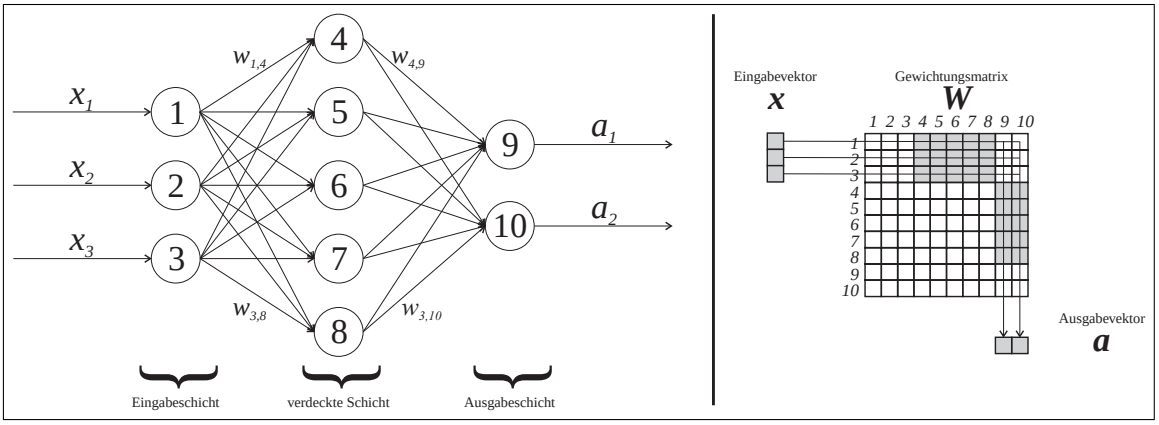

Abbildung 2: Neuronales Netz mit zwei verdeckten Schichten ohne Rückkopplung
(Entworfen und gezeichnet: Verfasser)

Alle Verbindungen, die von anderen Neuronen zu einem einzelnen Neuron j gehen, ergeben den Eingabevektor $\boldsymbol{x}_j$ von j. Da bei den meisten neuronalen Netzen die Eingabe gewichtet wird, kann man die Verbindungsstruktur in Form einer Matrix beschreiben (s. Abb. 2 rechts). Zeilen und Spalten identifiziert man mit den Neuronen. In den Kreuzungspunkt schreibt man das Gewicht der Verbindung. In der Schreibweise

$$\boldsymbol{W} = [w_{i,j}] \tag{1}$$

[6]Vgl. Neuro-Fuzzy-AG, Prof. Dr. Wolfram M. Lippe, Institut für Informatik, Westfälische Wilhelms Universität - Münster: Einführung in Neuronale Netze, Online im Internet: http://wwwmath.uni-muenster.de/SoftComputing/lehre/material/wwwnnscript, Stand 16.1.2001, Abruf 27.3.2002

gilt dann:

$$w_{i,j} \begin{cases} = 0 & \text{keine Verbindung von Neuron } i \text{ zu Neuron } j \\ < 0 & \text{hemmende Verbindung der Stärke } |w_{i,j}| \\ > 0 & \text{anregende Verbindung der Stärke } |w_{i,j}| \end{cases} \tag{2}$$

3.2 Das künstliche Neuron

3.2.1 Bestandteile

Das erst Modell eines Neurons wurde 1943 von W. S. McCulloch und W. Pitts entworfen.[7] Ein künstliches Neuron besteht im wesentlichen aus folgenden Bestandteilen:

- Eingangssignale x_i
- Gewichte $w_{i,j}$[8]
- Eingangsfunktion $\varepsilon(\boldsymbol{e})$
- Nettosignal net
- Transferfunktion $a(net)$
- Ausgangssignal $\boldsymbol{a}$[9]

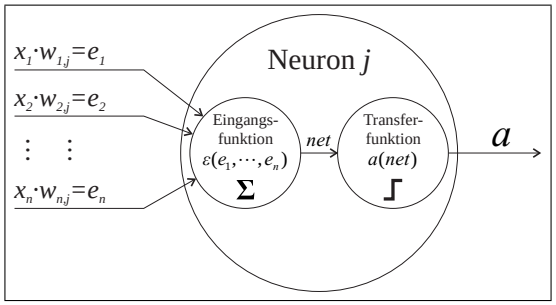

Abbildung 3: Schema eines künstlichen Neurons
(Entworfen und gezeichnet: Verfasser)

3.2.2 Eingangsfunktion

Die Eingangssignale x_i werden mit den Gewichten $w_{i,j}$ multipliziert und stellen so den Eingangsvektor $\boldsymbol{e}$ dar. Als Eingangsfunktion bezeichnet man die Funktion, die den Eingangsvektor $\boldsymbol{e}$ auf einen eindimensionalen Wert net, dem Nettosignal abbildet, der der Transferfunktion $a(net)$ (s. Absch. 3.2.3) als Eingabe dient. Oft wird hier das Skalarprodukt aus Eingangssignalvektor $\boldsymbol{x}$ und Gewichtungsmatrix $\boldsymbol{W}$ verwendet (s. Gl. 3).

[7] Vgl. Neuro-Fuzzy-AG, Prof. Dr. Wolfram M. Lippe, Institut für Informatik, Westfälische Wilhelms Universität - Münster: Einführung in Neuronale Netze, Online im Internet: http://wwwmath.uni-muenster.de/SoftComputing/lehre/material/wwwnnscript, Stand 16.1.2001, Abruf 27.3.2002

[8] Vgl. Reif, Gerald, Diplomarbeit: Moderne Aspekte des Wissensverarbeitung, ein interaktiver Lernbehelf für das Web Based Training, Technischen Universität Graz, Online im Internet: http://www.iicm.edu/greif/thesis.html, Stand 20.1.2000, Abruf 17.4.2002

[9] Vgl. Hoffmann, Norbert: Neuronale Netze, Anwendungsorientiertes Wissen zum Lernen und Nachschlagen, Braunschweig 1993, S. 29

$$\begin{aligned} net &= \varepsilon\left([e_i]\right) \\ &= \varepsilon\left(\boldsymbol{e}\right) \\ &= \varepsilon\left(\boldsymbol{x}, \boldsymbol{W}\right) \\ \text{z.B.} \qquad &= \boldsymbol{x} \cdot \boldsymbol{W} \\ &= \sum_{i=1}^{n} w_{i,j} \cdot e_i \end{aligned} \tag{3}$$

3.2.3 Transferfunktion

Die Transferfunktion besteht aus zwei Funktionen, der Aktivierungsfunktion und der Ausgangsfuntion. Die Aktivierungsfunktion berechnet die Aktivität eines Neurons aus dem Eingangsvektor und der Aktivität des Neurons vom vorherigen Zeitpunkt:[10]

$$c(t) = c\left(\varepsilon\left(t\right), c\left(t-1\right)\right) \tag{4}$$

Diese Funktion modelliert eine Eigenschaft der natürlichen Nervenzelle, die bei den meisten künstlichen Neuronentypen nicht verwendet wird, d.h. die Aktivität ist identisch mit dem Eingangsvektor ($\boldsymbol{c} = \varepsilon(\boldsymbol{e})$, man spricht hier von der Identität).

Die Übertragungsfunktion besteht somit meistens nur aus der Ausgangsfunktion, welche häufig den Netto-Input auf einen Werteberich beschränkt (z.B. das Intervall $[0; 1]$ oder den Wertebereich$\{0; 1\}$).[11]

Beispiele hierfür sind die Schwellenwertfunktion und die sigmoiden Funktionen (s. Abb. 4). Der Vorteil einer sigmoiden Funktion ist, daß sie ein differenzierbares Ausgangssignal liefert, wodurch einige Lernverfahren wie z.B. das backpropagation-Verfahren (s. Absch. 3.4.2.3) erst möglich sind.

3.3 Vernetzung/Topologien

3.3.1 Schichten

Ein Netz zerfällt meist in mehrere Teile, die man Schichten (auch Gruppen oder Lagen) nennt. Für die Einteilung in Schichten gibt es keine Regeln, für gewöhnlich werden Neuronen zu einer Schicht zusammengefaßt, die eine bestimmte Aufgabe erfüllen.[12]

Einschichtige Netzwerke sind solche, die nur aus einer Neuronenschicht bestehen und bei denen einige Neuronen der Eingangsvektor aufnehmen und andere den Ausgangsvektor ausgeben. Beispiel für solches System ist das Hopfield-Modell.[13]

[10] Vgl. Kratzer, Klaus Peter: Neuronale Netze, Grundlagen und Anwendungen, 2. Aufl., München 1993, S. 25

[11] Vgl. Hoffmann, Norbert: Neuronale Netze, Anwendungsorientiertes Wissen zum Lernen und Nachschlagen, Braunschweig 1993, S. 29

[12] Vgl. Hoffmann, Norbert: Neuronale Netze, Anwendungsorientiertes Wissen zum Lernen und Nachschlagen, Braunschweig 1993, S. 39

[13] Vgl. Grauel, Adolf: Neuronale Netze, Grundlagen und mathematische Modellierung, Mannheim 1992, S. 33

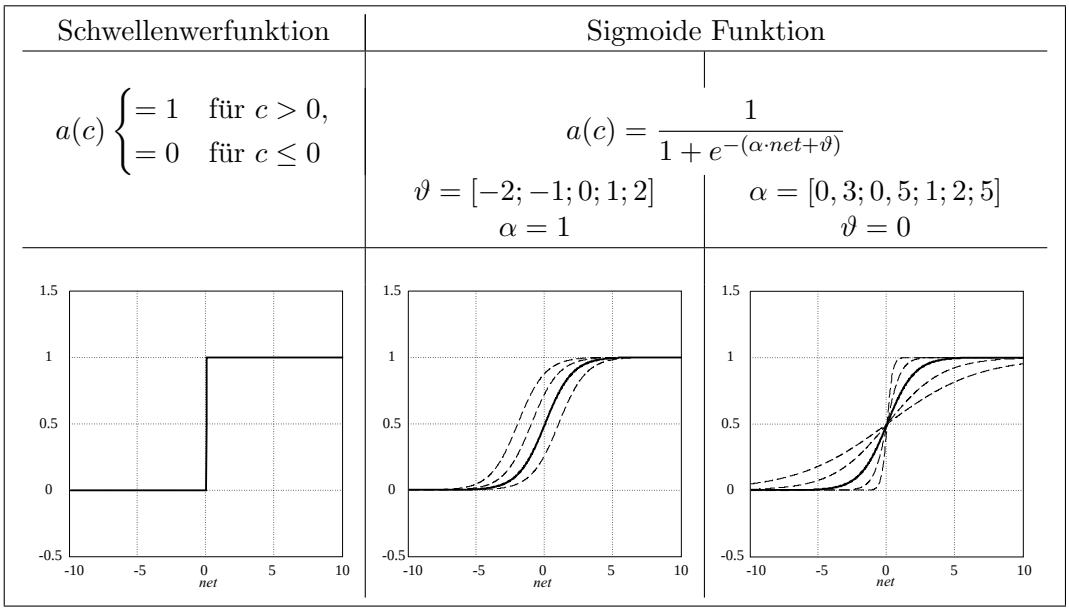

Abbildung 4: Schwellenwertfunktion, Sigmoide Funktionen

Gezeichnet: Verfasser,

Vgl. Reif, Gerald, Diplomarbeit: Moderne Aspekte des Wissensverarbeitung, ein interaktiver Lernbehelf für das Web Based Training, Technischen Universität Graz, Online im Internet: http://www.iicm.edu/greif/thesis.html, Stand 20.1.2000, Abruf 17.4.2002

Leistungsfähiger sind allerdings mehrschichtige Modelle (die Bool'sche XOR-Funktion läßt sich z.B. nicht mit einem einschichtigen Netz abbilden, s. Abb. 5). Neuronen, die von außen zugänglich sind heißen sichtbare Neuronen (Eingabe- und Ausgabeschicht, s. Abb. 2), alle anderen nennt man verborgene Neuronen (in Abb. 2 die verdeckte Schicht).[14] Die Neuronen der Eingabeschicht dienen nur dazu, die einzelnen Eingabedaten auf sämtliche Neuronen der darunterliegenden Schicht zu verteilen, wobei in den Neuronen unterschiedliche Werte ankommen, da ja jede Verbindung gewichtet ist.[15]

3.3.2 Rückkopplung

Neuronale Netze lassen hinsichtlich des Datenflusses in Netze ohne Rückkopplung (*feedforward*-Netze) und Netze mit Rückkopplung (*rekurrente* Netze) einteilen.

Feedforward-Netze können entweder schichtenweise verbunden sein (sog. ebenenweise verbundenen feedforward-Netze), wobei es nur Verbindungen von einer Schicht zu nächsten gibt (vollständig verbunden under nur teilweise), oder es kann Verbindungen (sog. shortcut connections) geben, die Schichten überspringen. Hier spricht man dann von allgemeinen feedforward-Netzen.

In rekurrenten Netzen kann es verschiedene Arten von Rückkoplungen geben:

[14] Vgl. Hoffmann, Norbert: Neuronale Netze, Anwendungsorientiertes Wissen zum Lernen und Nachschlagen, Braunschweig 1993, S.39

[15] Vgl. Klein, Thorsten, Diplomarbeit: Untersuchungen zur Struktur- Eigenschaftsbeziehung reiner Stoffe nach inkrementellen und nicht-inkrementellen Verfahren, Universität-Gesamthochschule Paderborn, Online im Internet: http://ac16.uni-paderborn.de/arbeitsgebiete/diplomarbeiten/thorsten_klein/diplom.html, Abruf 27.3.2002

Netze mit direkten Rückkopplungen (*direct feedback*): Die Neuronen haben eine Verbindung von ihrer Ausgabe zurück zu ihrer Eingabe und beeinflussen sich somit selbst.

Netze mit indirekten Rückkopplungen (*indirect feedback*): Hier führen die Rückkopplungen von Neuronen höherer Schichten zu Neuronen niederer Schichten. Hierbei erreicht man eine Aufmerksamkeitssteuerung auf bestimmte Bereiche von Eingabeneuronen oder auf bestimmte Eingabemerkmale.

Netze mit Rückkopplungen innerhalb einer Schicht (*lateral feedback*): Netze mit Rückkopplungen innerhalb derselben Schicht werden oft für Aufgaben eingesetzt, bei denen nur ein Neuron einer Gruppe aktiv werden soll. Jedes Neuron hat dann hemmende (also negativ gewichtete) Verbindungen zu den anderen Neuronen und eine aktivierende direkte Rückkopplung zu sich selbst. Das Neuron mit der stärksten Aktivierung, der Gewinner, hemmt dann die anderen Neuronen. Eine solche Topologie nennt man daher auch *winner-takes-all*-Netzwerk.

vollständig verbundene Netze: Diese Netze haben Verbindungen zwischen allen Neuronen. Sie sind insbesondere als Hopfield-Netze bekannt.

Netze mit Rückkopplungen werden auch eingesetzt um Zeitabhängigkeiten bei Daten, wie z.B. der Struktur einer Schwingung modellieren zu können.[16]

3.4 Lernen

3.4.1 Allgemein

Zu einem künstlichen neuronalen Netzwerk gehören auch Methoden, die die Parameter (insbesondere die Gewichte) vor dem Einsatz (der Reproduktionsphase) eingestellen. Dieser Vorgang wird als Lernen bezeichnet.[17]

Soll das Netz die während der Ablaufphase verrauschten oder verzerrten Eingaben wieder entzerrt „erkennen“ und ausgeben (z.B. Gesichtserkennung), so ist das Lernziel autoassoziativ. Soll das Netz irgendeinen Zusammenhang erlernen, die Eingaben also völlig anders als die Ausgaben sind, ist das Lernziel heteroassoziativ.[18]

Grundsätzlich kann man zwei Arten des Lernen unterscheiden, nämlich Lernen der Struktur und Lernen der Parameter. Gewöhnlich werden Anpassungen der Struktur nicht in den Lernvorgang einbezogen, da diese meist unveränderlich vorgegeben ist. Üblicherweise werden beim Lernen nur die Gewichte der Verbindungen angepaßt.[19] Mit der Wahl von Werten für die Verbindungsgewichte können

[16] Vgl. Neuro-Fuzzy-AG, Prof. Dr. Wolfram M. Lippe, Institut für Informatik, Westfälische Wilhelms Universität - Münster: Einführung in Neuronale Netze, Online im Internet: http://wwwmath.uni-muenster.de/SoftComputing/lehre/material/wwwnnscript, Stand 16.1.2001, Abruf 27.3.2002

[17] Vgl. Hoffmann, Norbert: Neuronale Netze, Anwendungsorientiertes Wissen zum Lernen und Nachschlagen, Braunschweig 1993, S. 57

[18] Vgl. Klein, Thorsten, Diplomarbeit: Untersuchungen zur Struktur- Eigenschaftsbeziehung reiner Stoffe nach inkrementellen und nicht-inkrementellen Verfahren, Universität-Gesamthochschule Paderborn, Online im Internet: http://ac16.uni-paderborn.de/arbeitsgebiete/diplomarbeiten/thorsten_klein/diplom.html, Abruf 27.3.2002

[19] Vgl. Hoffmann, Norbert: Neuronale Netze, Anwendungsorientiertes Wissen zum Lernen und Nachschlagen, Braunschweig 1993, S. 57

existierende Verbindungen zerstört ($w_{ij=0}$) und in vorhergehenden Schritten zerstörte Bindungen wieder funktionsfähig ($w_{ij} \neq 0$) gemacht werden. $w_{ij} > 0$ kennzeichnet eine erregende Verbindung und mit $w_{ij} < 0$ wird eine hemmende Wirkung gekennzeichnet.[20]

Man unterscheidet drei Lernverfahren:

Lernen mit Unterweisung (*supervised learning*): Bei jedem Lernschritt wird der vom Netz gelieferte Outputvektor ***o*** mit dem Zielvektor ***a*** verglichen. Die Gewichte werden nach einer Lernformel verändert, falls ***o*** ungleich ***a*** ist. Dieses Verfahren bezeichnet man auch als überwachtes Lernen.

Lernen ohne Unterweisung(*unsupervised Learning*): Hier werden die Gewichte durch Zufallszahlen leicht verändert und durch eine Bewertungsformel stellt man fest, ob der mit den veränderten Gewichten berechnete Output besser ist als der alte. In diesem Fall werden die veränderten Gewichte gespeichert, andernfalls vergessen. Diese Art des Lernens bezeichnet man auch als Bestärkendes Lernen (*reinforcement learning*).

Lernen durch Selbstorganisation(*self-organized learning*): Die Gewichte ändern sich bei jedem Lernschritt. Die Änderung ist abhängig von

1. den Nachbarschaften der Eingabemuster,
2. der Wahrscheinlichkeitsverteilung, mit der die zulässigen Eingabemuster beim Lernen angeboten werden.[21]

Das Netz versucht also ohne Beeinflussung von außen die präsentierten Daten in Ähnlichkeitsklassen aufzuteilen.[22]

Allgemein kann man die Änderung der Gewichte beim Lernen folgendermaßen schreiben:[23]

$$w_{ij}^{\text{neu}} = w_{ij}^{\text{alt}} + \Delta w_{ij} \tag{5}$$

Man unterscheidet auch noch nach den Zeitpunkten des Lernens. Wird das Netz erst trainiert und später eingesetzt, spricht man von einem *offline*-Lernverfahren. Lernt das Netz erst während seines Einsatzes, handelt es sich um ein *online*-Lernverfahren.

Eine Schwierigkeit der Lernalgorithmen ist die richtige Wahl der Parameter des jeweiligen Verfahrens, z.B. der Lernrate. Bei ungeeigneten Parametern konvergieren die Gewichte nicht sonder divergieren oder oszillieren.[24]

Die folgenden Lernregeln gelten nur für feedforward-Netze.

[20] Vgl. Grauel, Adolf: Neuronale Netze, Grundlagen und mathematische Modellierung, Mannheim 1992, S.35

[21] Vgl. Kinnebrock, Werner: Neuronale Netze, Grundlagen, Anwendungen, Beispiele, 2. Aufl., München 1994, S. 31,32

[22] Neuro-Fuzzy-AG, Prof. Dr. Wolfram M. Lippe, Institut für Informatik, Westfälische Wilhelms Universität - Münster: Einführung in Neuronale Netze, Online im Internet: http://wwwmath.uni-muenster.de/SoftComputing/lehre/material/wwwnnscript, Stand 16.1.2001, Abruf 27.3.2002

[23] Kinnebrock, Werner: Neuronale Netze, Grundlagen, Anwendungen, Beispiele, 2. Aufl., München 1994

[24] Vgl. Kinnebrock, Werner: Neuronale Netze, Grundlagen, Anwendungen, Beispiele, 2. Aufl., München 1994, S. 34

3.4.2 Überwachtes Lernen

3.4.2.1 Hebb'sche Lernregel Ausgangspunkt dieser Regel war eine Vermutung des Psychologen Donald Hebb 1949 über die Veränderung von Synapsen in Nervensystemen, welche besagt, daß wenn zwei verbundene Nervenzellen gleichzeitig feuern, ihre Verbindung stärker wird. Diese Regel läßt sich nur auf einschichtige Netze anwenden[25] und wird meist bei binären Aktivierungsfunktionen benutzt, wobei oft -1 und 1 als Aktivierungen verwendet werden, um auch eine Verminderung der Beträge der Gewichte zu ermöglichen. [26]

$$\Delta w_{ij} = \eta S_i E_j \tag{6}$$

S_i sind die Sollwerte (daher die Einschränkung auf einschichtige Netze, da diese Werte ja nur für den Ausgang des Netzes bestimmt sind) und E_i die Eingangswerte. Der Faktor η heißt Lernrate. Die Formel (6) gilt für ein einzelnes Musterpaar, das einmal gelernt wird. Initialisiert man die Gewichte vor dem Lernvorgang mit

$$w_{ij} = 0 \tag{7}$$

und lernt jedes Muster einmal, so kann man die einzelnen Gewichtsänderungen Δw_{ij} ausummieren und erhält:

$$w_{ij} = \eta \sum_{\mu=1}^{p} S_i^\mu E_j^\mu \tag{8}$$

Man sieht, daß hier kein Vergleich zwischen Soll- und Ist-Werten beim Lernen erfolgt. Dieser Nachteil schränkt die Verwendung Hebb'schen Lernregel zusätzlich ein.[27]

3.4.2.2 Delta- oder Widrow-Hoff-Lernregel Diese Lernregel ist eine Erweiterung der Hebb'schen Lernregel, die die Ausgabe des Netzes bei der Änderung der Gewichte berücksichtigt.[28] Damit ist die Gewichtsveränderung Δw_{ij} proportional zur Differenz von Soll- und Ist-Ausgabe $(S_i - A_i)$.[29]

$$\Delta w_{ij} = \eta \left(S_i - A_i\right) E_j \tag{9}$$

Analog zur Hebb'schen Lernregel ist die Delta-Lernregel nur auf einschichtige Netze anwendbar.[30]

[25] Vgl. Hoffmann, Norbert: Neuronale Netze, Anwendungsorientiertes Wissen zum Lernen und Nachschlagen, Braunschweig 1993, S. 58 ff.

[26] Vgl. Lampel, Johannes: Einsatz von neuronalen Netzen in einem Bot für Counterstrike, Online im Internet: http://www.joebot.net/lampel/johannes/bll137.html, Stand 16.11.2001, Abruf 19.4.2002, S. 18

[27] Vgl. Hoffmann, Norbert: Neuronale Netze, Anwendungsorientiertes Wissen zum Lernen und Nachschlagen, Braunschweig 1993, S. 59,60

[28] Vgl. Hoffmann, Norbert: Neuronale Netze, Anwendungsorientiertes Wissen zum Lernen und Nachschlagen, Braunschweig 1993, S.60

[29] Vgl. Lampel, Johannes: Einsatz von neuronalen Netzen in einem Bot für Counterstrike, Online im Internet: http://www.joebot.net/lampel/johannes/bll137.html, Stand 16.11.2001, Abruf 19.4.2002, S. 18

[30] Vgl. Hoffmann, Norbert: Neuronale Netze, Anwendungsorientiertes Wissen zum Lernen und Nachschlagen, Braunschweig 1993, S.60

3.4.2.3 Backpropagation-Lernregel Die Backpropagation-Lernregel ist eine Verallgemeinerung der Delta-Regel für mehrstufige Netze. Hierbei wird der Fehler, der in der Ausgabeschicht direkt berechnet werden kann, zurückpropagiert, d.h. es werden alle Fehler der Neuronen, mit denen das betreffende Neuron in einer höheren Schicht verbunden ist, unter Beachtung der Gewichte der entsprechenden Verbindungen, addiert. Da die Aktivierungsfunktion bei der Vorwärtspropagierung und somit auch bei der Rückwärtspropagierung eine Rolle spielt, muss auch sie mit in die Rechnung einbezogen werden.

Backpropagation ist ein sogenanntes Gradientenabstiegsverfahren. Wenn man sich die Fehlerfläche eines neuronalen Netzes bildlich vorstellt, so sucht dieses Verfahren stets den kürzesten Weg ins Tal. Die Fehlerfläche eines Netzes ist aber nicht dreidimensional, sondern hat so viele Dimensionen wie es Gewichte gibt.[31]

Es besteht hierbei immer die Gefahr, auf einer suboptimalen Ebene oder in einem lokalen Minimum „hängen“ zu bleiben.

Die Gesamtfehlerfunktion, die minimiert werden soll ist wie folgt definiert (E steht hier für Error, nicht wie oben für Eingangswerte):

$$E = \sum_p E_p \tag{10}$$

Hierbei ist p der Index für die einzelnen Trainingsmuster. Für die einzelnen Muster lautet sie:

$$E_p = \frac{1}{2} \sum_j (S_{pj} - A_{pj})^2 \tag{11}$$

Da nur die Gewichte in die Berechnung von A_{ij} eingehen, ist der Funktionswert von E nur noch von den Gewichten abhängig (das $\frac{1}{2}$ und das ^{2} vereinfacht nur das spätere Differenzieren). Da man das Minimum finden will, muß man die Gewichte entgegen dem Gradienten verändern:

$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}} \tag{12}$$

Auch hier ist η wieder die Lernrate, die angibt, wie schnell die Gewichte verändert werden sollen.[32]

[31] Vgl. Lampel, Johannes: Einsatz von neuronalen Netzen in einem Bot für Counterstrike, Online im Internet: http://www.joebot.net/lampel/johannes/bll137.html, Stand 16.11.2001, Abruf 19.4.2002, S. 19

[32] Vgl. Lampel, Johannes: Einsatz von neuronalen Netzen in einem Bot für Counterstrike, Online im Internet: http://www.joebot.net/lampel/johannes/bll137.html, Stand 16.11.2001, Abruf 19.4.2002, S. 19 ff.

Als Ergebnis der Ableitungen erhält man die Backpropagation-Lernregel:

$$\Delta w_{ij} = \eta \sum_{p} \delta_{pi} e_{pi} \tag{13}$$

Dabei ist e_{pi} der Wert vom Eingabemuster p am Eingang des Neurons i und δ_{pi} das Fehlermaß, daß beim Eingabemuster p am Neuron i entsteht. Der Lernfehler wird durch die einzelnen Schichten des Netzes zurückgeführt. Daher der Name *Backpropagation.*[33]

3.4.3 Unüberwachtes Lernen

Die bekanntesten Netze, die nach einem unüberwachten Lernverfahren arbeiten, sind die Selbstorganisierenden Karten (Self Organizing Maps). Bei SOM beeinflussen aktive Neuronen ihre räumlich nahen Nachbarneuronen und bewirken damit, daß sich ähnliche Muster an bestimmten Stellen „sammeln“, Gruppen bilden. Den SOMs ähnliche Strukturen wurden im visuellen Kortex von Säugetieren gefunden, sodass diese Modelle auch biologisch plausibel sind.

Die für dieses Lernverfahren nötige Suche nach dem am stärksten aktivierten Neuron wird allerdings bei der Simulation fast immer durch eine einfache Maximumsuche realisiert, da ein rekurrentes Netz, das die gleiche Aufgabe erfüllen soll, immer mehr Zeit benötigt, einen stabilen Punkt zu erreichen, bzw. keinen findet, also ständig oszilliert. Beispiel für den Ablauf eines unüberwachten Lernprozesses am Beispiel von SOM:

- Präsentation der Eingabe durch entsprechende Aktivierung der Eingabeneuronen
- Propagierung
- Suche nach dem am stärksten aktivierten Neuron (*winner-takes-all*)
- Änderung der Gewichte anhand der Entfernung vom zu aktualisierenden Neuron zum Gewinnerneuron[34]

4 Populäre Netzwertopologien

4.1 Allgemein

Aus all diesen Bauteilen von künstlichen neuronalen Netzen lassen sich Netze „zusammenbauen“. Die bekanntesten Netze sollen hier kurz vorgestellt werden. In vielen Dieser Netze ist ein sogenannter *BIAS* vorhanden. Dieser hat den Wert 1 und hat zu jedem Neuron eine gewichtete Verbindung.

[33] Vgl. Hoffmann, Norbert: Neuronale Netze, Anwendungsorientiertes Wissen zum Lernen und Nachschlagen, Braunschweig 1993, S. 87

[34] Vgl. Lampel, Johannes: Einsatz von neuronalen Netzen in einem Bot für Counterstrike, Online im Internet: http://www.joebot.net/lampel/johannes/bll137.html, Stand 16.11.2001, Abruf 19.4.2002, S.

Damit normiert man die verschiedenen Schwellenwerte der einzelnen Neuronen auf 0 und speichert den eigentlichen Schwellenwert in den Gewichten vom BIAS zum Neuron. Das vereinfacht u.a. die Programmierung von Neuronen-Klassen. Objektorientierte Programmierung ist bei der Modellierung neuronaler Netze sehr von Vorteil.[35]

4.2 Adaline

Adaline ist eine Abkürzung für „adaptive linear neuron“. Das Netz ist einschichtig (s. Abb. 5), die vom Eingang bzw. Ausgang anzunehmenden binären Werte sind -1 und $+1$ ($a = \operatorname{sgn} \varepsilon$).[36] Die Eingangsfunktion ist das Skalarprodukt ($net = \sum_{i=1}^{n} w_{i,j} \cdot e_i$) und die Aktivierungsfunktion ist die Identität (s. Absch. 3.2.3). Der Lernalgorithmus für Adaline ist die Deltaregel[37] (s. Absch. 3.4.2.2).

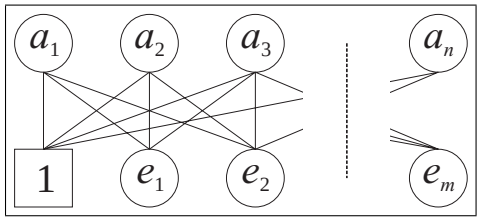

Abbildung 5: Topologie des Adaline

(Quelle: Kinnebrock, Werner: Neuronale Netze, Grundlagen, Anwendungen, Beispiele, 2. Aufl., München 1994, S. 32)

4.3 Perzeptron

Das Perzeptron ist mit dem Adaline hinsichtlich der Topologie identisch und unterscheidet sich nur im Outputbereich (statt $\{-1; 1\}$ ist er $\{0; 1\}$) und etwas in der Berechnung der Gewichtsveränderungen.[38] Die Namensgebung verdankt es dem Ziel-Einsatzgebiet, das ihm von seinem Schöpfer, Frank Rosenblatt, zugedacht war: der Interpretation visueller Eindrücke. Die erste Schicht modelliert die Retina des menschlichen Auges und rastert und verdichtet die Eingangssignale (z.B. Bilder). Diese werden dann der zweiten Schicht, dem eigentlichen Perzeptron zugeführt, das die Interpretation der Signale vornimmt.[39]

4.4 Madaline

Da einschichtige Netze einige Funktionen nicht darstellen können war eine Erweiterung des Konzepts auf mehrere Schichten unumgänglich. Ein zweistufiges Netz ist z.B. das Madaline („multiple adaptive linear neuron“). Es besteht aus einem Adaline und einer zusätzlichen verdeckten Schicht mit einem Ausgabeelement. Eine Besonderheit ist, daß es von jedem Neuron der zweiten, verborgenen Schicht genau eine Verbindung mit dem Gewicht 1 zum Ausgabeelement gibt.[40]

4.5 Multi-Layer-Perzeptron

Das heute am meisten verwendete feedforward-Netz ist das Multi-Layer-Perzeptron (auch Mehrschichten-Netzwerk genannt). Es hat verschiedene verdeckte Schichten und jedes Element kann mit jedem Ele-

[35] Im Internet findet man viele Java-Klassen von Neuronen

[36] Vgl. Kinnebrock, Werner: Neuronale Netze, Grundlagen, Anwendungen, Beispiele, 2. Aufl., München 1994, S.32

[37] Vgl. Hoffmann, Norbert: Neuronale Netze, Anwendungsorientiertes Wissen zum Lernen und Nachschlagen, Braunschweig 1993, S. 33

[38] Vgl. Kinnebrock, Werner: Neuronale Netze, Grundlagen, Anwendungen, Beispiele, 2. Aufl., München 1994, S. 34

[39] Vgl. Kratzer, Klaus Peter: Neuronale Netze, Grundlagen und Anwendungen, 2. Aufl., München 1993, S. 53

[40] Vgl. Kinnebrock, Werner: Neuronale Netze, Grundlagen, Anwendungen, Beispiele, 2. Aufl., München 1994, S. 38

ment der folgenden Schicht verbunden sein. BIAS (s. Absch. 4.2) sind in jeder Schicht möglich. Als Übertragungsfunktion hat sich die sigmoide Funktion

$$a(net) = \frac{1}{1 + e^{-c \cdot net}} \tag{14}$$

als günstig erwiesen. Sie ermöglicht auch den seit 1986 verfügbaren Backpropagation-Algorithmus (s. Absch. 3.4.2.3)

5 Beispiel: Das XOR-Problem

Ein einschichtiges Netz, wie in Abb. 6 links skizziert, kann abhängig vom Schwellenwert nur die Bool'schen Funktionen OR und AND modellieren. Grund dafür ist, daß ein solches Netz die Eingaben nur linear separieren kann.

Man könnte in diesem Beispiel zwar noch an den Gewichten drehen, aber das würde zur gleichen Schlußfolgerung führen: Die XOR-Funktion ist nicht abzubilden. Was die XOR-Funktion abbilden soll ist in Tabelle 1 dargestellt.

XOR-Funktion		
Eingaben		Ausgabe
E_1	E_2	A
0	0	0
1	0	1
0	1	1
1	1	0

Tabelle 1: XOR-Funktion

In Tabelle 2 ist das Netz für alle möglichen Eingaben durchgerechnet. Das Netz verwendet für die Aktivierung die binären Werte $\{0; 1\}$ und die Eingangsfunktion ist das Skalarprodukt.

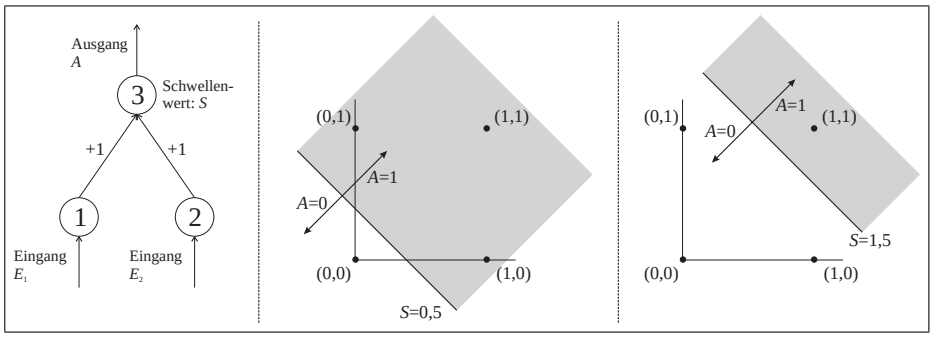

Abbildung 6: Rechenbeispiel: Einschichtiges Netz(Entworfen und gezeichnet: Verfasser)

E_1	E_2	S	net	A	Fkt.
0	0	0,5	$0 \cdot 1 + 0 \cdot 1 = 0 < 0,5$	0	OR
1	0	0,5	$1 \cdot 1 + 0 \cdot 1 = 1 > 0,5$	1	
0	1	0,5	$0 \cdot 1 + 1 \cdot 1 = 1 > 0,5$	1	
1	0	0,5	$1 \cdot 1 + 1 \cdot 1 = 2 > 0,5$	1	
0	0	1,5	$0 \cdot 1 + 0 \cdot 1 = 0 < 1,5$	0	AND
1	0	1,5	$1 \cdot 1 + 0 \cdot 1 = 1 < 1,5$	0	
0	1	1,5	$0 \cdot 1 + 1 \cdot 1 = 1 < 1,5$	0	
1	0	1,5	$1 \cdot 1 + 1 \cdot 1 = 2 > 1,5$	1	

Tabelle 2: Berechnung im einschichtigen Netz

Es wird mindestens eine verdeckte Schicht mit einem Neuronen benötigt, alle anderen Annahmen bleiben gleich (s. Abb 7). Die Berechnungsn für dieses Netz sind in Tabelle 3 durchgeführt.

Vergleicht man die Ein- und Ausgaben mit der geforderten Abbildung in Tabelle 1, sieht man, daß dieses Netz die XOR-Funktion korrekt abbildet.

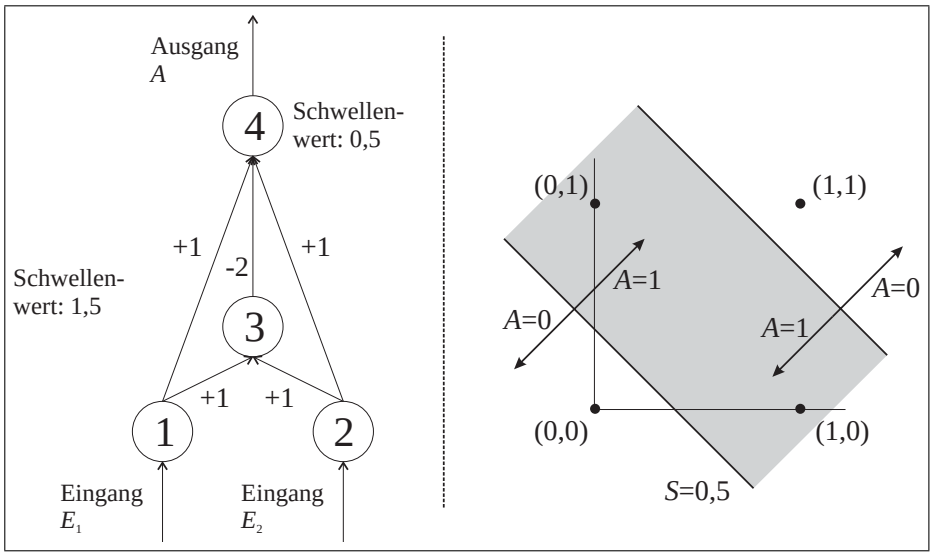

Abbildung 7: Rechenbeispiel: Zweischichtiges Netz

(Quelle: Neuro-Fuzzy-AG, Prof. Dr. Wolfram M. Lippe, Institut für Informatik, Westfälische Wilhelms Universität - Münster: Einführung in Neuronale Netze, Online im Internet: http://wwwmath.uni-muenster.de/SoftComputing/lehre/material/wwwnnscript, Stand 16.1.2001, Abruf 27.3.2002; Kratzer, Klaus Peter: Neuronale Netze, Grundlagen und Anwendungen, 2. Aufl., München 1993, S. 38)

E_1	E_2	net_3	A_3	net_4	A_4
0	0	$0 \cdot 1 + 0 \cdot 1 = 0 < 1,5$	0	$0 \cdot 1 + 0 \cdot 1 + 0 \cdot (-2) = 0 < 0,5$	0
1	0	$1 \cdot 1 + 0 \cdot 1 = 1 < 1,5$	0	$1 \cdot 1 + 0 \cdot 1 + 0 \cdot (-2) = 1 > 0,5$	1
0	1	$0 \cdot 1 + 1 \cdot 1 = 1 < 1,5$	0	$0 \cdot 1 + 1 \cdot 1 + 0 \cdot (-2) = 1 > 0,5$	1
1	1	$1 \cdot 1 + 1 \cdot 1 = 2 > 1,5$	1	$1 \cdot 1 + 1 \cdot 1 + 1 \cdot (-2) = 0 < 0,5$	0

Tabelle 3: Berechnung im zweischichtigen Netz

Literatur

[1] **Grauel, Adolf:** *Neuronale Netze,* Grundlagen und mathematische Modellierung, Mannheim 1992

[2] **Hoffmann, Norbert:** *Neuronale Netze,* Anwendungsorientiertes Wissen zum Lernen und Nachschlagen, Braunschweig 1993

[3] **Kinnebrock, Werner:** *Neuronale Netze,* Grundlagen, Anwendungen, Beispiele, 2. Aufl., München 1994

[4] **Klein, Thorsten:** Diplomarbeit: Untersuchungen zur Struktur- Eigenschaftsbeziehung reiner Stoffe nach inkrementellen und nicht-inkrementellen Verfahren, Universität-Gesamthochschule Paderborn, Online im Internet: http://ac16.uni-paderborn.de/arbeitsgebiete/diplomarbeiten/thorsten_klein/diplom.html, Abruf 27.3.2002

[5] **Kratzer, Klaus Peter:** *Neuronale Netze,* Grundlagen und Anwendungen, 2. Aufl., München 1993

[6] **Lampel, Johannes:** Einsatz von neuronalen Netzen in einem Bot für Counterstrike, Online im Internet: http://www.joebot.net/lampel/johannes/bll137.html, Stand 16.11.2001, Abruf 19.4.2002

[7] **Linder, Hermann:** Biologie, Lehrbuch für die Oberstufe, 20. Aufl., Stuttgart 1989

[8] **Mertens, Peter:** *Lexikon der Wirtschaftsinformatik,* 3. Aufl., Berlin 1997

[9] **Neuro-Fuzzy-AG, Prof. Dr. Wolfram M. Lippe,** Institut für Informatik , Westfälische Wilhelms Universität - Münster: Einführung in Neuronale Netze, Online im Internet: http://wwwmath.uni-muenster.de/SoftComputing/lehre/material/wwwnnscript, Stand 16.1.2001, Abruf 27.3.2002

[10] **Reif, Gerald:** Diplomarbeit: Moderne Aspekte des Wissensverarbeitung, ein interaktiver Lernbehelf für das Web Based Training, Technischen Universität Graz, Online im Internet: http://www.iicm.edu/greif/thesis.html, Stand 20.1.2000, Abruf 17.4.2002

Layout ..... : TEXVersion 3.14159 (MiKTeX 2.1, Win32)
Format-File.: LATEX 2_ε
Makropakete: `umlaute.sty` `layout.sty` `amsmath.sty` `fancyheadings.sty` `german.sty` `picinpar.sty` `a4.sty` `eurosym.sty` `graphicx.sty`