G E R H A R D – M E R C A T O R – U N I V E R S I T Ä T
D U I S B U R G

- Fachbereich Wirtschaftswissenschaft -
Wirtschaftsinformatik und Operations Research
Prof. Dr. Peter Chamoni

Seminar:

Knowledge Discovery in Databases

Thema:

Konnektionistische Systeme mit unüberwachter Adaption

Stefan Schankat
xxx

xxx
xxx

Wirtschaftsinformatik
Produktion / Industrie
6. Fachsemester
Matrikelnummer:xxx

xxx

Inhaltsverzeichnis

Seite

Abbildungsverzeichnis

Seite

Symbolverzeichnis

a_k	Einheit der Ausgabeschicht
d_{Euklid}	euklidische Distanz
e_i	Einheit der Eingabeschicht
r_{ij}	Rückkopplungskoeffizient zwischen zwei Einheiten
v_k	Einheit der Verarbeitumgsschicht
w_{ik}	gewichtete Verbindung zwischen zwei Einheiten
Δw_{ik}	Änderung einer Gewichtung zwischen zwei Einheiten
σ^2	Radius des Korrektureinflusses einer Einheit (Varianz)
ε	effektiver Eingang einer Einheit
δ	Lernrate

1 Einleitung

In den letzten Jahren hat sich die Informationstechnologie im hohen Maße weiterentwickelt. So stellt unzureichender Datenspeicherplatz heutzutage kein Problem mehr dar. Als Resultat dessen, werden immer größere Datenbanken mit immer umfangreicheren Datenbeständen angelegt. Mit wachsender Größe wird es jedoch auch immer aufwendiger und langwieriger, Wissen aus diesen Datenbeständen abzuleiten.[1] Um sich Wettbewerbsvorteile zu sichern, sind jedoch eine schnelle Analyse und Interpretation von Datenbeständen für alle Unternehmen unverzichtbar geworden. Um dieses zu verwirklichen bietet es sich an, computergestützte Verfahren und Methoden wie *Knowledge Discovery in Databases* zu verwenden. Knowledge Discovery in Databases ermöglicht es, Wissen, das implizit in Datenbeständen vorhanden ist zu entdecken und explizit zu machen. Im Unterschied zu Methoden aus der statistischen Datenanalyse oder dem maschinellen Lernen, umfasst Knowledge Discovery in Databases den gesamten Prozess der Wissensentdeckung und beschränkt sich nicht nur auf einzelne Phasen. Dieser komplette Prozess lässt sich in folgende Phasen unterteilen:

- Auswahl des Datenbestandes,
- Aufbereitung der Daten,
- Festlegung der Ziele und Ansätze,
- Analyse des Datenbestandes,
- Interpretation.

Diese Arbeit beschäftigt sich ausschließlich mit der Analysephase, in der versucht wird, Regelmäßigkeiten bzw. Auffälligkeiten in den Datenbeständen zu entdecken und diese als logische oder funktionale Beziehungszusammenhänge abzubilden. Dieser Prozess wird auch als Data Mining bezeichnet. Data Mining bedient sich unterschiedlichster Methoden. Einige dieser Methoden (ohne Anspruch auf Vollständigkeit) sind z. B.:

- Entscheidungsbaumverfahren,
- Verfahren zur Abhängigkeitsentdeckung,
- Clusterverfahren,

[1] Vgl. hierzu und zum Folgenden: Düsing (1999), S. 347 ff.

- Konnektionistische Systeme.

Im Rahmen dieser Arbeit wird gezeigt, wie konnektionistische Systeme für Data Mining genutzt werden können. Dabei wird insbesondere auf konnektionistische Systeme mit unüberwachter Adaption eingegangen. Wie sich zeigen wird, sind diese Verfahren vor allem für die Einteilung eines Datenbestandes in Cluster geeignet. Nach einem allgemeinen Überblick über die Struktur und das Funktionsprinzip von konnektionistischen Systemen wird dann das Verfahren der unüberwachten Adaption näher betrachtet, das schließlich an einem Beispiel einer selbstorganisierenden Karte nach Kohonen konkretisiert wird. Den Abschluss der Arbeit bildet ein Überblick über mögliche Anwendungsgebiete.

2 Konnektionistische Systeme

2.1 Zielsetzung konnektionistischer Systeme

Die ursprüngliche Intention bei der Erforschung konnektionistischer Systeme (bzw. künstlicher neuronaler Netze) war der Versuch einer Nachbildung der neurophysiologischen Vorgänge im menschlichen Gehirn. Es war bekannt, dass das Gehirn aus untereinander verbundenen Nervenzellen (den Neuronen) besteht, die sich gegenseitig durch die Übermittlung elektrischer Signale beeinflussen.[2] 1943 entwickelten W. McCulloch und Walter Pitts ein Modell, das die Arbeitsweise eines solchen Neurons erklären sollte. Dieses Modell entsprach allerdings nur im eingeschränkten Maße der biologischen Funktionsweise eines Neurons und wurde nicht dem Anspruch einer realistischen Umsetzung gerecht.
Heutzutage ist bei konnektionistischen Systemen nicht mehr nur die Nachbildung der Neurophysiologie das Ziel; vielmehr wurde erkannt, dass konnektionistische Systeme durch ihre speziellen Eigenschaften zur Lösung von schlecht strukturierten Problemen beitragen können.[3] Dabei erwiesen sich folgende Eigenschaften als besonders nützlich:

---

[2] Vgl. Nauck et al. (1996), S. 11.

[3] Vgl. Kerling / Poddig (1994), S. 429.

Robustheit:

Konnektionistische Systeme sind auch dann noch funktionsfähig, wenn Teile des Systems ausfallen.[4] Das hat seine Ursache darin, dass das gespeicherte „Wissen“ nicht auf einzelne Einheiten beschränkt, sondern gleich in mehreren Einheiten vorhanden ist. Auf ein herkömmliches serielles System ist eine solche Robustheit nicht übertragbar. Der Ausfall einer einzigen Einheit (z.B. des Prozessors) wird hier schon zum Ausfall des kompletten Systems führen.

Fehlertoleranz:

Konnektionistische Systeme sind bis zu einem gewissen Grad tolerant in Bezug auf fehlerhafte oder verzerrte Eingabedaten.[5] So wie ein Mensch einen teilweise verdeckten Gegenstand in den meisten Fällen dennoch als diesen Gegenstand identifizieren kann, so können auch konnektionistische Systeme fehlende oder verrauschte Daten akzeptieren.

Generalisierungsfähigkeit:

Bei Auswahl geeigneter Beispiele kann ein konnektionistisches System Entscheidungsregeln entwickeln, die nicht nur für die Trainingsbeispiele, sondern allgemein gültig und anwendbar sind. Herkömmliche algorithmische Verfahren sind dagegen immer nur auf eine spezielle Problemstellung anwendbar.

Nichtlinearität:

Konnektionistische Systeme können selbstständig nichtlineare Zusammenhänge in den Datenbeständen erkennen, ohne explizit dafür programmiert zu werden.[6] Dadurch stehen weit mehr Möglichkeiten bei der Mustererkennung zur Verfügung, als bei herkömmlichen Methoden.

---

[4] Vgl. hierzu und zum Folgendem: Patterson (1996), S. 38 ff.

[5] Vgl. hierzu und zum Folgenden: Biethahn (1998), S. 5.

[6] Vgl. hierzu und zum Folgenden: Kerling / Poddig (1994), S. 429.

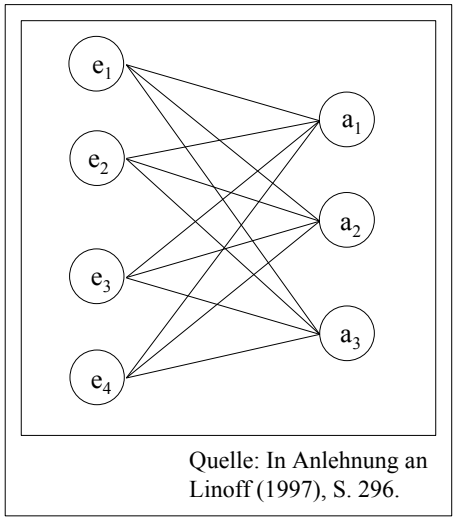
2.2 Strukturprinzip konnektionistischer Systeme

Ein konnektionistisches System ist ein Netz von unabhängigen, parallel arbeitenden Einheiten, die über gewichtete Verbindungen miteinander verknüpft sind. Diese Einheiten repräsentieren die Elemente, die in der Neurophysiologie die Neuronen sind. Alle konnektionistischen Systeme haben gemeinsam, dass einzelne Einheiten unterschiedliche Aufgaben erfüllen können.[7] Es gibt Eingangseinheiten, die lediglich die Eingabewerte in das System einbringen, Verteilungseinheiten, die für die Informationsverarbeitung innerhalb des Netzes zuständig sind und Ausgangseinheiten, die ihre Signale an die Netzausgänge weiterleiten. Aufgrund dessen ist es möglich das Netz in Schichten zu unterteilen, in denen Einheiten mit gleichen Aufgaben zusammengefasst werden. Die Eingabeschicht (input layer) enthält die Eingangseinheiten e_1 bis e_n, die Ausgabeschicht (output layer) die Ausgangseinheiten a_1 bis a_m und eine gegebenenfalls vorhandene Zwischenschicht enthält die Verarbeitungseinheiten v_1 bis v_o. Eine solche Zwischenschicht wird auch als versteckte Schicht (hidden layer) bezeichnet, da es für einen Außenstehenden keine Möglichkeit gibt, mit dieser Schicht in Kontakt zu treten. Typisch für ein konnektionistisches System ist, dass jede Einheit einer Schicht mit jeder Einheit der nachfolgenden Schicht verbunden ist. Alle Verbindungen sind mit Gewichten belegt, die festlegen, wie stark eine Einheit ihren Nachfolger beeinflusst. Es sind sowohl positive als auch negative Gewichte denkbar. Weist eine Verbindung negative Gewichtungen auf, so hemmt eine Einheit ihren Nachfolger.

Mit der Zeit bildeten sich verschiedene Topologien konnektionistischer Systeme heraus. Es gibt einfache Netze, die nur Eingangseinheiten und Ausgangseinheiten aufweisen (vgl. Abb. 1). Sind zusätzlich noch Verarbeitungseinheiten vorhanden, so besitzt das Netz mindestens eine versteckte Schicht von Einheiten (vgl. Abb. 2). Theoretisch ist eine unbegrenzte Anzahl von versteckten Schichten denkbar. Wie viele Einheiten in einer Schicht vorhanden sind, und wie viele Schichten insgesamt existieren sollen, kann vom Anwender je nach Bedarf frei gewählt werden. Es ist jedoch zu bedenken, dass die Vorgänge in einer versteckten Schicht dem Anwender fast völlig verborgen bleiben. Nur die

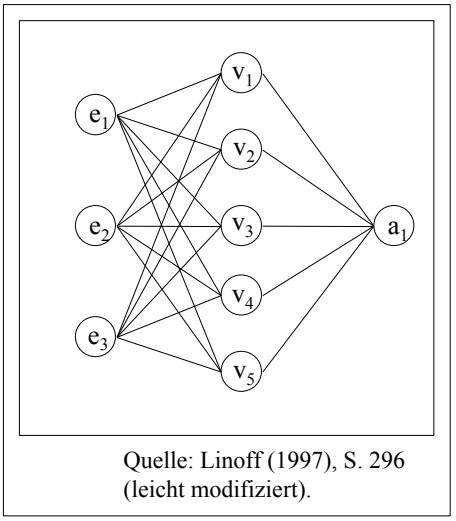
[7] Vgl. hierzu und zum Folgenden: Hoffman (1993), S. 36 ff.

Eingangsschicht und die Ausgangsschicht stehen in einem direkten Kontakt zu ihrer Unwelt.



Quelle: In Anlehnung an Linoff (1997), S. 296.

Abb. 1: Einfaches konnektionistisches System mit 4 Eingabeeinheiten und 3 Ausgabeeinheiten.



Quelle: Linoff (1997), S. 296 (leicht modifiziert).

Abb. 2: Konnektionistisches System mit einer versteckten Schicht.

Es kann zusätzlich zwischen Netzen ohne Rückkopplung (feedforward-Netze) und Netzen mit Rückkopplung (rekurrente-Netze) unterschieden werden.
Feedforward-Netze erlauben lediglich einen Datenfluss in eine Richtung; ausgehend von den Eingangseinheiten bis zu den Ausgangseinheiten, d. h. es existiert keine Möglichkeit, eine Einheit ein zweites Mal zu erreichen.[8] Gerade das ist aber in rekurrenten-Netzen erwünscht. Die Rückkopplung kann, abhängig von der gewählten Topologie, innerhalb der Schichten von Einheit zu Einheit (lateral feedback), zwischen den Schichten (indirect feedback) oder nur auf einzelne Einheiten bezogen erfolgen (direct feedback).

2.3 Funktionsprinzip konnektionistischer Systeme

Alle konnektionistischen Systeme, unabhängig von ihrer Architektur oder ihrem Lernverfahren, funktionieren nach einem übergeordneten Prinzip. An die Eingangseinheiten werden Werte angelegt, die durch das Netz fließen, dort verarbeitet werden und an den Ausgangseinheiten einen Output erzeugen. In einem zweiten Schritt werden die Gewichte der Einheiten nach einem

[8] Vgl. hierzu und zum Folgenden: Biethahn (1998), S. 8.

vorgegebenen Verfahren verändert. Die Korrektur der Gewichte stellt den eigentlichen Adaptionsvorgang des Netzes dar. Diese beiden Vorgänge bilden einen iterativen Prozess, der für alle Elemente des Datenbestandes durchgeführt wird. Für jede Eingabe wird erst ein Ausgabewert errechnet und dann eine Gewichtskorrektur vorgenommen.

2.3.1 Berechnung der Ausgabewerte

Die Berechnung der Ausgabewerte erfolgt durch den Fluss der Eingabewerte durch das Netz bis zu den Ausgangseinheiten. Eine Eingabe die an die Eingangseinheiten angelegt wird, wird mit dem Wert der Verbindung zur nachfolgenden Einheit gewichtet. Der effektive Eingang ε einer beliebigen Einheit bestimmt sich also aus den Ausgabewerten der Vorgängereinheiten, multipliziert mit den Gewichten der Verbindungen zu eben dieser beliebigen Einheit. Sowohl Ausgabewerte als auch Gewichte können als Vektoren aufgefasst werden, so dass sich der effektive Eingang einer Einheit auch als Skalarprodukt dieser beiden Vektoren berechnen lässt:

$$\varepsilon = \sum_{i=1}^{n} w_i e_i = we \qquad (2.1).$$

Der Eingabevektor $\boldsymbol{e}$ hat gerade so viele Elemente (nämlich n), wie Eingabeeinheiten vorhanden sind. Auch der Gewichtsvektor $\boldsymbol{w}$ weist diese Anzahl von n Elementen auf, da jede Einheit der nachfolgenden Schicht mit jeder Eingabeeinheit verbunden ist.

Aus dem effektiven Eingang wird innerhalb der Einheit über die Aktivierungsfunktion die Aktivität c berechnet, aus der dann durch die Ausgabefunktion der Ausgabewert a ermittelt wird. Da eine Unterscheidung von Aktivierungsfunktion und Ausgabefunktion nur bei bestimmten Netztypen notwendig ist, wird häufig der Ausgabewert nur über eine sogenannte Transferfunktion berechnet. Es ist zu beachten, dass diese Prozedur bei den Eingangseinheiten nicht durchgeführt wird. Der Output der Eingangseinheiten ist identisch mit den eingegebenen Werten.[9]

Ein biologisches Neuron kann nur zwei mögliche Ausgabewerte aufweisen: 0 oder 1. Entweder die Zelle feuert (1) oder sie verbleibt in ihrem Ausgangszustand

[9] Vgl. Chamoni (1999), S. 368.

(0). Die meisten konnektionistischen Systeme erfordern jedoch Transferfunktionen, die Ausgabewerte aus einem stetigen Intervall erlauben. Häufig verwendete Transferfunktionen, die dieses Kriterium erfüllen, sind lineare Funktionen, sigmoide (logistische) Funktionen und der Tangens hyperbolicus. Die Verwendung einer linearen Transferfunktion ist kritisch zu beurteilen, da durch sie lediglich eine Regression berechnet wird. Die sigmoide Funktion und der Tangens hyperbolicus sind dagegen nichtlineare Funktionen und entsprechen damit dem Anspruch von konnektionistischen Systemen, nichtlineare Zusammenhänge erkennen zu können. Der Wertebereich der logistischen Funktion reicht von 0 bis 1, der Wertebereich des Tangens hyperbolicus von -1 bis +1.[10] Welche Funktion verwendet wird, ist abhängig von der gewählten Netztopologie. Soll eine hemmende Wirkung von Einheiten auf ihre Nachfolger möglich sein, so ist der Tangens hyperbolicus zu wählen. Ein negativer Ausgabewert einer Einheit bewirkt dann die Hemmung der nachfolgenden Einheit.

Die Ausgabe einer Einheit wird schließlich an die Nachfolgereinheit weitergegeben, deren effektiver Eingang sich dann wieder durch Gewichtung der Ausgabewerte aller Vorgängereinheiten mit den Verbindungswerten errechnet.

Wie viele Ausgaben das System wieder an die Umwelt ausgibt, ist abhängig von der gewählten Struktur der Ausgabeschicht. Es ist denkbar, dass *m* Ausgabeeinheiten definiert wurden, die dann auch *m* verschiedene Werte ausgeben. Wird eine Klassifikation gewünscht, so wird von den *m* vorhandenen Ausgabeeinheiten nur die Einheit feuern, die die entsprechende Klasse repräsentiert. Werden konnektionistische Systeme dagegen zur Prognose benutzt, so ist ein Netz wie in Abbildung 2 zu wählen, bei dem nur eine Ausgabe, nämlich der erwünschte Prognosewert erfolgt.

2.3.2 Korrektur der Gewichte

Die Ausgabe eines oder mehrere Werte zieht eine Korrektur der Gewichte nach sich. Dies ist der eigentliche Vorgang der Adaption. Dabei wird zwischen überwachter, verstärkender und unüberwachter Adaption unterschieden. Konnektionistischen Systemen, die überwacht lernen, wird neben den

[10] Vgl. Berry / Linoff (1997), S. 298 f.

Eingabewerten auch noch der erwünschte Ausgabewert präsentiert. Aufgrund der Differenz zwischen Ist-Ausgabe und Soll-Ausgabe erfolgt dann die Korrektur der Gewichte. Beim verstärkenden Lernen ist die Soll-Ausgabe zwar bekannt, wird dem Netz aber nicht präsentiert. Es wird lediglich angegeben, ob richtig oder falsch klassifiziert wurde. Bei der unüberwachten Adaption werden nur die Eingabedaten an das Netz angelegt, eine Soll-Ausgabe ist in diesem Fall nicht bekannt und deshalb auch nicht vorhanden. Das Netz versucht, die Gewichte in einem Prozess der Selbstorganisation eigenständig zu korrigieren.

Die Grundidee der Gewichtsänderung liegt darin, dass ein Vergleich zwischen der erwünschten Ausgabe und der tatsächlichen Ausgabe einer Einheit stattfindet. Abhängig von der Höhe der Differenz wird dann eine entsprechend starke Gewichtsänderung vollzogen. Eine Methode dieser Art wird durch die Delta-Lernregel beschrieben:

$$\Delta w_{ik} = \delta * e_i * \Delta a_k \qquad (2.2).$$

Hierbei ist Δw_{ik} die Änderung des Gewichtes von Einheit i zu Einheit k, e_i die Eingabe, Δa_k die Differenz zwischen der tatsächlichen Ausgabe und der erwünschten Ausgabe und δ die sogenannte Lernrate.[11] Diese liegt zwischen 0 und 1 und bestimmt in welchem Maße die Gewichtsanpassung durchgeführt wird. Es ist einfach zu erkennen, dass keine Veränderung des Gewichtwertes erfolgt, wenn tatsächliche und erwünschte Ausgabe übereinstimmen, Δa_j also 0 ist.

3 Unüberwachte Adaptionsverfahren

In Kapitel 2 wurde eine allgemeine Darstellung der Funktionsweise von konnektionistischen Systemen entworfen. Im Folgenden wird nun näher auf die unüberwachte Adaption eingegangen.

In konnektionistischen Systemen mit unüberwachter Adaption findet nur die Eingabe der Eingangswerte statt. Da a priori keine Soll-Ausgabe bekannt ist, kann auch keine Differenz zwischen tatsächlicher Ausgabe und erwünschter Ausgabe berechnet werden, aufgrund derer die Gewichte korrigiert werden. Dennoch ändern sich die Netzgewichte mit jeder Eingabe. Diese Form der Adaption ist biologisch plausibler als ein überwachter Lernvorgang, da auch in der Realität

nicht immer im Voraus eine korrekte Lösung bekannt ist. Wenn eine Soll-Ausgabe fehlt, versucht das Netz selbstständig eine Adaptionsmethode zu finden, die eine optimale Korrektur der Gewichte bietet. Dem Netz ist dabei nicht bekannt, in welche Richtung die Korrektur zu erfolgen hat. Aufgrund dieser Eigenschaften werden konnektionistische Systeme mit unüberwachter Adaption häufig auch als selbstorganisierende Netze bzw. selbstorganisierende Karten bezeichnet. Diese Systeme werden hauptsächlich zur Clusterbildung eingesetzt. Sie sind in der Lage, charakteristische Häufungen zu erkennen und für diese selbstständig Klassifikationskriterien zu entwickeln. Konnektionistische Systeme, die überwacht lernen, können Daten nur in vorgegebene Klassen einteilen, selbstorganisierende Netze dagegen bilden diese Klassen selber. Das bringt insbesondere Vorteile, wenn keine Informationen über Zugehörigkeiten innerhalb des Datenbestandes und über eine mögliche Anzahl von Klassen bestehen. Das selbst entwickelte Klassifikationsschema wird durch die Gewichtskonstellation des Netzes repräsentiert. Diese Gewichtskonstellation stellt eine Art Abbildung der „Umwelt“ des Netzes dar. Somit sollte das Netz auch in der Lage sein, sich anzupassen und zu reorganisieren, wenn sich relevante Umwelteinflüsse ändern.

Selbstorganisierende Systeme können verschiedene Formen aufweisen.[12] Denkbar sind Netze mit Eingabeschicht, Ausgabeschicht und zwischengeschalteten verborgenen Schichten. In solchen Systemen findet die Selbstorganisation schichtweise statt. Eine andere Möglichkeit wäre eine Struktur, die nur eine Eingabeschicht und eine Ausgabeschicht aufweist. Auf eine solche Struktur wird im nächsten Kapitel im Rahmen der selbstorganisierenden Karten nach Kohonen noch näher eingegangen.

Konnektionistische Systeme mit unüberwachter Adaption verwenden häufig einen Lernalgorithmus, der einzelne Einheiten, die besonders intensiv auf Eingaben reagieren bevorzugt. In den Schichten des Systems konkurrieren die Einheiten um die „beste“ Ausgabe und nur die Einheit mit dem besten Wert darf dann ihre Ausgabe weitergeben, während alle anderen inaktiv bleiben. Die übrigen Einheiten einer Schicht werden durch die Siegereinheit gehemmt. Zusätzlich erhält auch nur diese Siegereinheit das Recht auf eine Gewichtskorrektur

---

[11] Vgl. Kinnebrock (1994), S. 24.

[12] Vgl. hierzu und zum Folgenden: Haykin (1994), S. 352.

(„winner-takes-all“). Diese Art des Lernens, das auch in beliebig vielen Schichten durchgeführt werden kann, wird als Wettbewerbslernen bezeichnet. Sind nur eine Eingabeschicht und eine Ausgabeschicht vorhanden, wird in einem solchen Fall die Ausgabeschicht auch als Wettbewerbsschicht bezeichnet.

4 Selbstorganisierende Karten nach Kohonen

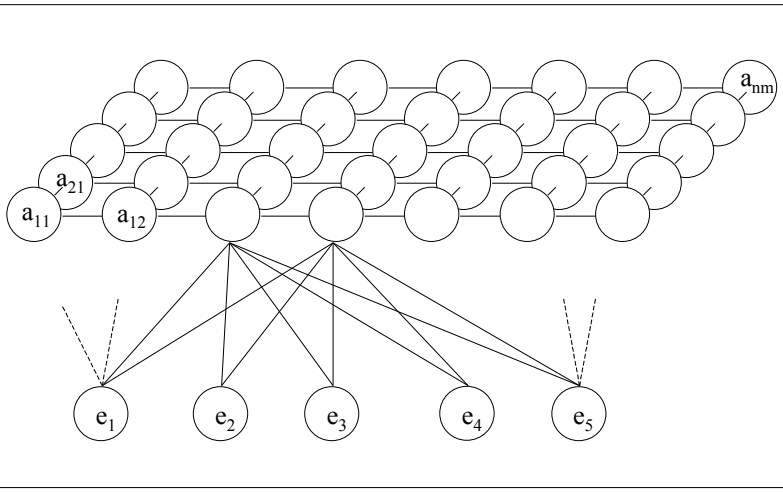
Die selbstorganisierenden Karten von Kohonen sind ein spezielles Beispiel für konnektionistische Systeme mit unüberwachter Adaption. In einer selbstorganisierenden Karte existiert lediglich eine Schicht von Eingangseinheiten und eine Schicht von Ausgangseinheiten. Jede Einheit der Ausgabeschicht ist vollständig mit jeder Einheit der Eingabeschicht verbunden. Die Anzahl der Eingangseinheiten hängt von der Anzahl der zu untersuchenden Merkmale ab, während die Anzahl der Ausgabeeinheiten frei wählbar ist. Die 1 bis n Eingangseinheiten repräsentieren somit die relevanten Einflüsse der Umwelt einer selbstorganisierenden Karte.

Selbstorganisierende Karten verwenden einen unüberwachten Adaptions-algorithmus, der die Einheiten der Ausgabeschicht während der Selbstorganisation auf einer räumlichen Karte so anordnet, dass jede Einheit nach dem Training einem bestimmten Cluster von Eingabevektoren entspricht.[13] Die Ausgangseinheiten sind dabei in einer vorher festgelegten topologischen Struktur angeordnet. Diese Struktur kann unterschiedliche Formen annehmen. Am weitesten verbreitet sind zweidimensionale Strukturen, bei denen die Ausgabeeinheiten in der Form eines Rechtecks bzw. eines Gitters mit n * m Einheiten angeordnet sind. Denkbar sind allerdings auch eindimensionale Topologien, die den Einheiten eine Kette als Struktur zuweisen, bzw. multidimensionale Topologien, die die Struktur eines Hyperquaders aufweisen. Ziel einer selbstorganisierenden Karte ist es nun, einen multidimensionalen Eingabevektor auf eben diese gewählten Dimensionen zu reduzieren und damit besser analysierbar zu machen.[14] Im weiteren werden die Einheiten der Eingabeschicht als e_1 bis e_n und die Ausgangseinheiten als a_1 bis a_m bezeichnet.

[13] Vgl. Bigus (1996), S. 71.

[14] Vgl. Bigus (1996), S. 72.

Die gewichtete Verbindung zwischen der i-ten Einheit der Eingangsschicht und der k-ten Einheit der Ausgangsschicht wird durch w_{ik} gekennzeichnet.
Im Unterschied zu anderen unüberwachten Lernverfahren besteht bei einer Kohonen Karte zusätzlich eine Nachbarschaftsbeziehung zwischen den Einheiten. Nach der Adaptionsphase sind die Ausgabeeinheiten so angeordnet, dass benachbarte Einheiten ähnlich auf eine Eingabe reagieren, während entfernte Einheiten für andere Eingabevektoren zuständig sind. Konkret lässt sich diese Nachbarschaft in den Gewichtsvektoren erkennen. Eng beieinander liegende Einheiten weisen ähnliche Gewichte auf und reagieren damit auch ähnlich auf spezielle Eingaben. In Abbildung 3 ist eine solche selbstorganisierende Karte exemplarisch dargestellt. Die 35 Einheiten der Ausgangsschicht sind vollständig mit jeder Eingangseinheit verbunden. Aus Gründen der Übersichtlichkeit wurden jedoch nur die Verbindungen von 2 Ausgangseinheiten dargestellt. Die bereits erwähnten Nachbarschaftsbeziehungen zwischen den Einheiten sollen durch die Gitterstruktur der Ausgabeschicht verdeutlicht werden.



Quelle: In Anlehnung an Scherer (1997) S. 94.

Abb. 3: Selbstorganisierende Karte nach Kohonen.

Zum Beginn des Lernalgorithmus werden die Gewichte zwischen Ausgangseinheiten und Eingangseinheiten zufällig gewählt. Graphisch betrachtet sind die Gewichtsvektoren somit frei und wahllos im Raum verteilt. Mit der Zeit sollen sich diese Vektoren im Raum organisieren und gleichmäßig anordnen.

Wird nun ein Eingangsvektor an das Netz angelegt, so wird es eine Ausgangseinheit geben, die am stärksten auf diesen Vektor reagiert, weil ihr Gewichtsvektor zufällig dem Eingabevektor am ähnlichsten ist. Die Ermittlung dieses Erregungszentrums a_j kann auf unterschiedliche Art und Weise geschehen. Sind beide Vektoren normiert, so gewinnt die Einheit, deren Skalarprodukt den höchsten Wert aufweist. Bei nicht normierten Vektoren wird häufig die euklidische Distanz ermittelt, indem die Vektorelemente paarweise verglichen werden.

In diesem Fall gewinnt die Einheit, die die minimalste euklidische Distanz aufweist:

$$d_{Euklid} = \sqrt{\sum_{i=1}^{n} (w_{ik} - o_{e_i})^2} \qquad (4.1).$$

Hierbei ist o_{e_i} der Wert, der an die Eingabeeinheit e_i angelegt und von dieser auch wieder in dieser Form ausgegeben wird. Gleichzeitig ist o_{e_i} die i-te Komponente des Eingabevektors, der ja so viele Komponenten enthält, wie es Eingabeeinheiten gibt. Die Variable w_{ik} bezeichnet den Wert der Gewichtung zwischen der Eingabeeinheit e_i und einer Ausgabeeinheit a_k.

Nach der Ermittlung des Erregungszentrums erfolgt die Korrektur der Netzgewichte. Diese beschränkt sich nicht nur auf das Erregungszentrum, sondern schließt auch die direkte Nachbarschaft der Siegereinheit ein. Es ist zu beachten, dass die Gewichtsvektoren der Nachbareinheiten gerade zu Beginn des Trainings nicht unbedingt die Vektoren sein müssen, die dem Eingabevektor am zweitähnlichsten sind. Durch die anfangs gewählten Zufallswerte werden sie sich wahrscheinlich drastisch von diesem unterscheiden. Entscheidend ist hier nur die Nachbarschaft zu Siegereinheit.[15] Die Nachbarschaft wird vorher durch eine gewählte Funktion spezifiziert. Vorzugsweise wird eine abnehmende Nachbarschaftsfunktion gewählt, so dass zu Beginn des Trainings viele Einheiten im Umkreis der Siegereinheit eine Gewichtskorrektur erfahren. Mit fortschreitendem Training wird die Nachbarschaft dann jedoch immer kleiner. Es ist sogar denkbar, dass gegen Ende des Trainings nur noch die Gewichte der Siegereinheit korrigiert werden. Dieses Verfahren wird verwendet, da das System so zunächst eine Grobstruktur in dem dargebotenen Datenbestand erkennen kann.

Mit abnehmender Nachbarschaft wird dann die Feinstruktur ausgebildet. Um die Nachbarschaft eines Neurons zu ermitteln, kann auf unterschiedliche Vorgehensweisen zurückgegriffen werden.

1. Ermittlung der Nachbarschaft aufgrund der definierten Gitterstruktur:
   Die Nachbarschaft einer Einheit kann mit dieser Methode direkt anhand des definierten Gitters bestimmt werden.[16] Wird z. B. eine Nachbarschaft von 2 vorgegeben, so sind alle Einheiten betroffen, die mit 2 oder weniger Schritten von a_j erreicht werden können. In Abbildung 3 wären dies dann insgesamt 12 Einheiten. Eine Nachbarschaft von 1 dagegen würde nur 4 Einheiten betreffen. Es ist zu beachten, dass in Abbildung 4 keine diagonalen Verbindungen bestehen und die 4 Einheiten, die diagonal zum Erregungszentrum positioniert sind deshalb auch nicht der Nachbarschaft zugeordnet werden können. Eine solche Nachbarschaftsfunktion kann nur die Werte 0 und 1 annehmen, ist damit also eine scharf begrenzte Nachbarschaftsfunktion. Ist eine Einheit nicht Teil der Nachbarschaft, weil ihre Distanz zum Erregungszentrum größer als die vorher gewählte Nachbarschaft ist, so nimmt die Funktion den Wert 0 an und es findet keine Gewichtsänderung statt.
   Eine Verringerung der Nachbarschaft kann z. B. durch eine Funktion dieser Form erfolgen:

$$r(t) = r(0) * (1 - \frac{t}{T}) \qquad (4.2).$$

   Dabei ist *r(0)* ein anfänglich vorgegebener Wert für den Nachbarschaftsradius, *T* die Anzahl der insgesamt zu präsentierenden Eingabevektoren und *t* die Anzahl der bisher präsentierten Vektoren.[17]

2. Nachbarschaftsfunktionen mit Abstufungen
   [18]Mit Hilfe einer solchen Nachbarschaftsfunktion, sind stetige Abstufungen bezüglich der Nachbarschaft des Erregungszentrums a_j möglich.

---

[15] Vgl. Biethahn (1998), S. 25 f.

[16] Vgl. hierzu und zum Folgenden: Scherer (1997), S. 98

[17] Vgl. Nauck et al. (1996), S.130 f.

[18] Vgl. hierzu und zum Folgenden: Kinnebrock (1994), S. 83 f.

Nachbarschaftsfunktionen dieser Art werden häufig durch eine Gauß Funktion beschrieben:

$$r_{ij} = e^{-((a_i - a_j)^2 / 2*\sigma^2)} \qquad (4.3).$$

Hierbei stellt r_{ij} den Rückkopplungskoeffizienten zwischen dem Erregungszentrum a_j und einer beliebigen Ausgabeeinheit a_i dar. Die Varianz σ^2 bestimmt die räumliche Ausdehnung der Gaußkurve und damit auch den Einflussbereich der Siegereinheit. Für große σ^2 existiert eine weite Rückkopplung, für kleine σ^2 verengt sich der Radius des Korrektureinflusses. Auch bei dieser Form der Nachbarschaftsfunktion kann noch eine Verringerung derselben erfolgen. Dazu muss z.B. einfach festgelegt werden, dass nach x Iterationen σ^2 um einen beliebigen Wert reduziert wird.

Durch die Korrektur der Gewichte werden die Gewichtseinheiten bildlich betrachtet in die Richtung des Eingabevektors gezogen. Die Siegereinheit erhält die stärkste Korrektur, die Gewichte der Nachbarschaftseinheiten werden nur in abgeschwächter Weise korrigiert. Üblicherweise nimmt nicht nur die Nachbarschaft im Verlauf des Trainings ab, auch die Lernrate δ wird sich verringern. Das äußert sich darin, dass die Gewichte anfangs stark korrigiert werden, mit der Zeit aber nur noch minimale Änderungen erfolgen. Die Änderung der Lernrate kann analog zur Änderung der Nachbarschaft in Formel (4.2) erfolgen:

$$\delta(t) = \delta(0) * (1 - \frac{t}{T}) \quad (4.4).$$

δ ist die anfänglich gewählte Lernrate, T die Anzahl der insgesamt zu präsentierenden Eingabevektoren und t die Anzahl der bisher präsentierten Vektoren.[19]

Die Gewichtsänderungen in einer selbstorganisierenden Karte werden also durch 2 Aspekte bestimmt:

- Die Lernrate legt fest, wie stark die Gewichte verändert werden. Üblicherweise nimmt die Lernrate im Trainingsverlauf ab.

[19] Vgl. Nauck et al. (1996), S. 129.

- Die Nachbarschaftsfunktion legt fest, wie viele Einheiten das Recht erhalten, ihre Gewichte ändern zu lassen. Auch diese Größe sollte im Zeitablauf abnehmen.

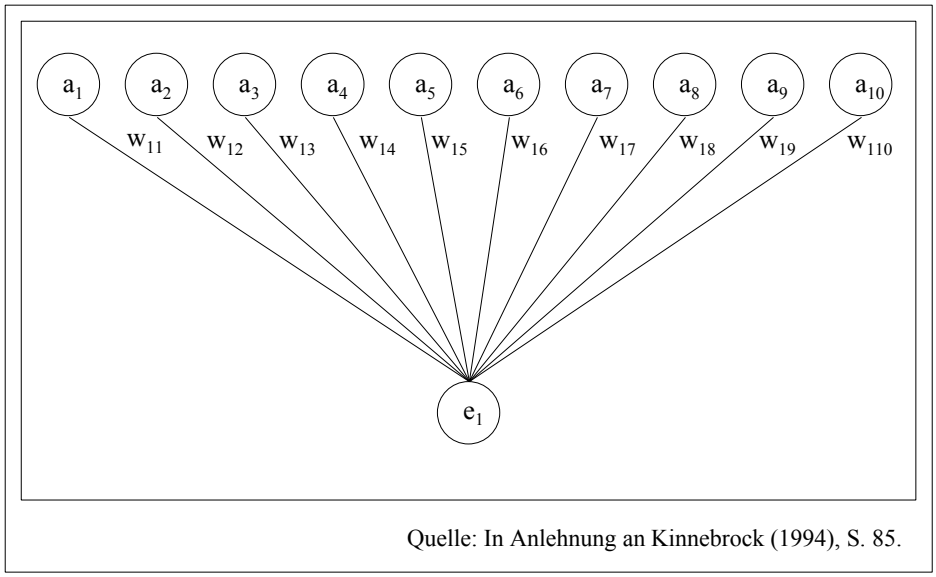
Als Resultat aus den oben beschriebenen Verfahren ergibt sich folgenden Formel für die Gewichtsänderung:

$$w_{ik}^{neu} = w_{ik}^{alt} + \delta(t) * r_{ij} * \left(o_{e_i} - w_{ik}^{alt}\right) \quad (4.5).$$

Das neue Gewicht von Einheit e_n zu Einheit a_k ergibt sich aus dem alten Gewichtswert, zuzüglich der Differenz zwischen Eingabevektor und Ausgabevektor, gewichtet mit der Lernrate δ und dem Rückkopplungskoeffizienten r_{ij}.
Nach dem Abschluss der Adaptionsphase sollten sich die Gewichtvektoren gleichmäßig im Raum verteilt haben, so dass jede Ausgabeeinheit, bzw. jeder Cluster von Ausgabeeinheiten auf einen speziellen Eingabevektor reagiert. Durch den Lernalgorithmus wurden die Gewichtsvektoren so angepasst, dass jeder Gewichtsvektor einem Cluster von Eingabevektoren entspricht.[20]

Die Funktionsweise einer selbstorganisierenden Karte lässt sich am besten an einem konkreten Beispiel darstellen. Aus Gründen der besseren Verständlichkeit wird ein sehr einfaches Modell gewählt, das lediglich eine Einheit in der Eingangsschicht und 10 Einheiten in der Ausgangsschicht aufweist. Die 10 Einheiten der Ausgangsschicht sind in einer linearen Kette angeordnet. So ist jeder Ausgangseinheit nur ein Gewicht zugewiesen (siehe Abbildung 4).

[20] Vgl. Rojas (1993), S. 339.



Quelle: In Anlehnung an Kinnebrock (1994), S. 85.

Abb. 4: Eindimensionale selbstorganisierende Karte mit eindimensionaler Eingabe.

Die Gewichte werden zufällig aus den natürlichen Zahlen zwischen 0 und 4 ausgewählt. Dem System werden dann nacheinander zufällig ausgewählte natürliche Zahlen zwischen 0 und 4 präsentiert. Es soll nun exemplarisch gezeigt werden, wie sich das Netz selbst organisiert, so dass die Gewichte der Ausgangseinheiten schließlich der Größe nach geordnet sind. Im Folgenden wird nach dem, bereits beschriebenen Schema vorgegangen:

1. Es erfolgt zunächst eine zufällige Initialisierung der Verbindungsgewichte.
2. Der Eingangseinheit werden nacheinander zufällig ermittelte Werte aus dem zulässigen Wertebereich präsentiert.
3. Ermittlung des Erregungszentrums durch Verwendung der Formel für die euklidische Distanz.
4. Gewichtskorrektur durch Formel (4.5).
5. Fortfahren bei 2.

Es wird eine Lernrate von $\delta = 0.8$ und eine Varianz von $\sigma^2 = 2$ gewählt. Für die Nachbarschaftsfunktion wird die in 2. beschriebene Funktion mit Abstufungen gewählt, um verdeutlichen zu können, dass Einheiten in der Nachbarschaft der Siegereinheit eine abgeschwächte Gewichtskorrektur erfahren.

Die zufällige Gewichtsinitialisierung hat folgende Werte für die 10 Verbindungen ergeben. Dabei bezeichnet w_{ik} den Wert der Gewichtung von der i-ten Eingabeeinheit zur k-ten Ausgabeeinheit.

w_{11}=3 w_{12}=1 w_{13}=0 w_{14}=2 w_{15}=3 w_{16}=4 w_{17}=3 w_{18}=2 w_{19}=0 w_{110}=2

Die erste dem System präsentierte Eingabe hat den Wert 1. Ausgabeeinheit 2 gewinnt, da ihr Verbindungsgewicht mit der Eingabe übereinstimmt, die Distanz also 0 ist. Somit ist das Erregungszentrum a_j = 2. Die Ermittlung des Rückkopplungskoeffizienten erfolgt nun für jede Einheit nach Formel (4.4). Es ergibt sich z.B. für Ausgabeeinheit 1 ein Koeffizient von:

$$r_{12} = e^{-\left[(1-2)^2/2*2\right]} = 0{,}7788.$$

Damit kann nun nach (4.5) das neue Gewicht w_{11} berechnet werden:

$$w_{11}^{neu} = 3 + \left(0{,}8 * 0{,}7788 * \left[1-3\right]\right) = 1{,}754.$$

Da das System nur mit natürlichen Zahlen operieren soll, wird auf 2 aufgerundet. Für die anderen Ausgangseinheiten ergeben sich natürlich andere Rückkopplungskoeffizienten. In der folgenden Tabelle sind nun die jeweiligen Rückkopplungskoeffizienten und geänderten Gewichte für 2 Iterationen eingetragen. Außerdem zeigt die letzte Spalte den Zustand nach 100 Wiederholungen.[21] Es ist deutlich zu erkennen, dass nur das Erregungszentrum eine vollständige Korrektur erfährt. Je größer die Entfernung einer Einheit vom Erregungszentrum ist, umso schwächer ist die Korrektur. Nach 100 Iterationen ist bereits zu erkennen, dass die Gewichte der Größe nach geordnet wurden. Die Ausgabeeinheiten sind nun für spezielle Eingaben zuständig. So sind a_1 und a_2 z. B. auf den Eingabevektor $\boldsymbol{e}$ = (4) spezialisiert.

[21] Die Werte wurden durch ein Pascal Programm errechnet, das auf einem Beispiel von Kinnebrock basiert. Vgl. Kinnebrock (1994), S. 158 ff.

Ausgabeeinheit mit Anfangsgewicht	Iteration 1: v = 1		Iteration 2: v = 4		nach Iteration 100	
	r_{ij}	w_{ik}^{neu}	r_{ij}	w_{ik}^{neu}		w_{ik}^{neu}
a_1 : 3	0,7788	2	0,0019	2		4
a_2 : 1	**1**	1	0,0183	1		4
a_3 : 0	0,7788	1	0,1054	1		3
a_4 : 2	0,3679	2	0,3679	2		3
a_5 : 3	0,1054	3	0,7788	4	...	2
a_6 : 4	0,0183	4	**1**	4		2
a_7 : 3	0,0019	3	0,7788	4		2
a_8 : 2	0	2	0,3679	3		1
a_9 : 0	0	0	0,1054	0		0
a_{10} : 2	0	2	0,0183	2		0

Tabelle 1: Rückkopplungskoeffizienten und neue Gewichte.

5 Anwendungen konnektionistischer Systeme

Konnektionistische Systeme mit unüberwachten Adaptionsverfahren werden hauptsächlich im Bereich der Mustererkennung eingesetzt. Kohonen versuchte mit seinen selbstorganisierenden Karten eine Methode zur Spracherkennung zu realisieren. Es sind allerdings auch betriebswirtschaftliche Anwendungen denkbar. Unüberwachte Adaption wird verwendet, wenn das Ergebnis einer Datenbestandsanalyse nicht im Voraus bekannt ist. Die durch das System zu klärende Frage ist dann, in welcher Beziehung die Daten untereinander stehen.[22] Eine mögliche Anwendung für diesen Fall ist z. B. eine Käuferanalyse, bzw. eine Marktsegmentierung im Marketing. Ein konnektionistisches System ist in der Lage, die verborgenen Strukturen des Datenbestandes zu entdecken und kann auf diese Weise z.B. homogene Käufergruppen identifizieren. Dazu ist es natürlich nötig, während der Trainingsphase zu verfolgen, welche Eingabe welchem Ausgabecluster zugeordnet wird.

Obwohl es der eigentlichen Idee eines unüberwacht lernenden Systems widerspricht, können sie auch zur Klassifikation verwendet werden. Der

[22] Vgl. Bigus (1996), S 63.

entscheidende Unterschied zu überwachten Methoden ist der, dass das System die Klassen zunächst aufgrund des Datenbestandes selbstständig definiert, während diese bei überwachten Verfahren vorgegeben werden. In einem zweiten Schritt können dann neue Datensätze diesen vorher gebildeten Klassen zugeordnet werden. Natürlich wäre es auch möglich, dass das System auch für die neuen Daten noch Gewichtsänderungen durchführt, wenn diese signifikant von den bisher betrachteten Daten abweicht. Das System reagiert praktisch ständig auf sich ändernde Umwelteinflüsse.
Einmal analysierte und strukturierte Datenbestände können natürlich auch als Grundlage weiterer Analysemethoden verwendet werden. Eine in verschiedene Cluster unterteilte Menge könnte z.B. die Grundlage eines Entscheidungsbaumes bilden.

6 Zusammenfassung und Fazit

Es wurde gezeigt, dass konnektionistische Systeme mit unüberwachter Adaption innerhalb des Knowledge Discovery in Databases Prozess insbesondere zur Erkennung von Regelmäßigkeiten und zur Bildung von Clustern genutzt werden können. Die speziellen Eigenschaften konnektionistischer Systeme machen sie insbesondere zur Lösung von schlecht strukturierten Problemen attraktiv. Der Einsatz von konnektionistischen Systemen ist somit nicht mehr nur auf den Versuch einer Nachbildung von biologischen Vorgängen beschränkt. Es ist jedoch anzumerken, dass konnektionistische Systeme auch negative Eigenschaften aufweisen. So ist das Ergebnis im hohen Maße abhängig von der Qualität des ausgewählten Datenbestandes. Aus einer Menge von Daten, die für das eigentliche Ziel irrelevant sind, kann auch ein konnektionistisches System kein brauchbares Ergebnis ableiten. Ein zweiter Kritikpunkt ist die fehlende Introspektive. Konnektionistische Systeme stellen häufig eine *„Black Box“* dar. Die genauen Vorgänge in einem Netz bleiben weitestgehend verborgen bzw. sind oft nicht nachvollziehbar. Es ist z. B. sehr schwer nachzuvollziehen, was das System wirklich „weiß“ und wie dieses Wissen im Netz verteilt ist.
Trotz dieser Kritikpunkte sind konnektionistische Systeme geeignete Werkzeuge für die Analysephase innerhalb des Knowledge Discovery in Databases Prozesses.

Literaturverzeichnis

Berry / Linoff (1997)

Berry, M. J. A.; Linoff, G.: Data mining techniques: for marketing, sales and customer support. New York et al. 1997.

Biethahn (1998)

Biethahn, Jörg (Hrsg.): Betriebswirtschaftliche Anwendungen des Soft Computing. Braunschweig 1998.

Bigus (1996)

Bigus, J. P.: Data mining with neural networks: solving business problems – from application development to decision support. New York et al, 1996.

Chamoni (1999)

Chamoni, Peter: Ausgewählte Verfahren des Data Mining.
In: Chamoni / Gluchowski (1999), S. 355 – 373.

Chamoni / Gluchowski (1999)

Chamoni, Peter; Gluchowski, Peter (Hrsg.):
Analytische Informationssysteme – Data Warehouse, On-Line Analytical Processing, Data Mining. 2. Auflage Berlin et al. 1999.

Düsing (1999)

Düsing, Roland: Knowledge Discovery in Databases und Data Mining.
In : Chamoni / Gluchowski (1999), S. 345 – 353.

Haykin (1994)

Haykin, Simon: Neural Networks – A Comprehensive Foundation. New York 1994.

Hoffman (1993)

Hoffmann, Norbert: Kleines Handbuch neuronale Netze : anwendungsorientiertes Wissen zum Lernen und Nachschlagen. Braunschweig et al. 1993.

Kerling / Poddig (1994)

Kerling, Matthias; Poddig, Thorsten: Klassifikation von Unternehmen mittels KNN. In: Rehkugler / Zimmermann (1994), S. 427 – 490.

Kinnebrock (1994)

Kinnebrock, Werner: Neuronale Netze : Grundlagen, Anwendungen, Beispiele. 2. Auflage, München et al. 1994.

Nauck et. al (1996)

Nauck, Detlef; Klawonn, Frank; Kruse, Rudolf: Neuronale Netze und Fuzzy-Systeme - Grundlagen des Konnektionismus, Neuronaler Fuzzy-Systeme und der Kopplung mit wissensbasierten Methoden. 2. Auflage, Braunschweig et al. 1996.

Patterson (1996)

Patterson, Dan: Künstliche neuronale Netze : das Lehrbuch. 2. Auflage, München et al. 1996.

Rehkugler / Zimmermann (1994)

Rehkugler, Heinz; Zimmermann, Hans Georg (Hrsg.):
Neuronale Netze in der Ökonomie : Grundlagen und finanzwirtschaftliche Anwendungen. München 1994.

Rojas (1993)

Rojas, Raúl: Theorie der neuronalen Netze: Eine systematische Einführung. Berlin et al. 1993.

Scherer (1997)

Scherer, Andreas: Neuronale Netze: Grundlagen und Anwendungen. Braunschweig et al. 1997.