

**WIRTSCHAFTSUNIVERSITÄT WIEN**  
**INSTITUT FÜR INFORMATIONSWIRTSCHAFT**

Seminararbeit

**Data-Mining mit neuronalen Netzen und  
genetischen Algorithmen für BI**

**JOHANNES MAJER**

Betreuer:  
Priv.Doiz. Dr. Michael Hahsler

## Inhaltsangabe

In der Arbeit wird Data-Mining mit Hilfe von neuronalen Netzen und genetischen Algorithmen behandelt. Dem Leser wird zuerst eine Einführung in den Data-Mining Prozess vermittelt, bevor auf die zwei ausgewählten Methoden näher eingegangen wird. Es werden weiters einige bekannte Varianten der beiden Methoden erläutert und deren Vorteile bzw. Nachteile erklärt. In dem Kapitel „Trainieren von Neuronalen Netzen mit genetischen Algorithmen“ wird außerdem die gemeinsame Verwendung der beiden Methoden vorgestellt. Fallbeispiele runden die Kapitel zu den jeweiligen Methoden ab.

## Abstract

In this paper data-mining with the utilisation of neural networks and genetic algorithms will be explained. Before both methods are further elaborated, the data mining process will be discussed in detail. Then neural networks will be brought closer with emphasis on learning techniques. Also the different implementations of genetic algorithms will be examined and the pro and cons will be specified. Last but not least two case studies will show how both methods can be implemented.

**Keywords:** Data-Mining, Business Intelligence, Neuronale Netze, Genetische Algorithmen

# Inhaltsverzeichnis

<b>1</b>	<b>Einleitung</b>	<b>1</b>
<b>2</b>	<b>Grundlage von Data Mining</b>	<b>2</b>
2.1	Der Data Mining Prozess . . . . .	3
<b>3</b>	<b>Neuronale Netze</b>	<b>5</b>
3.1	Grundlage von neuronalen Netzen . . . . .	5
3.2	Anwendungsgebiete von künstlichen neuronalen Netzen . . . . .	7
3.3	Funktion des Menschlichen Gehirns . . . . .	7
3.4	Grundlage von künstlichen neuronalen Netzwerken . . . . .	8
3.5	Unterteilung von neuronalen Netzen . . . . .	10
3.6	Das Lernen von künstlichen neuronalen Netzen . . . . .	11
3.7	Kundenbewertung mit neuronalen Netzen . . . . .	13
<b>4</b>	<b>Genetische Algorithmen</b>	<b>14</b>
4.1	Einführung in Genetische Algorithmen . . . . .	14
4.2	Modifikation . . . . .	16
4.3	Trainieren von Neuronalen Netzen mit genetischen Algorithmen	18
4.4	Evolutionäre Algorithmen im Einsatz für Data Mining . . . . .	19
<b>5</b>	<b>Zusammenfassung</b>	<b>20</b>

# 1 Einleitung

Diese Arbeit beschäftigt sich mit dem Einsatz von Neuronalen Netzen und Genetischen Algorithmen für die Anwendung in Data Mining. Diese beiden Methoden sind in der Praxis nicht so weit verbreitet, bieten aber einen alternativen und spannenden Blickwinkel für das Auffinden von neuen Informationen. Diese beiden Methoden wurden ausserdem gewählt, weil sie nicht nur beide von der Biologie inspiriert sind, sondern weil sie sich auch sehr gut ergänzen. Beide Methoden haben wie aus der dem Literaturverzeichnis ersichtlich ist schon einen langen Weg hinter sich. Dementsprechend gibt es viele Modifikationen sowohl bei den Neuronalen Netzen als auch bei den Genetischen Algorithmen. Diese Abänderungen werden in beschränktem Umfang behandelt. Bei den neuronalen Netzen werden am Anfang auf die Grundlagen eingegangen, um später zu verstehen wie es dem Netzwerk möglich ist zu lernen. Eine ähnliche Vorgangsweise wurde auch bei dem Kapitel über Genetische Algorithmen gewählt. Zuerst wird erklärt wie das Konzept aus der Natur übernommen wurde und später werden konkrete Beispiele, welche hoffentlich die Genetischen Algorithmen verständlicher machen, behandelt. Kurz wird auch gezeigt, wie sich die beiden Methoden verbinden lassen und somit die Effizienz des Data Minings erhöhen.

Der zweite Kapitel dient als Einführung in Data Mining. Es werden die Vorteile, die sich durch die Anwendung ergeben näher erläutert. Danach wird der Data Mining Prozess behandelt.

Im dritten Kapitel werden die neuronalen Netze und ihre Anwendung untersucht. Es wird aufgezeigt wie sich die neuronalen Netze entwickelt haben und wie das biologische Vorbild funktioniert. Der Schwerpunkt des Kapitels sind die unterschiedlichen Varianten des Lernens von neuronalen Netzen.

Im vierten Kapitel gibt es eine Einführung in die genetischen Algorithmen. Es wird anhand eines Beispiels die Funktion erläutert und danach auf bekannte Varianten des Algorithmus eingegangen. Es folgt ein Abschnitt über die gemeinsame Verwendung von Neuronalen Netzen und genetischen Algorithmen. Den Abschluss des Kapitels bildet ein Anwendungsfall im Bereich von Marketing.

## 2 Grundlage von Data Mining

Mit der Möglichkeit der vermehrten Speicherung von Daten auf allen Gebieten des Daseins durch grössere Kapazitäten der Hardware, stellte sich immer mehr die Frage wie man diese Daten effizient aufbereiten kann um aus ihnen neue Informationen zu gewinnen bzw. um die vorhandenen Daten besser zu verstehen. Die Suche nach Informationen, welche wertvoll sind für das Unternehmen, wird durch den Einsatz von statistischen Methoden zum Auffinden von Mustern bewerkstelligt. Die Disziplin, die sich damit beschäftigt nennt sich Data Mining. Unterstützt wurde diese Entwicklung durch einen Druck auf die Unternehmen, die sich immer wieder neuen Umwelteinflüssen stellen müssen und zweitens einen hohen Druck durch die Konkurrenz abwehren müssen. Die Vorteile, die durch den Einsatz von Data Mining entstehen, liegen auf der Hand. Um nur beispielhaft zwei Einsatzgebiete zu erwähnen, gibt es erstens die Möglichkeit aus den Kundendaten, Umsatzzahlen und anderen Transaktionsdaten den Kunden eines Unternehmens besser zu verstehen und somit das Produkt an die Wünsche der Käufer anzupassen. Zweitens wie auch in (Big96, S. 6) erwähnt wird, kann der Absatz besser vorherbestimmt werden, wodurch wieder Einsparungen bei Lagerkosten und eine genauere Belieferung von Filialen ermöglicht wird. Data Mining bietet somit gute Gründe für dessen Einsatz. Durch die Masse an Daten ist auch nicht daran zu denken, dass diese allein durch menschliche Abstraktion in einer akzeptablen Zeitraum bearbeitet und Muster bzw. Informationen ausgelesen werden können.

Eine weiterer Wunsch der Unternehmen war immer den Kunden besser zu bedienen und ihm langfristig an das Unternehmen zu binden. Mit dem Einsatz von CRM Software konnte dies schon halbwegs gut bewerkstelligt werden. Der Kunde konnte nicht nur besser und schneller betreut werden (zum Beispiel durch „Sales Force Automation“), was zu einer Reduktion der aufgewendeten Ressourcen führt, sondern er wurde durch dieses Qualitätsmanagement auch dem Unternehmen treuer. Aber was den CRM-Systemen fehlte, waren Anwendungen (BI Tools), welche Entscheidungen im Unternehmen unterstützen. Zu den Business Intelligence Tools zählen Abfragemöglichkeiten, Reporting, OLAP, statistische Methoden und nicht zuletzt Data Mining (DH01, vgl. S. 4).

Die Frage, die sich nun stellt ist, wofür braucht man Data Mining. Der Unterschied zu Zeiten vor 50 Jahren ist, dass alles weniger überschaubar ist, weil sich die Grössen verändert haben. Heutzutage erzeugt ein Unternehmen meist viel unterschiedliche Produkte. Seine Kunden können über die ganze Welt verteilt sein. Es gibt viele Möglichkeiten um seine Zielgruppe anzusprechen und Geschäfte werden viel schneller abgewickelt. Manager haben durch die globalen Kommunikationsmedien auch nicht mehr die Zeit Entscheidungen lange zu überdenken, sonst kann es passieren, dass die Konkurrenz

schneller ist. Data Mining versucht in diesem Chaos dem Entscheidungsträger zu helfen, indem es 4 Arten von Problemen löst (DH01, vgl. S. 33) Ersten ist Data Mining in der Lage Verbindungen zwischen verschiedenen Entitäten zu erfassen (Linking Analysis). Zum Beispiel wurde eine Analyse von einem Supermarkt gemacht, welche Produkte von den Kunden gemeinsam gekauft werden. Dabei stellte sich heraus, dass Windeln und Bier gerne von jungen Vätern gegen Feierabend gekauft wurden. Durch diese Art der Warenkorbanalyse können auf den Weg zwischen den zwei Produkten verschiedene „cross-selling Strategien“ eingebaut werden und so der Umsatz des Supermarktes gesteigert werden. Zweitens können Alternativen evaluiert werden. Die Ressourcen in einem Unternehmen sind logischerweise knapp und sollten nur für Geschäftsmöglichkeiten verwendet werden, die besser sind als deren Alternativen. Drittens hat Data Mining die Funktion aus historischen Daten auf zukünftige Ereignisse zu schließen. Übliche Anwendungen sind das Schätzen von Bedarf eines Produktes bzw. ob ein neuer Kunde mehr Geld bringt als ein anderer. Letztens können auch Prozesse im Unternehmen evaluiert werden, indem man schaut wie sich die Effektivität verändern würde, wenn man bestimmte Faktoren beeinflusst.

## 2.1 Der Data Mining Prozess

Der Data Mining-Ablauf ist ein iterativer Prozess mit mehreren Phasen (DH01, siehe S. 34), (ZK03, S. 65). Am Anfang des Data Minings steht die Problemdefinition. Es werden Fragen definiert, welche das Data Mining-Modell später beantworten soll. Eine oft gestellte Frage könnte zum Beispiel lauten: Welche Charakteristika hat mein durchschnittlicher Kunde? Also wieviel Geld hat er zur Verfügung bzw. wofür gibt er es aus? Diese Fragen haben mehr einen erforschenden Charakter, wo hingegen die Frage nach zukünftigen Rahmenbedingungen mehr einen vorsagenden Modell entsprechen würde. In dieser Phase sollten auch die Ressourcen bereits aufgestellt werden, um im nächsten Schritt der Datenaufbereitung keinen Leerlauf zu produzieren. Die müssen in diesem Schritt so verändert werden bzw. repräsentativ sein, dass eine Stichprobe für die statistische Analysen die Verteilung der Grundgesamtheit wiedergibt. Hat man das geschafft braucht man zwei verschiedene Stichproben. Die erste wird verwendet um die Parameter von den Daten zu schätzen (also zum Beispiel Mittelwert, Standardabweichung und ähnliches). Daher wird die erste Stichprobe auch benötigt um das Modell anzupassen und zu trainieren (siehe neuronale Netze). Ist die Lernphase beendet, wird das Modell getestet mit den Daten aus einer anderen Stichprobe. Nun stellt sich die Frage wie man die Daten für die Analyse zur Verfügung stellt. Dabei bieten sich verschiedene Methoden an. Unabhängige Data Marts haben den Vorteil, dass die Daten verändert werden können. Sie beanspruchen dafür aber die Bandbreite von den Verbindungen. Ein anderer Weg Daten zur Verfügung zu stellen ist, diese nicht zu bewegen und das Data Mining in die

Datenbank einzubauen. Somit fehlt die Bewegung der Daten weg, aber der Nachteil sind meist proprietäre Formate der Anbieter von Datenbanken. Im nächsten Schritt werden die Daten gereinigt und es werden erste zur Verfügung stehenden Methoden überprüft. Danach wird im vierten Schritt die Methode festgelegt und die zuvor gestellten Fragen formalisiert. Diese Hypothesen dienen dann um den Ablauf der Testreihen zu verankern. In der nächsten Phase kommt dann unser eigentliches Modellerstellen mit den Trainingsdaten, die zuvor durch eine Stichprobe gezogen wurden. Hat man das Modell so weit angepasst wie erwünscht, beginnt die fünfte Phase mit der Evaluierung des Modells. Hier werden dann zum ersten Mal die Hypothesen überprüft und die Qualität der Aussagekraft ermittelt. Schliesslich wenn auch die Hürde gemeistert wurde, wird das System noch implementiert. Dieser Schritt beinhaltet das Schreiben von Dokumentationen und ebenfalls das Erstellen eines sinnvollen Interface damit der End-User das Werkzeug benutzen kann. Abschliessend bleibt noch zu erwähnen, dass der meiste Aufwand darin besteht die Daten zu reinigen bzw. aufzubereiten (je nach Literatur werden für diesen Schritt zwischen fünfzig und siebzig Prozent veranschlagt).

### 3 Neuronale Netze

Neuronale Netze, welche im Data Mining eingesetzt werden, sind informationsverarbeitende Systeme, die aus einer großen Anzahl einfacher Einheiten (Neuronen) bestehen, die sich Informationen in Form der Aktivierung der Zellen über gewichtete Verbindungen (connections, links) zusenden. Das Kapitel über neuronale Netze stellt grösstenteils eine Zusammenfassung der Bücher (Zel00) und (Big96) dar.

Die Faszination, die hinter der Erforschung der neuronalen Netze liegt, beruht auf der enormen Leistungsfähigkeit des menschlichen Gehirns, das man in seiner Struktur erfassen will und in künstlichen neuronalen Netzwerken für komplexe Anwendungen ausnützen möchte. Das Potential unseres Gehirns liegt nicht in der Verarbeitungsgeschwindigkeit der Informationen oder in der Genauigkeit der Operationen begründet, hier ist nämlich die Maschine dem menschlichen Gehirn bei weiten überlegen, sondern ist in der hochgradig parallelen Informationsverarbeitung, also der netzwerkartigen Struktur der Nervenverbände, zu suchen. Die Orientierung in ihrem Aufbau und ihrer Konzeption an der Funktionsweise des menschlichen Gehirns und das Abgehen von der Arbeitsweise konventioneller Rechner der klassischen von Neumann-Architektur ermöglicht es den neuronalen Netzen geistige Fähigkeiten des Menschen wie das Lernen aus Beispielen, das Verallgemeinern von Beispielen, das Abstrahieren, das schnelle Erkennen und Vervollständigen komplizierter Muster, das assoziative Speichern und Abrufen von Informationen etc. nachzubilden und zu simulieren. Neuronale Netze weisen viele positive Eigenschaften auf, die herkömmliche Ansätze nicht besitzen, dabei sind folgende von zentraler Bedeutung. Erstens die Parallelverarbeitung unter der man versteht, dass ein neuronales Netz die notwendigen Berechnungen nicht nacheinander durchführt, sondern gleichzeitig. Allerdings muss man beachten, dass diese Parallelverarbeitung bei der Simulation neuronaler Netze am Computer zur Zeit nur theoretischer Natur ist, da die meisten heutigen Recheneinheiten nur seriell arbeiten können, daher nur nacheinander durchgeführt werden können. Zweitens ist die verteilte Speicherung zu erwähnen. Bei Festplatten werden üblicherweise bestimmte Informationen an bestimmten Stellen der Platte gespeichert (lokale Speicherung). Im Gegensatz dazu werden Informationen in neuronalen Netzen im ganzen Netzwerk oder zumindest über einen Teil davon gespeichert. Drittens besteht eine Lernfähigkeit, welche es intelligenten Lebensformen ermöglicht aus Erfahrungen zu lernen und das Gelernte auf neue, aber ähnliche Situationen und Ereignisse anzuwenden (Zel00, vgl. S.23).

#### 3.1 Grundlage von neuronalen Netzen

Die Geburtsstunde für die Verbreitung von künstlichen neuronalen Netzen geht auf das Jahr 1943 zurück (Zel00, vgl. S. 28). In diesem Jahr beschrie-

ben Warren McCulloch und Walter Pitts in ihrem Aufsatz „A logical calculus of the ideas immanent in nervous activity“ (MP88) Netzwerke, welche diese wie beim menschlichen Gehirn in kleinen Einheiten, genannt Neuronen, unterteilten. Weiters wurde dargestellt, dass selbst primitive neuronale Netze in der Lage sind jegliche Art von arithmetischen und logischen Funktionen zu berechnen. Der nächste für die Weiterentwicklung der neuronalen Netze entscheidende Entwicklungspunkt war im Jahre 1949 als Donald O. Hebb in seiner Arbeit „The Organization of Behaviour“ (Heb49) feststellt, wie sich Neuronen verknüpfen und verstärken lassen. Die Hebbsche Lernregel ist von großer Wichtigkeit für die Entwicklung von neuronalen Netzen, da diese es erst ermöglichte neuronalen Netzen lernen zu lassen um Muster zu erkennen. Die erste Implementierung wurde kurze Zeit später am MIT durchgeführt. Dieser erste Prototyp eines neuronalen Netzwerks wurde hauptsächlich für die Erkennung von Mustern eingesetzt. Das Netzwerk war bereits in der Lage mit Hilfe eines 20x20 grossen Sensors auf Papier gedruckte Zahlen zu erfassen. Jedes Gewicht wurde dabei von einem motorgetriebenen Potentiometern vertreten um die Gewicht anpassen zu können. Sechs Jahre davor hatte Marvin Minsky es bereits geschafft die Gewichte seiner Konstruktion automatisch anpassen zu lassen. Jedoch wurde es nie in der Praxis übergeführt. Des Weiteren erlangten in dieser Zeit (zwischen 1955 und 1969) viel Wissenschaftler wie zum Beispiel Oliver Selfridge, Karl Steinbuch oder Bernard Widrow Ruhm und Anerkennung und leisteten erhebliche Beiträge im Bereich neuronaler Netze. Ein Standardwerk der neuronalen Netze heisst „Perceptrons“ (MP69). Es stammt von Marvin Minsky und Seymour Papert und legte eine Behauptung dar, dass das Perzeptron (entwickelt von Frank Rosenblatt) doch nicht in Lage ist alles zu repräsentieren. Die Konsequenz, dass auch leistungsfähigere Netzwerke als das Perzeptron nicht in der Lage sein würden, die Probleme (xor-Problem) zu lösen (was heute widerlegt ist), läutete eine Periode der Stagnation auf dem Forschungsgebiet der neuronalen Netze ein, was aber nicht bedeutete, dass wesentliche Grundlagen in den folgenden fünfzehn Jahren gelegt wurden. Hierbei traten John Hopfield (Hopfield-Netze) und Teuvo Kohonen, bekannt durch seine selbstorganisierenden Karten, hervor (Lip).

Erst Mitte der 80er Jahre erfuhren die neuronalen Netze eine Wiederbelebung, die oft mit dem Artikel „Neural Computation of Decisions in Optimization Problems“ (HT85) von John Hopfield in Verbindung gebracht wird, in dem er zeigt, wie Hopfield-Netze schwierige Optimierungsaufgaben lösen können. Ein weiterer Schritt zur Wiederbelebung der Neuronalen Netze wurde mit der Publikation über das Lernverfahren Backpropagation (RHW86) 1986 durch Rumelhart, Hinton und Williams geleistet. Mit dem Lernverfahren Backpropagation stand im Vergleich zu den bisher entwickelten Verfahren ein sehr schnell und robuster Lernalgorithmus für mehrstufige feedforward-Netze zur Verfügung, das sich mathematisch elegant als Gradientenabstiegsverfahren des Netzwerkfehlers herleiten ließ. Seit circa 1986 hat sich die-

ser Forschungsbereich geradezu explosionsartig entwickelt und wird in vielen Wissenschaftsgebieten verwendet.

### 3.2 Anwendungsgebiete von künstlichen neuronalen Netzen

Neuronale Netze besitzen eine Reihe von unterschiedlichen Anwendungen. Dabei unterscheidet man grundsätzlich zwischen drei großen Anwendungsbereichen. Erstens der Einsatz von Neuronalen Netzen, die modelliert werden, um die Funktionsweise des menschlichen Gehirns besser kennen zu lernen. Bis vor wenigen Jahren war die Modellierung des menschlichen Gehirns die eigentliche Motivation für die Erforschung von künstlichen neuronalen Netzen. Man versuchte, künstliche neuronale Netze zu entwickeln, die in möglichst vielerlei Hinsicht dem biologischen Vorbild entsprechen, so dass durch die Simulation der Modelle Rückschlüsse auf noch ungeklärte Eigenschaften des biologischen Systems möglich wurden. Zweitens für die theoretischen Anwendungen im Bereich der Physik um chaotische Systeme möglichst gut nachzubilden. Und drittens wie im Data Mining werden künstliche neuronale Netze dazu verwendet um Prognosen für die Zukunft zu fällen (SNG90, vgl. S. 143).

### 3.3 Funktion des Menschlichen Gehirns

Die Neuronen sind Körperzellen, welche sich aber von den anderen Zellen gewaltig abheben durch das Aussehen ihrer Form, der Art der Zellmembran und der Eigenschaft, an den "Extremitäten" der Zellen Synapsen bilden zu können, die mit anderen Nerven- und auch Muskelfasern eine Verbindung aufbauen um Nervenimpulse weiterleiten zu können. Im Unterschied zu anderen Zellen haben Neuronen nach Beendigung ihrer Entwicklung nicht mehr die Fähigkeit sich zu vervielfältigen.

Ein Neuron setzt sich zusammen aus dem Zellkörper, welcher Soma genannt wird, aus dem Axon, einer Nervenleitung, die das Signal/Impuls aus dem Kern weiterleitet und aus den Dendriten. Der pyramidenförmige Zellkörper enthält die für den Zellstoffwechsel lebensnotwendigen Strukturen und Substanzen und umschließt den Zellkern (Nucleus), welcher der Träger der Erbinformation ist. Die Dendriten sind dünne, verzweigte Fortsätze, die die elektrischen Nervensignale aufnehmen. Sie befinden sich beim Zellkörper. Die Weiterleitung dieser Impulse erfolgt über die normalerweise viel längere Nervenfasern mit einer Geschwindigkeit von über 120m/s. Es unterscheidet sich von den Dendriten in der Struktur und den Eigenschaften der Membran. Am anderen Ende verzweigt es sich und bildet eine Verdickung, die sogenannte Synapse. Diese Synapsen sind die Verbindungsstellen zwischen den Enden des Axons eines Neurons und den Dendriten eines anderen Neurons und spielen für das Lernen eine zentrale Rolle (Ano).

Synapsen bestehen aus einer präsynaptischen Membran, an der die Impulse ankommen, einer postsynaptischen Membran, die zur Weiterleitung der Impulse dient, und aus einem mit Flüssigkeit gefüllten Raum zwischen den zwei Membranen, dem synaptischen Spalt. Die elektronischen Nervenimpulse führen an der präsynaptischen Membran zur Freisetzung eines chemischen (erregend oder hemmend wirkenden) Botenstoffes, eines Neurotransmitters, in den synaptischen Spalt. Diese Überträgersubstanz bindet sich an spezifische Rezeptoren, die in der postsynaptischen Membran sitzen. Diese Bindung bewirkt eine Veränderung der elektronischen Aktivität der postsynaptischen Zelle. Die Informationsübertragung an der Synapse kann nur in eine Richtung erfolgen, nämlich vom Axon eines Neurons zum Dendriten eines anderen Neurons. Der umgekehrte Weg ist ausgeschlossen. Man kann zwischen zwei Arten von Synapsen unterscheiden: den erregenden und den hemmenden Synapsen. Beim erregenden Synapsentyp erhöht der in den synaptischen Spalt ausgeschüttete Neurotransmitter die Wahrscheinlichkeit, dass die postsynaptische Zelle aktiv wird. Beim hemmenden Synapsentyp wird die Wahrscheinlichkeit, dass die postsynaptische Zelle aktiv wird, herabgesetzt. Je größer die Aktivität der präsynaptischen Zelle ist, desto stärker wird die postsynaptische Zelle gehemmt (BS96, S. 101).

Das Lernen wird wesentlich durch die Modifikation der Verbindung zwischen den Neuronen, also durch die Veränderung ihrer Synapsen, ermöglicht. Die Stärke der Verbindung von Neuronen hängt von den zwischen ihnen ausgebildeten Synapsen ab. Einige Einflussfaktoren für die Stärke der Verbindung sind die Anzahl und die Größe der Synapsen sowie verschiedene Details in ihrem Aufbau wie Menge und Art der gespeicherten Transmittersubstanz, Zahl der Rezeptoren für den Transmitter etc. Eine Veränderung dieser Parameter führt zu Veränderungen im Signalfluss und bewirkt letztendlich eine Verhaltensänderung des Organismus.

### 3.4 Grundlage von künstlichen neuronalen Netzwerken

Bei künstlichen neuronalen Netzen wird versucht durch Nachahmen das biologische Gehirn zu simulieren. Natürlich ist dieses Modell stark vereinfacht und besitzt nur einen Aufbau, der sich zusammensetzt aus einer Inputschicht und einer Outputschicht und Kanten, welche diese beiden verbinden. Dies wäre die primitivste Form von einem neuronalen Netzwerk. Meist haben die Netze deshalb noch mehrere Zwischenschichten, mit welchen von aussen meist nicht interagiert wird, sondern nur über ein sendendes Neuron der ersten Schicht (SNG90).

Der Ablauf des Netzwerkes läuft anders als im Gehirn meist in eine Richtung ab. Zuerst wird die Inputschicht mit Eingabewerten versorgt. Im nächsten Schritt wird die nächste Ebene aktiviert. Dies dauert solange bis die letzte Schicht erreicht ist. Durch Rückkopplung können die Werte aber auch wieder

an eine frühere Schicht zurückgegeben werden oder es kann eine Verbindung bestehen, welche eine Ebene überspringt. Die Ebenen sind wiederum durch ein oder mehrere Neuronen aufgebaut. Neuronen sind die kleinste Einheit von neuronalen Netzen.

Die Verknüpfung kann sich auf eine Ebene beschränken, sodass Neuronen einer Schicht verknüpft werden oder mehrere Schichten miteinander verbinden. Man unterscheidet hauptsächlich zwischen vollständiger, korrespondierender und zufälliger Verknüpfungen der Neuronen. Bei der vollständigen Verknüpfung wird der Output von jedem Neuron der vorherigen Ebene mit dem Input der nachhergehenden Ebene verbunden und für jedes Neuron in dem neuronalen Netzwerk. Bei der korrespondierenden Verknüpfung wird zum Beispiel jeweils das fünfte Element einer Ebene miteinander verbunden. Die letzte Verknüpfungsart, wie der Name schon vermuten lässt, ist zufällig. Nur durch die gegebene Wahrscheinlichkeiten werden Neuronen miteinander verbunden oder nicht.

Die Kanten, welche zwei Neuronen miteinander verbinden und über die Daten übertragen werden, wird ein Gewicht zugeordnet. Das Kantengewicht ist ähnlich der biologischen Leitfähigkeit in einem Gehirn. Man unterscheidet bei den Gewichten nach festen und flexiblen. Feste Kantengewichte bleiben gleich und sind unabhängig von dem Lernprozess des neuronalen Netzwerks. Hingegen verändern sich die flexiblen Kantengewichte je nach Lernrate verschieden schnell.

Die Kantengewichte können sowohl positiv als auch negativ sein. Bei positiven Werten wird die Weitergabe an das empfangene Neuron bestärkt, wohingegen bei negativen Wert die Weitergabe gehemmt wird.

Nachdem die Verbindungen ausführlich geklärt wurden, wird der Rest des Abschnitts sich mit den Funktionen der einzelnen Neuronen beschäftigen. Neuronen besitzen im Wesentlichen drei Funktionen, welche nach der Reihe durchlaufen und erklärt werden. Die erste Funktion verändert die Eingangswerte je nach der verwendeten Funktion. Meist handelt es sich dabei um eine Summenfunktion, welche die eingehenden Werte gemäss ihren Kantengewichten aufsummiert. Weitere Funktionen sind die Maximumfunktion und die Multiplikationsfunktion. Im zweiten Schritt wird der errechnete Wert an eine Aktivierungsfunktion übergeben (Zel100, S. 83). Die Funktion berechnet die Aktivität, die ein Neuron in Abhängigkeit von dem aktuellen Input annehmen soll. Die Aktivität ist eine Funktion der gewichteten Inputsumme, des Schwellenwerts (= Wert, den die Summe aller gewichteten Eingänge eines Neurons überschreiten muss, damit es aktiv wird) und der bisherigen Aktivität. Aus ihr wird der Wert des Ausgangssignals ermittelt. Die einfachste Transferfunktion ist linear, was bedeutet, dass die Summe aller gewichteten

Eingänge der Aktivität des Neurons entspricht. Im dritten und letzten Schritt wird eine Outputfunktion verwendet um die Neuronen innerhalb einer Ebene zu verknüpfen. Mit zum Beispiel der Funktion (der Sieger kriegt alles), bei der nur der höchste Wert weitergegeben wird, geben alle anderen Neuronen lediglich ein Null weiter. Auch hier gibt es noch einige andere Funktionen, die aber wegen ihrer mangelnden Bekanntheit auch hier nicht näher erläutert werden sollen.

### 3.5 Unterteilung von neuronalen Netzen

Im Lauf der Geschichte der neuronalen Netze haben sich diverse Netzwerktypen entwickelt, die auf sehr unterschiedlichen Motivationen und Zielsetzungen fußen. Die Entwicklung des Hopfield-Netzwerkes und der Boltzmann-Maschine wurde beispielsweise durch physikalische Modelle inspiriert, während viele andere Netzwerke eher biologische oder psychologische Grundlagen haben. Häufig lag der Entwicklung eines Netzwerkes die Frage nach einer geeigneten Lernregel zugrunde.

Die Unterteilung von den Netzwerken kann unter verschiedensten Möglichkeiten durchgeführt werden. Auswahlkriterien wären zum Beispiel ob in den Netzen eine Rückkopplung erlaubt ist, nach Tiefe der Ebenen, nach Anzahl der Verbindungen und aufgrund der verschiedenen Funktionen in den Neuronen. In diesem Kapitel wird die Unterteilung anhand der Rückkopplungen besprochen. Dabei gibt es folgende Unterscheidungen. Erstens gibt es Netze in den die Rückkopplung gänzlich verboten ist (feedforward Netze). Daher bewegen sich die Werte immer von der Inputschicht Richtung Outputschicht. Es wird keine Ebene zweimal durchlaufen. Aber es kann vorkommen, dass die Werte eine Ebene überspringen. Zweitens gibt es Netze, welche dies erlauben. Dabei handelt es sich um Netzwerke mit zum Beispiel direkter Kopplung. Daher ist das Neuron mit sich selbst verbunden und kann sich auf dies Art und Weise selbst verstärken. Im Unterschied dazu gibt es die sogenannten Netzwerke, bei denen nur über eine andere Ebene die Rückkopplung bestehen kann. Damit werden gleich mehrere Neuronen bestärkt. Als letztes sollen noch die Netze erwähnt werden, bei denen es eine Verbindung zwischen den Neuronen einer Ebene geben kann. Diese funktionieren nach dem Prinzip, dass alle miteinander in einer Schicht verbunden sind und nur der stärkste ist dann in der Lage seinen Wert an die nächste Ebene zu übergeben (dieses Prinzip haben wir schon bei der Behandlung von Outputfunktionen der Neuronen behandelt und sie hier nur der Vollständigkeit hier erwähnt).

Das nächste Kapitel zeigt, welche Methoden es gibt, damit künstliche neuronale Netze lernen und somit zum Beispiel vorhersage Kraft entwickeln können, um im Data Mining-Prozess eingesetzt zu werden.

### 3.6 Das Lernen von künstlichen neuronalen Netzen

Lernen bezeichnet einen Prozess bei dem auf Grund von Funktionen zum Beispiel die Gewichte jeder Kante in einem Iterationsschritt so angepasst werden bzw. verändert werden, dass der Output des neuronale Netzwerk gleich dem vorgegebenen Ergebnis entspricht. Dieser Prozess kann wie später beschrieben durch die Implementierung von genetischen Algorithmen beschleunigt beziehungsweise effizienter gestaltet werden. Aber was das neuronale Netzwerk von zum Beispiel statistischen Methoden unterscheidet ist, dass diese sich an die Struktur mehr oder weniger iterativ selbst anpassen (Zel00, vgl. S. 87).

Wie schon oben erwähnt ist die Veränderung von Kantengewichten eine Methode des neuronalen Netzes sich an die Struktur anzupassen. Andere Methoden ist das Bilden von neuen Verbindungen, das Auslösen von bestehenden Verbindungen, die Veränderung des Wertes in den Neuronen, die Veränderung der verschiedenen Funktionen (wie zum Beispiel der Ausgabefunktion) und die Entstehung bzw. Löschung von Neuronen. Meist wird in der Literatur nur die Abweichungen von den Kantengewichten besprochen. Die anderen Methoden fristen eher ein Schattendasein.

Bei den künstlichen neuronalen Netzen lässt sich zwischen drei Arten des Lernens unterscheiden (Zel00, vgl. S. 93). Erstens gibt es das überwachte Lernen (supervised Learning). Dabei wird von aussen dem Netzwerk nach jeder Iteration der gewünschte Ausgabewert übergeben und dadurch kann es seine Kantengewichte so verändern, dass es mit steigender Anzahl der Werte sich dem Output nähert. Das Ende ist erreicht, wenn das Netzwerk in einem ausreichenden Ausmass selbst von den Eingabedaten mit nur einem geringen Fehler auf die Outputwerte schliessen kann. Man spricht dann davon, dass das Netzwerk generalisiert ist. Es werden dabei folgende Schritte durchlaufen (als Beispiel wird ein Backpropagationnetzwerk genommen) (RHW86). Im ersten Schritt wird ein Eingabemuster von den Neuronen der ersten Schicht aufgenommen. Über die Kanten laufen die Werte durch das Netzwerk und generieren einen Output. Dieser wird mit dem gegebenen Outputwert der Trainingsdaten verglichen und die Abweichung bewirkt beim Rückwärtsdurchlaufen des Netzwerkes, dass sich die Kantengewichte dementsprechend verändern bzw. anpassen.

Die zweite Gruppe des Lernen nennt sich bestärkendes Lernen (reinforcement Learning) Diese Form ist eine weniger überwachte Form der Lernens. Dem neuronalen Netzwerk wird nicht nach jeder Lernphase der gewünschte Output übergeben, sondern es wird höchstens der Grad der Abweichung übermittelt. Dies hat zur Folge, dass es deutlich länger dauert bis das neuronale Netzwerk das Muster einigermaßen richtig erkennt. Denn das Netzwerk erkennt nicht wie es die Gewichte am besten verändern muss um zu dem richtigen Ergebnis zu kommen. Oft wird aber auch nur die Ausgabe bestätigt oder abgelehnt.

Die dritte Methode heisst unüberwachtes Lernen (Unsupervised Learning oder auch Self-Organized Learning). Dabei wird dem Netzwerk keine Informationen über den Output gewährt, sondern es versucht die Inputwerte in Klassen gleicher Werte zu speichern. Als Beispiel seien die Selbstorganisierenden Karten von Kohonen erwähnt.

Eine bekannte und weit verbreitete Regel für das Lernen von neuronalen Netzen stammt von Hebb. Es gibt viele Modifikationen, auf welche hier aber nicht näher eingegangen werden soll. Aber der Grundgedanke der Regel sei hier näher erläutert. Er besagt, dass je öfter zwei Zellen miteinander interagieren, umso öfter werden sich auch diese in Zukunft miteinander interagieren. Umgesetzt auf künstliche neuronale Netze bedeutet es, dass die Kantengewichte umso stärker werden je mehr die Knoten miteinander in Verbindung stehen. Die Abänderung der Kantengewichte ergibt sich dabei aus einer konstanten Rate für das Lernen mal der Aktivierungsgrösse von dem zweiten Neuron mal der Ausgabe des ersten Neurons.

Als letzte Lernregel sei noch die Delta-Regel besprochen. Hier wird eine Variable mit der Bezeichnung Fehlersignal eingeführt. Diese errechnet sich durch die Differenz zwischen gewolltem und tatsächlichem Aktivitätswert des Neurons. Sollte der gewollte Wert höher liegen als der tatsächliche werden die Kantengewichte zwischen den dem ausgebenden und dem aufnehmenden erhöht, natürlich nur unter der Bedingung, dass der Output der ausgebenden Zelle positiv ist. Ist dies nicht der Fall wird das Gewicht verkleinert. Im zweiten Fall, wo der gewollte gleich dem tatsächlichen Wert ist, bleibt alles unverändert. Im letzten Fall, wo der gewollte kleiner ist, wird genau das Gegenteil von dem ersten Fall gemacht. Somit werden bei positiven Werten die Verbindungen verringert und bei positiven Input vergrössert. Die Formel, in die man dieses errechnete Fehlersignal einsetzt, lautet ähnlich wie bei der Regel nach Hebb. Nämlich wird die Gewichtung errechnet durch die Multiplikation von der Lernrate mit dem Fehlersignal und dem beobachteten Wert. Ausserdem wird noch erreicht, dass die Veränderung der Kanten ähnlich hoch ist wie der Fehler. Bei den aussendenden Neuronen wird auch noch die Kantengewichtung je nach Beitrag zu dem Fehler stärker oder schwächer verändert. Die Höhe des Lernparameters gibt noch an, wie sehr es möglich ist das sich die Kantengewichte pro Iteration verändern können.

Immer muss aber im Auge behalten werden, dass das neuronale Netzwerk nicht zu sehr an die Trainingsdaten angeglichen wird. Man spricht dann von Overfitting. Darunter leidet die Fähigkeit des Netzwerkes das Muster zu erkennen und es werden mehr oder weniger nur die Eingabewerte verinnerlicht. Bei neuen Inputdaten hat das Netzwerk das Problem, dass es nicht abstrahieren kann und so sinkt die Qualität der Ausgabe. Es ist daher wichtig eine Balance zu finden zwischen zu viel Lernen und zu wenig.

### 3.7 Kundenbewertung mit neuronalen Netzen

Am Fallbeispiel der Kundenbewertung (Big96, vgl. S. 155) wird der Einsatz von neuronalen Netzen für einen Druckerbetrieb in der Praxis gezeigt. Bei der Bewertung von Kunden stellt sich unter anderem die Frage, welche Faktoren charakterisieren einen guten Kunden. Die Frage nach dem Kunden mit der höchsten Rentabilität würde sich mit einer SQL-Abfrage beantworten lassen. Aber bei der Beantwortung der ersten Frage sind neuronale Netze hilfreich.

Als erstens müssen geeignete Attribute aus der Kundendatenbank ausgewählt werden. Im konkreten Fall sind dies: die Jahre des Bestehens des Kundenunternehmens, Anzahl der Mitarbeiter, Sparte, Umsatz, durchschnittl. Anzahl der Aufträge pro Jahr, durchschnittl. Umsatz und Profit pro Auftrag. Als erwarteter Output wird die Rentabilität pro Jahr berechnet und in vier Untergruppen unterteilt. Anschliessend wird das Netzwerk trainiert. Nach einer akzeptablen Erkennungsrate von 90,2 Prozent wird das Training beendet. Das Netzwerk ist nun in der Lage für das Unternehmen profitable Kunden zu erkennen und kann so zum Beispiel helfen Marketingkosten zu sparen. Mit einer Sensitivitätsanalyse könnte weiters die einzelnen Parameter erforscht werden und zusätzliches Wissen für „Decision Support Systems“ generiert werden.

## 4 Genetische Algorithmen

### 4.1 Einführung in Genetische Algorithmen

Das folgende Kapitel beschäftigt sich mit genetischen Algorithmen. Genauso wie Neuronale Netze liegt der Ursprung der Methode in der Biologie. Nicht nur durch diesen Zusammenhang, sondern auch durch die Verwendung von genetischen Algorithmen für das Training von Neuronalen Netzen bietet sich eine tiefere Behandlung dieses Themas an. Grundlage für dieses Kapitel stellen unter anderem die Bücher (Lar06) und (BL04) dar.

Genetische Algorithmen entstanden in den 60er Jahren und wurden entwickelt um Ereignisse in der Biologie näher zu erforschen. Dabei handelte es sich zum Beispiel um Entwicklung von Populationen und im speziellen um die Erforschung der Weitergabe von bestimmten Merkmalen über die Gene. Wie in (BL04, S. 422) dargestellt, werden die Algorithmen verwendet um optimale Lösungen von Problemen zu finden. Für die leistungsfähigsten Computer ist es nicht möglich in angemessener Zeit alle Kombinationen miteinander zuvergleichen und die beste Lösung auszuwählen. Genetische Algorithmen versuchen den enormen Aufwand durch Vergleich aller Möglichkeiten zu umgehen und suchen optimale Lösungen durch evolutionäres Vorgehen. Ähnlich wie bei der Evolution durch Selektion und Mutation neue anpassungsfähigere und stärkere Individuen entstehen und sich dadurch mehr vermehren beziehungsweise ihre Überlebenschancen verbessern, wird auch bei genetischen Algorithmen Selektion, Mutation und Kreuzung auf Lösungen angesetzt. Bei der Selektion wird durch eine Funktion, genannt Fitnessfunktion, die Lösung ausgewählt, welche die beste Lösung liefert. Danach werden bei der Kreuzung die beiden selektierten Lösungen auf atomarer Ebene neu kombiniert und es entstehen zwei unterschiedlich von den Ausgangslösungen, neue Kinderlösungen. Diese besitzen wie bei der biologischen Fortpflanzung die Merkmale, welche sich von beiden Eltern durchgesetzt haben. Die dritte Methode bei genetischen Algorithmen ist die Mutation. Wie auch in der Natur kommt sie deutlich weniger oft zur Verwendung als die beiden anderen Methoden. Aber sie hat eine wichtige Aufgabe. Durch die rein zufällige Entstehung der Lösung mit geringeren Merkmalsausprägungen von den Eltern, wird der Umfang an Lösungen erweitert. Dadurch kann es vermieden werden, dass es zu schnell zu einfachen, lokalen Lösungen kommt.

Um zu optimalen Lösungen zu kommen, weisen genetische Algorithmen folgenden Ablauf auf (MCM83). Zuerst benötigt man eine erste Generation, welche aus einer Anzahl von beliebigen Parameterkombinationen besteht. Diese Lösungen werden binär codiert. Am Anfang des Algorithmus muss weiter festgelegt werden die Rate für Kreuzungen und eine Wahrscheinlichkeit für Mutationen. Wobei die Wahrscheinlichkeit für Kreuzungen deutlich höher liegen sollte als die von Mutationen. Die Fitnessfunktion muss natürlich gewisse

Begrenzungen aufweisen um unmögliche Lösungen, welche durch Mutationen aber auch Kreuzungen entstehen können, zu verhindern. Im zweiten Schritt werden die einzelnen Lösungen dieser Generation durch die Fitnessfunktion berechnet und evaluiert. Unter Evaluierung versteht man in diesem Zusammenhang die Zuordnung von Wahrscheinlichkeiten über das Vorkommen in der nächsten Generation. Diese ist abhängig von der errechneten Fitness der Lösung. Unterschiedlich zu dem Vorbild in der Natur, bleibt dabei die Anzahl der Population/Lösungen immer gleich. Aufgrund der Fitness werden dann Paare gebildet. Diese Paare werden dann mit der Wahrscheinlichkeit der Kreuzung, welche am Anfang unseres Prozesses definiert wurden, gekreuzt oder wenn ein Paar nicht zur Kreuzung ausgewählt wurde, werden einfach zwei Kinder erzeugt. Diese entsprechen Kopien der Eltern. Im gleichen Schritt wird auch mit der definierten Wahrscheinlichkeit für Mutationen bei manchen Kindern diese Mutation durchgeführt. Nachdem sich die neue Generation entwickelt hat und die alte Generation ersetzt hat, wird auf eine Abbruchbedingung geprüft. Diese Bedingung kann zum Beispiel die Differenz des Mittelwertes von der alten Generationen mit dem Mittelwert der neuen Generation darstellen. Wenn die Veränderung nur mehr gering ausfallen, so dass man keine andere Lösung mehr erwarten kann, ist es sinnvoll den Prozess zu beenden.

Zur besseren Veranschaulichung wollen wir ein Beispiel in Anlehnung an (Lar06, vgl.S. 243) näher betrachten. Es wird dabei versucht den maximalen Wert einer Normalverteilung  $N(16,4)$  zu finden. Mit der unteren Formel für die Normalverteilung werden Werte für die erste Generation berechnet. Die X-Werte können Werte zwischen 0 und 31 annehmen und werden dementsprechend binär codiert von 00000 bis 11111.

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Es werden also 5 X-Werte, welche durch einen Zufallsgenerator ausgewählt werden als Startgeneration verwendet. Weiters werden die Kreuzungsrate auf 0.75 und die Mutationsrate auf 0.002 festgelegt. Durch Einsetzen unseres X-Wertes in die Funktion erhalten wir unsere Fitness für die jeweilige Lösung. Für einen X-Wert von 9 erhalten wir eine Fitness von 0.021, wohingegen wir für einen Wert wie 31 nur eine Fitness von 0.00088 bekommen. Diese Fitness wird noch durch die Summe der anderen Fitnesswerte dividiert um zu der Selektionswahrscheinlichkeit zu kommen. Es zeigt sich, dass eine höhere Fitness, die Wahrscheinlichkeit für Selektion erhöht. Im nächsten Schritt wird zufällig ein Paar ausgewählt, wobei durch die Selektionswahrscheinlichkeit in unseren Fall die 9 mit höherer Wahrscheinlichkeit Teil eines Elternpaares wird als die 31. Nehmen wir trotzdem an, dass beide als ein Elternpaar gezogen werden. 9 ist binär mit 01001 und 31 mit 11111 codiert. Durch den

weiteren Einsatz von einem Zufallsgenerator wird erstens festgelegt ob es zu einer Kreuzung kommt, wobei die Wahrscheinlichkeit ziemlich gut steht mit 75 Prozent, und zweitens wird die Stelle gewählt an der sich die beiden Zahlen im binären Zustand kreuzen. Bei einer Kreuzung der beiden Zahlen an der Stelle zwei würden als Kinder die Lösungen 11 (01011) und 25 (11101) entstehen. Es kann natürlich, wie vorher erwähnt auch der Fall eintreten, dass ein Elternpaar nur aus einer gleichen Doppellösung besteht. Wäre zum Beispiel 9 zweimal im Elternpaar enthalten, würden die Kinder beim Kreuzungsvorgang gleich dem Elternpaar sein. Nur durch den Mutationsschritt könnte sich eine andere Lösung ergeben. Die Mutation hat zwar keine hohe Wahrscheinlichkeit, würde sie aber zur Anwendung kommen, würde sich ein zufällig gewähltes Bit der Kinderlösungen einfach invertieren. An dieser Stelle wird ein nächstes Paar aus der gesamten Generation gezogen. Dieser Vorgang wiederholt sich so lange bis die Anzahl der Kinder genau der Anzahl der Elterngeneration entspricht. In unserem Fall erzeugen wir sechs Kinder bis es zum Abbruch kommt. Somit muss eins zufällig wieder fallen gelassen werden um auf die gewünschte Population von fünf Kindern zu kommen. An diesem Punkt ersetzt die neue Generation die alte. Dann werden wiederum die Fitnesswerte errechnet und geprüft in wie weit der alte Mittelwert der Lösungen sich vom neuen unterscheidet. Hier entscheidet dann ein vorher definierter Differenzwert, ob noch eine Iteration nötig ist oder die Genauigkeit schon das angestrebte Niveau erreicht hat.

Was in dem Beispiel noch fehlt, ist die Behandlung von Einschränkungen. Wir haben nur Werte zwischen 0 und 31 gewählt. Wäre die Binärdarstellung länger, könnten sich auch andere Werte ergeben. Deshalb sollte man die Eingabe der Fitnessfunktion überprüfen und fehlt dieser Wert in einen Bereich, der nicht gewünscht wird, kann man diesen über Beschränkungen der Funktion eliminieren indem man für den Eingabewert einfach einen Minimalen Wert (in dem Beispiel von oben wäre das 0) ausgibt. Somit wird dieser Wert sogleich bei der Selektion verworfen.

## 4.2 Modifikation

In der Literatur lassen sich zahlreiche Abwandlungen von dem oben beschriebenen Algorithmus finden. Besonders die Wahl der Selektionsmethode (Lar06, S. 245) entscheidet das Ergebnis des genetischen Algorithmus sehr. Es muss dabei versucht werden ein Mittelmaß zwischen Diversifikation und einer fitnesslastigen Methode zu finden. Bei einer zu hohen Diversifikation ergibt sich nämlich das Problem, dass es sehr viele Iterationen braucht um auf ein akzeptables Ergebnis zu kommen. Demgegenüber steht das Problem von Methoden, welche die Selektionswahrscheinlichkeit zu stark von der Fitness abhängig machen. Das Problem, welches dabei entsteht, nennt sich „crowding“. Durch die hohe Fitness von einzelnen Lösungen, kommt es dazu, dass nur

noch diese „Alpha“-Lösungen sich reproduzieren und andere Lösungen nicht mehr beachtet werden. Somit ist es nahezu unmöglich in andere Bereiche einer Funktion vorzudringen und es ergeben sich möglicherweise somit nur lokale Optima (nur günstige Startwerte wären somit in der Lage das globale Optimum zu erreichen). Als Lösung wurde unter anderem (DG89) eine fitness-sharing function vorgeschlagen. Dabei werden Ähnlichkeiten von einzelnen Vertretern von einer Generation zueinander bestraft. Das heisst, je näher die Werte der beiden Lösungen übereinstimmen umso mehr wird den beiden Mitglieder von ihrem Wert abgezogen. Drei weitere in der Literatur oft wiederkehrende Ansätze seien der Vollständigkeit halber noch erwähnt. Erstens die Methode des „sigma-scaling“. Diese wurde von Forrester vorgeschlagen (For85). Der Hintergedanke ist dabei den Selektionsdruck über die Zeit fast gleich zu halten. Dies geschieht indem am Anfang, wo natürlich die Unterschiede zwischen den einzelnen Kandidaten noch relativ groß sind, bei dominanten Lösungen ihre Selektionsvorteile zu verkleinern. Im Gegensatz dazu werden gegen Ende die Unterschiede nicht mehr allzu ins Gewicht fallen und somit können sich bessere Lösungen wieder leichter durchsetzen. Als zweite bekannte Methode sei die Boltzmannselektion erwähnt (Lar06, vgl.S. 246). Sie funktioniert ähnlich wie jene von Forrester. Es wird dabei nämlich eine sogenannte Temperaturvariable eingeführt, welche es ebenfalls ermöglicht am Anfang weite Gebiete zu erforschen und gegen Ende mit sinkender „Temperatur“ den Druck auf schlechtere Funktionen zu erhöhen um schneller an ein globales Maximum zu kommen. Die dritte und letzte Methode, welche vorgestellt wird, wird Elitism genannt und wurde von De Jong (Jon75) entwickelt. Die Hauptidee, welche dabei zum Zug gekommen ist folgende: Sehr dominante Lösungen werden geschützt vor anderen weniger starken Teilnehmern. Somit sollen Veränderungen durch Mutationen und Kreuzungen mehr oder weniger ausgeschaltet werden. Es wird berichtet, dass diese Art des Algorithmus es schafft die Effizienz deutlich zu steigern.

Auch bei der Kreuzung von den einzelnen Elternpaaren haben sich unterschiedliche Implementierungen verbreitet (BL04). Besonders zu erwähnen ist dabei die Kreuzung, die mehrere ganze Abschnitte von den Eltern austauschen. Die Vorgehensweise ist dabei folgende. Es werden mehrere, frei gewählte Abschnitte zwischen den zwei Eltern gekreuzt und somit werden sogenannte Fehler aufgrund der Positionierung vermieden. Sollte aber die Positionierung eine gewisse Information widerspiegeln, ist dieser Fehler nicht bedeutend. Das Problem besteht aber darin, dass man solch ein Wissen oft nicht von vornherein zur Verfügung hat. Somit hat man mit dem mehrfachen abschnittsfachen Kreuzen eine gute Methode in der Hand um dieses Problem zu entschärfen. Die zweite Methode nennt sich „uniform crossover“ (ES03). Bei dieser Methode wird die Durchmischung beim Kreuzen des Elternpaares am stärksten betrieben. Die Elternknoten werden dabei bitweise durchlaufen und mit einer Wahrscheinlichkeit von fünfzig Prozent werden entweder

das Bit des einen Elternteils oder des anderen in die erste Kindlösung hineingegeben. Das zweite Kind erhält daher immer das Bit eines Elternteils, welches nicht zum Zug gekommen ist beim ersten Nachfahren. Die Vorteile dieser Methode liegen auf der Hand. Durch diese starke Durchmischung werden ganz neue Lösungen gefunden und somit ist das Gebiet, welches man von einer gegebenen Startgeneration erreichen kann um einiges grösser als bei den vorher angeführten Methoden. Es bleibt zu hinterfragen in wie weit dies nicht einfach durch eine Erhöhung der Mutationsrate ebenfalls bewerkstelligt werden könnte. Weiters wird bei diesem Ansatz mehr der Mutation zugetraut in einer fixen Region durch Abänderungen Optimallösungen zu finden. Literatur, welche diese Art der Implementierung näher erklärt und die Resultate bezüglich der Effizienz genauer begutachtet, war zu diesem Zeitpunkt nicht zu finden.

### 4.3 Trainieren von Neuronalen Netzen mit genetischen Algorithmen

Wie im Kapitel über neuronale Netze schon erwähnt, müssen die Netze trainiert werden. Dies geschieht über die Anpassung der Gewichte der einzelnen Kanten, welche die künstlichen Neuronen miteinander verbinden. Das Ziel des Trainings besteht darin, dass der Input der Trainingsdaten den gewünschten und vorgebenen Output liefern soll. Die Startgewichte werden meist als Zahlen und nicht in binärer Darstellung angegeben. Daher muss beim Einsatz von evolutionären/genetischen Algorithmen zuerst einmal alle Gewichte codiert werden. Diese werden dann mit Hilfe der Fitnessfunktion, welche den Output des Netzes mit dem des gewünschten vergleicht, evaluiert. Das Ziel ist es das Minimum zu finden in dem der Fehler zu den vorgebenen Output immer weiter verkleinert wird. Eine andere Verwendung von genetischen Algorithmen ist das Auffinden von einem günstigen Netzplan, welcher wieder den Wunsch nach guten Trainingsergebnissen erfüllen soll (BL04, S. 440). Genetische Algorithmen sind auch im Besonderen dafür geeignet die Nachteile, die durch Back-Propagationen Netzwerke entstehen könnten, zu verhindern (UMC03). Back-Propagation Netzwerke haben nämlich, anders als bei Genetischen Algorithmen das Problem, dass sie ihre Gewichte von einem Trainingszyklus zum nächsten nur sehr gering ändern (abhängig von dem Fehler zwischen Inputdaten und dem Ergebnis) daher kommt es sehr darauf an von Anfang an die Gewichte dementsprechend ordentlich auszuwählen. Dies verlangt einiges an Wissen und Erfahrung. Im Unterschied dazu spielen bei den genetischen Algorithmen die Startgewichte keine so große Rolle. Denn durch Mutation und Kreuzung können sehr wohl lokale Optima verlassen werden. Der Nachteil an der Nutzung von Genetischen Algorithmen ist aber, dass wesentlich mehr Rechenzeit von Nöten ist als bei einem standard Back-Propagation Netzwerk. Für bessere Resultate muss man auch längere Trainingszeiten in Kauf nehmen.

#### 4.4 Evolutionäre Algorithmen im Einsatz für Data Mining

Die Fallstudie (Bha00), welche in diesem Kapitel behandelt wird, beschäftigt sich mit dem Einsatz von genetischen Algorithmen im Bereich vom Marketing. Es kommt zum Einsatz ein genetischer Algorithmus, welcher Lösungen finden soll, die nicht nur auf ein Ziel maximiert sind, sondern auf eine Anzahl von Zielen, die sich mitunter auch entgegengesätzlich verhalten. Das Ziel ist paretoeffiziente Lösungen zu erhalten. Diese unterscheiden sich hauptsächlich durch die Höhe ihrer tradeoffs. Mehrdimensionale Optimierungen sind sehr häufig in der realen Welt gewünscht beziehungsweise erforderlich. Meist wird aber in der Praxis eine Aggregationsfunktion verwendet, die die Ziele nach bestimmten Kriterien fix gewichtet. Diese Entscheidungen sind bei komplexen Data Mining-Projekten nur schwer einsehbar und es können dabei wichtige Informationen verloren gehen.

Das Ergebnis der Studie zeigt das Elitism, welches dominante Lösungen schützt, tatsächlich gut geeignet ist um schnell gute Lösungen zu erhalten. Hier kommt es aber auch zu einer Wechselwirkung zwischen Anzahl der Mitglieder einer Generation und dem Einsatz von Elitism. Getestet wurde mit Populationsgrößen von 50,100 und 200. Es wurde nämlich gezeigt, dass je kleiner die Population ist, desto grösser ist der Effekt, den Elitism bewirkt im Vergleich zu einem Ergebnis ohne die Verwendung dieser Methode. Ein Problem ergibt sich aber aufgrund der Startwerte. Sollte nämlich in einer kleinen Population viele nicht-dominierte Lösungen auftauchen, so ist Elitism recht wirkungslos und es führt zu schlechteren Ergebnissen. Elitism ist aber trotzdem sehr wichtig für den Einsatz in der Praxis, denn grosse Populationen verbrauchen mehr Rechnerzeit und stellen eine unnötige Belastung von Ressourcen dar.

## 5 Zusammenfassung

In der Arbeit wurde am Anfang die Grundlagen von Data Mining erklärt. Es wird erläutert warum Data Mining immer mehr an Bedeutung gewonnen hat und wofür es eingesetzt wird. Danach wurde der grundsätzliche Ablauf eines Data-Mining Projektes, der Data-Mining-Process, behandelt und auf die einzelnen Phasen näher eingegangen. Es wurde dabei aufgezeigt, wie wichtig es ist, möglichst früh im Projekt das Problem genau zu beschreiben und eine formale Spezifikation zu kreieren. Weiters wurde beschrieben, dass die Aufbereitung der Daten nicht unterschätzt werden darf, denn diese ist meist vom Umfang deutlich höher anzusetzen als zum Beispiel das Erstellen eines Modells. Danach wurde auf die unterschiedlichen Möglichkeiten der Implementierung in Data Marts oder ähnlichen Alternativen eingegangen.

Zwei interessante Methoden wurden dann ausgewählt, welche im Data Mining eingesetzt werden können. Diese Methoden führen beide ein Schattendasein und werden in der Praxis weniger oft benutzt als andere Methoden. Trotzdem werden die Vorteile der Methoden gezeigt und vor allem beschrieben, wie der Prozess abläuft indem die Methoden aus einer Menge von Daten Modelle bilden können. Beide Methoden haben viele unterschiedliche Varianten anzubieten. Bei den neuronalen Netzen wurde auf diese zum Beispiel unterschiedliche Arten zu lernen hingewiesen. Es wurde auch auf das standard Backpropagation-Netzwerk kurz eingegangen und erklärt. Die neuronalen Netze können verwendet werden für die Ermittlung von zukünftigen Umsatzzahlen, Kundenbewertungen (siehe oben) oder für Marktsegmentierungen (Big96). Bei den evolutionären Algorithmen gibt es aber auch einige Varianten, welche auch dieser Arbeit präsentiert wurden. Die bekannteste ist Elitism, welche bereits scheinbar gute Lösungen versucht zu schützen. Diese wurde auch verglichen in einem Fallbeispiel mit Methoden ohne diesen Ansatz und es zeigt sich, dass dadurch eine deutliche Verbesserung möglich wird insbesondere, wenn die Grössen der Generationen klein sind (zwischen 50 und 100 Startvarianten in der ersten Generation). Es wird auch kurz angeschnitten wie diese beiden Methoden zusammen eingesetzt werden können. Weiters wurde auch am Anfang der jeweiligen Kapitel gezeigt wie diese Modelle aus der Natur abgeleitet wurden und mit einer kurzen Einführung wurde der biologische Hintergrund dargestellt. Auch dies fasziniert den Autor und trug zur Entscheidung der zu wählenden Methoden bei.

## Literatur

- [Ano] Anonymous. Axon. Website. <http://de.wikipedia.org/wiki/Axon>  
Accessed 8.7.2007.
- [Bha00] Siddhartha Bhattacharyya. *Evolutionary Algorithms in Data Mining*. KDD, 2000.
- [Big96] Joseph P. Bigus. *Data Mining with Neural Networks*. McGraw-Hill, 1996.
- [BL04] Micheal J.A. Berry and Gordon S. Linoff. *Data Mining Techniques*. Wiley, 2004.
- [BS96] Niels Birbaumer and Robert Schmidt. *Biologische Psychologie*. Springer Verlag, 1996.
- [DG89] Kalyanmoy Deb and David Goldberg. *An Investigation of niche and species formation in genetic function optimization*. Morgan Kaufmann, 1989.
- [DH01] Rhonda Delmater and Monte Hancock. *Data Mining Explained*. Digital Press, 2001.
- [ES03] A. Eiben and Jim Smith. *Introduction to Evolutionary Computing*. Springer Verlag, 2003.
- [For85] Stephanie Forrest. *Scaling of fitnesses in the genetic algorithm*. Morgan Kaufmann, 1985.
- [GS06] Rayid Ghani and Carlos Soares. *Data mining for business applications: KDD-2006 workshop*, volume 8. ACM Press, 2006.
- [Heb49] Donald O. Hebb. *The Organization of Behavior*. Wiley, 1949.
- [HK00] Jiawei Han and Micheline Kamber. *Data Mining :Concepts and Techniques*. Morgan Kaufmann Publishers, 2000.
- [HMS01] David Hand, Heikki Mannila, and Padhraic Smyth. *Principles of Data Mining*. Mit Press, 2001.
- [HT85] J.J. Hopfield and D.W. Tank. *Neural computation of decisions in optimization problems*. Biological Cybernetics, 1985.
- [Jon75] Kenneth De Jong. *An Analysis of the behavior of a class of genetic adaptive systems*. University of Michigan, 1975.
- [Lar06] Daniel T. Larose. *Data Mining: Methods and Models*. Wiley, 2006.

- [Lip] Wolfram-M. Lippe. Geschichte neuronaler netze. <http://cs.uni-muenster.de/Professoren/Lippe/lehre/skripte/wwwnscript/geschichte.html> Accessed 9.7.2007.
- [MCM83] R. S. Michalski, J. G. Carbonell, and T. M. Mitchell. *Machine Learning*. Morgan Kaufmann, 1983.
- [MP69] M. Minsky and S. Papert. *Perceptrons*. MIT Press, 1969.
- [MP88] Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. pages 15–27, 1988.
- [RHW86] D.E. Rumelhart, G.E. Hinton, and R.J. Williams. Learning representations by back-propagating errors. 323:533–536, 1986.
- [Rud01] Olivia Parr Rud. *Data Mining Cookbook*. Wiley, 2001.
- [SNG90] Eberhard Schöneburg, Hasen Nikolaus, and Andreas Gawelczyk. *Neuronale Netze*. Markt and Technik Verlag, 1990.
- [UMC03] Nishant Unnikrishnan, Ajay Mahajan, and Tsuchin Chu. *Intelligent system modeling of a three-dimensional ultrasonic positioning system using neural networks and genetic algorithms*. Journal of Systems and Control Engineering, 2003.
- [Zel00] Andreas Zell. *Simulation Neuronale Netze*. Addison-Wesley, 2000.
- [ZK03] Arlene Zaima and James Kashner. *A Data Mining Primer for the .. Business Intelligence Journal*, 2003.