Carl von Ossietzky
Universität Oldenburg
Fachbereich Informatik

Diplomarbeit

Unscharfe Methoden zur Analyse individuenorientierter Simulationsmodelle in der Epidemiologie

Jörg Cassens

Oldenburg, den 6. Oktober 1998

Meinen Eltern

*Bei der wissensbasierten Entscheidungsunterstützung in der Medizin geht es nicht primär um technische oder informatische Ziele. Es geht in erster Linie um den Patienten und um ein besseres Verständnis von Erkrankungen. Forschungsarbeiten auf dem Gebiet der wissensbasierten Entscheidungsunterstützung in der Medizin müssen sich daran messen lassen, ob sie einen nachweisbaren Nutzen für die Patientenversorgung oder für die Erkenntnisgewinnung in der Medizin erzielen konnten. Wer diesen Nutzen nicht aufzeigen kann, muß sich die Frage stellen, ob die Forschungsarbeiten nicht doch eher informatische Spielereien waren (*Haux *et al., 1997, S. 26).*

Inhalt

Kapitel 1

Einleitung und Motivation

*Wissenschaft ist Luxus. Aus diesem Grund kann die Rechtfertigung der wissenschaftlichen Bearbeitung eines Gegenstandes nicht mit Notwendigkeit und Allgemeinheit gelingen (*Bensch*, 1994, S. 3).*

1.1 Einleitung

Diese Diplomarbeit soll einen Beitrag zu Untersuchungen zum Einsatz individuenorientierter Simulationsmodelle und deren Analyse zur Unterstützung der umweltepidemiologischen Forschung liefern. Ziel ist es, das Design empirischer Studien zu optimieren, die Identifizierung von Ursache-Wirkungs-Zusammenhängen zu erleichtern sowie die Entwicklung neuer und den Test bekannter Hypothesen zu unterstützen (vergleiche Köster und Sonnenschein (1998)). In der vorliegenden Arbeit wird die Analyse individuenorientierter Simulationsmodelle mittels unscharfer und KI-basierter Methoden des Maschinellen Lernens untersucht.

1.2 Problemstellung

Im Rahmen einer epidemiologischen Untersuchung wird ausgehend von der Beobachtung einer extremen gesundheitlichen Situation innerhalb der Bevölkerung eine empirische Studie durchgeführt, durch welche Hypothesen zur Erklärung der Gesundheitssituation überprüft werden sollen. Dazu werden die in der Studie erhobenen Daten mittels statistischer Methoden ausgewertet. Diese Auswertung kann die angenommenen Hypothesen erhärten oder dazu führen, daß die ursprüngliche Annahme verworfen oder zumindest modifiziert wird, und daß eventuell weitere Untersuchungen notwendig werden (siehe Abbildung 1.1).

1.2.1 Identifikation von Problembereichen

Ein solches Vorgehen kann aus verschiedenen Gründen zu unbefriedigenden Resultaten führen. So kann eine zu geringe Größe der Studienpopulation bzw. vom Umfange her nicht ausreichen-

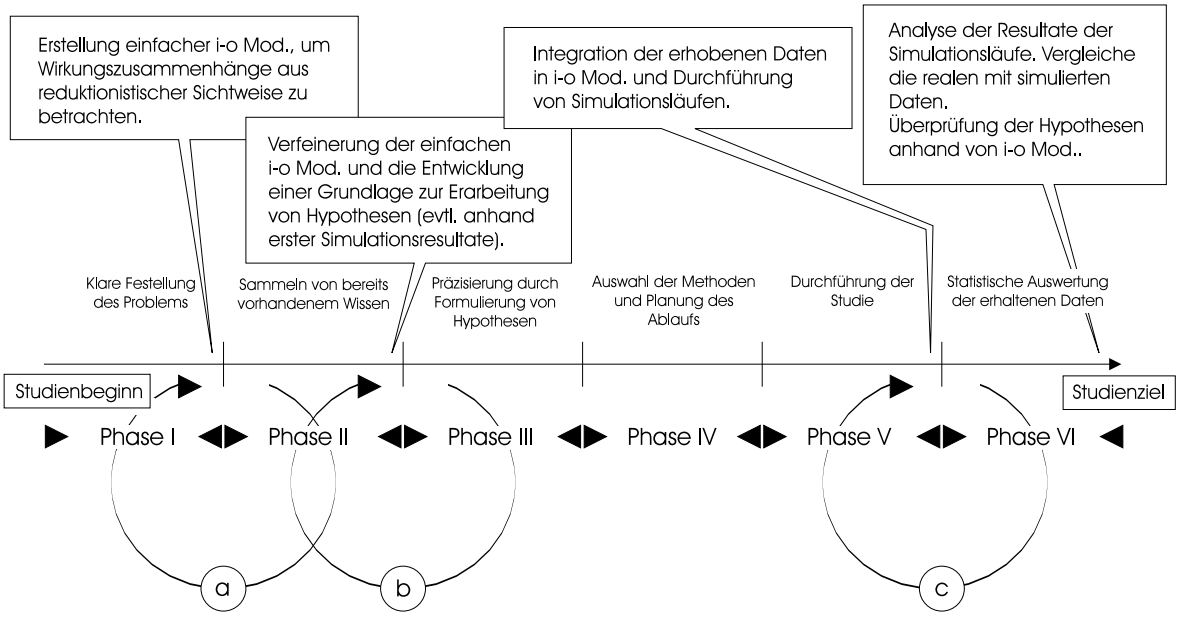

Abbildung 1.1: *Diese Graphik zeigt schematisiert die Integration individuenorientierter Modellbildung und Simulation in epidemiologische Studien.*

de Datenerhebung die Auswertung erschweren oder unmöglich machen. Durch individuelle Verhaltensmuster beeinflußbare oder zeitlichen sowie räumlichen Schwankungen unterliegende Aspekte, wie beispielsweise die Exposition bzgl. bestimmter Giftstoffe, sind einer statistischen Analyse insbesondere dann schwer zugänglich, wenn es keine präzisen Hypothesen über die zu untersuchenden Ursache-Wirkungs-Zusammenhänge gibt.

Eine im naturwissenschaftlichen Experiment übliche Variation von Rahmenbedingungen, z.B. der Menge aufgenommener Giftstoffe, verbietet sich aus ethischen Gründen.

Ein simulationsbasiertes Werkzeug zur Unterstützung der epidemiologischen Forschung könnte die Identifikation und nähere Untersuchung von Kausalzusammenhängen vereinfachen und beschleunigen. Die rechte Seite der Abbildung 1.2 soll das illustrieren. Ausgehend von der zu überprüfenden Hypothese wird ein Simulationsmodell erstellt. Bereits vorhandene Modelle und Daten aus inhaltlich verwandten Erhebungen sind hierbei zu integrieren. Die in der Simulation gewonnen Daten werden analysiert und dienen zur Modifikation der Hypothese. Ausgehend von derart erhärteten Annahmen kann dann eine empirische Studie erfolgen, welche die Kausalzusammenhänge im Idealfall deutlich besser erfassen kann.

1.2.2 Individuenorientierte Modelle

Im Zusammenhang mit dieser Arbeit werde ich mich mit individuenorientierten Simulationsmodellen beschäftigen (vgl. SONNENSCHEIN UND VOGEL (1996)). Während klassische Modelle den zu simulierenden Ausschnitt der realen Welt oft undifferenziert als Gesamtheit betrachten, erfolgt bei der individuenorientierten Methode eine Spezifikation des Verhaltens einzelner Individuen und Umweltkomponenten, aus denen sich wiederum das ganze modellierte System

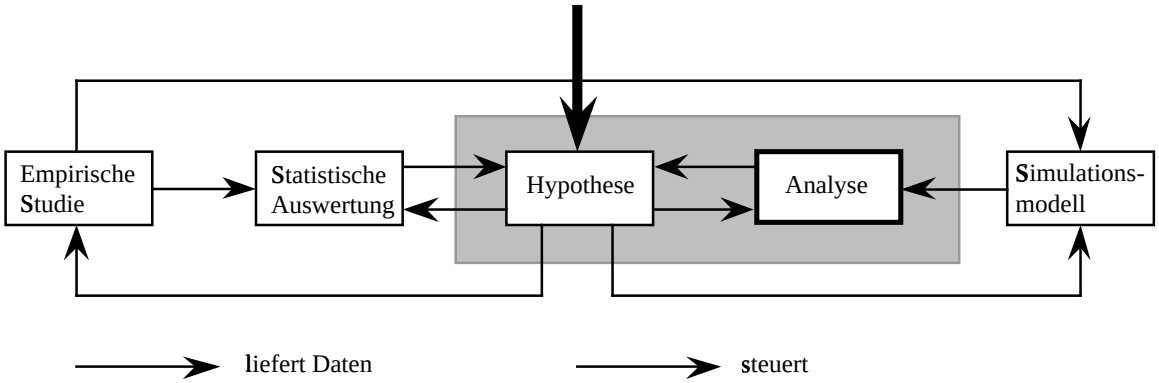

Abbildung 1.2: *Ausgangs- und Endpunkt jeder epidemiologischen Untersuchung ist eine Hypothese. Die Graphik zeigt auf der linken Seite den Weg über empirische Studien und statistische Auswertungen zurück zur evtl. zu modifizierenden Hypothese. Auf der rechten Seite ordnen sich Simulationsmodelle und deren Analyse ein. Grau hinterlegt ist der für meine Arbeit zentrale Bereich im Zusammenspiel von Analyse und Hypothese.*

zusammensetzt. Attribute einzelner Individuen können modifiziert werden, womit zum Beispiel individuelle Unterschiede sowie die Interaktion verschiedener Individuen modelliert und in Experimenten variiert werden können.

Der Einsatz individuenorientierter Simulationsmodelle eröffnet in dem hier betrachteten Anwendungsgebiet eine Reihe von Möglichkeiten.

So können beispielsweise statische Beschreibungen, die u.a. aus Umfragen gewonnen werden, mit dynamischen Faktoren, wie z.B. dem bei einer Langzeitmessung ermittelten zeitlichen Verlauf von Schadstoffkonzentrationen, in das dynamische System eines Simulationsmodells integriert werden. Hieraus ergibt sich eine detailliertere Sicht auf das zu untersuchende System. Faktoren, die in der realen Welt nicht veränderbar sind, können zur Überprüfung von Hypothesen variiert werden.

Daten aus unterschiedlichen Erhebungen können kombiniert, bereits anderweitig gesicherte Erkenntnisse integriert werden. Auch einer empirischen Studie nicht oder nur sehr schwer zugängliche Daten können aufgenommen werden, so z.B. über einen langen Zeitraum zu messende oder nur subjektiv zugängliche Aspekte wie die tatsächliche Exposition eines Individuums bzgl. eines bestimmten Giftstoffes.

Der Ansatz eines simulationsbasierten Gesamtsystem und die sich daraus ergebenden Möglichkeiten sind in KÖSTER UND SONNENSCHEIN (1998) dargestellt.

Ein grundlegendes Konzept eines solchen Simulationssystems für die umweltepidemiologischen Forschung und seine prototypische Implementierung wird in BARTELS (1998) beschrieben. Eine Beschreibung des Einsatzes von Techniken des Data Mining zur Analyse der so gewonnenen Daten findet sich in KÖSTER UND SONNENSCHEIN (1999).

1.3 Ziel der Arbeit

Im Rahmen meiner Arbeit werden die Spuren, d.i. eine protokollierte Historie, der Individuen zur Klassifizierung von Individuengruppen genutzt. Mit Hilfe unscharfer und KI-basierter Methoden sollen Cluster in diesen Daten ermittelt werden, die über gemeinsame Merkmale verfügen. Entspricht einem derart ermittelten Cluster eine Gruppe von Menschen im modellierten Weltausschnitt, bei denen z.B. eine zu untersuchende Krankheit auftrat, könnten die gemeinsamen Merkmale des Clusters Hinweise auf Ursachen für die Krankheit geben, falls es gelingt, die gefundenen Merkmale explizit im Datenmaterial zu lokalisieren und auf ihre medizinische Plausibilität zu untersuchen.

Dadurch kann zum einen die Überprüfung bekannter Hypothesen unterstützt werden. Zum anderen können aber auch Muster deutlich werden, die vorher nicht sichtbar waren. Somit kann die Bildung neuer Hypothesen durch die anwendenden Forscher stimuliert werden. Der grau unterlegte Bereich in Abbildung 1.2 soll den engen Zusammenhang zwischen Hypothesenbildung und Analyse der Simulationsdaten unterstreichen.

1.3.1 Analyse von Simulationsmodellen

Ein wesentlicher Teil der Arbeit wird sich mit der Möglichkeit befassen, bei der Auswertung von Spuren einzelner Individuen KI-basierte und unscharfe Methoden zu verwenden. Ziel einer solchen Analyse soll sein, die Erarbeitung und den Test epidemiologischer Hypothesen zu unterstützen.

Bei den Spuren handelt es sich in erster Linie um über die Zeit aufgenommene Größen, sogenannte *Zeitreihen*. Diese liegen in diskreter Form vor, d.h. daß ihr Wert für unterschiedliche, diskrete Zeitpunkte angegeben ist.

Jedes der Individuen hat solche Zeitreihen als Werteausprägungen verschiedener Attribute. Dabei kann es sich z.B. um die im Verlauf der Simulation mehrfach errechnete Exposition gegenüber verschiedenen Giftstoffen handeln, oder um die Anzahl der pro Tag gerauchten Zigaretten, oder andere individuell unterschiedliche und über die Zeit variierende Merkmalsausprägungen.

In dieser Arbeit werden drei Arten von Zeitreihen unterschieden:

1. **Linear:** Liegt der Wert einer solchen Zeitreihe für einen bestimmten Zeitpunkt nicht vor, so läßt er sich aus den unmittelbar benachbarten, diskreten Zeitpunkten linear interpolieren.

2. **Diskret:** Eine solche Interpolation ist bei diesem Typus nicht möglich. Als Wert zu einem bestimmten Zeitpunkt gilt der Wert zum unmittelbar vorhergehenden diskreten Zeitpunkt.

3. **Statisch:** Ein Attribut dieses Wertebereichs bleibt über die gesamte Zeitdauer konstant.

Ohne Beschränkung der Allgemeinheit wird angenommen, daß die vorliegenden Zeitreihen auf das Einheitsintervall $[0 \ldots 1]$ normalisiert sind. Dies gilt sowohl für die t-Achse, also die

diskreten Zeitpunkte, als auch für die y-Achse, also die Werte. Eine solche Normalisierung ist zum einen möglich, weil die Analyse nicht auf quantitative Aspekte vergleichbarer Merkmale abhebt, sondern qualitative Aspekte zwischen heterogenen Merkmalen zum Ziel hat. Zum anderen kann durch die Anbindung an individuenorientierte Smulationsmodelle die Normierung der Erhebungsfunktionen sichergestellt werden.

Für eine reale Anwendung muß sicherlich genauer untersucht werden, ob die dadurch gewonnene Vereinfachung im Analyseprozeß durch einen unverhältnismäßig großen Aufwand in der Vorverarbeitung erkauft wird, oder ob die Ausgabe des Simulators eine solche Normalisierung ohne Zusatzaufwand zur Verfügung stellen kann.

1.3.2 Hypothesenbildung

Die einzelnen Indiduen sind als *krank* oder *gesund* gekennzeichnet. Ziel ist es, mit Hilfe von Techniken des Maschinellen Lernens einen expliziten Klassifikator für diese beiden Zielkonzepte zu erstellen. Dieser Klassifikator repräsentiert eine mögliche Hypothese. Sie soll Aufschluß über mögliche Kausalzusammenhänge geben, die zwischen der bestimmten Ausprägung eines Attributes und der Klassifikation bestehen.

Die so generierte Hypothese soll durch eine Expertin des Anwendungsgebietes auf epidemiologische Plausibilität überprüft werden. Dadurch ist es prinzipiell möglich, bisher ungekannte Einflußfaktoren zu identifizieren. Einflußfaktoren in diesem Sinne sind sowohl bestimmte Ausprägungen einer Zeitreihe, die den Ausbruch der Krankheit begünstigen, aber auch solche, die ihn hemmen.

1.4 Aufbau der Arbeit

1.4.1 Theorie

Zunächst werden einige wichtige Grundlagen aus dem Bereich des Maschinellen Lernens erläutert.

Es wird weiterhin ein Überblick der verschiedenen Ansätze aus klassischer, regelbasierter KI, neuronaler Netze und Fuzzy-Systemen gegeben, wie sie beispielsweise im Bereich des Data Mining eingesetzt werden. Das schließt eine Diskussion über die Anwendbarkeit auf den hier behandelten Themenbereich ein.

In jeweils gesonderten Kapiteln werden symbolische Verfahren des Maschinellen Lernens und (subsymbolische) Clustering-Verfahren vorgestellt.

1.4.2 Praxis

Aufbauend auf dieser Übersicht und basierend auf den vorgestellten Techniken wird ein Konzept zur Hypothesengenerierung entwickelt und dessen Anwendbarkeit in der epidemiologischen Forschung diskutiert.

Implementierung

Das vorgeschlagene Konzept ist prototypisch in C++ implementiert. Dabei wird auf der $\mathcal{MLC}$++-Bibliothek (Machine Learning Library in C++) aufgesetzt, die z.B. in KOHAVI *et al.* (1996) beschrieben wird. Die Anwendbarkeit wird anhand von ausgewählten Datensätzen exemplarisch überprüft.

Diskussion und Ausblick

Den Schlußteil der Arbeit bildet eine Diskussion der erzielten Resultate mit dem Schwerpunkt auf zwei Fragestellungen: Bilden individuenorientierte Simulationsmodelle aus erkenntnistheoretischer Sicht einen sinnvollen Ansatz in der (epidemiologischen) Forschung, und, wenn ja, unter welchen Voraussetzungen? Können KI-basierte und unscharfe Analysemethoden individuenorientierter Simulationsmodelle eine Hilfestellung bei der Erarbeitung und dem Test epidemiologischer Hypothesen geben?

Teil I

Theorie

Kapitel 2

Grundlagen: Maschinelles Lernen

*Unfortunately, it is not always easy to separate fact from media hype (*Fayyad *et al., 1996a, S. 38).*

2.1 Einleitung

Auf die Frage „Was ist Maschinelles Lernen?“ versucht Simon (1983) die folgende Antwort zu geben: „Learning denotes changes in the system that are adaptive in the sense that they enable the system to do the same task or tasks drawn from the same population more efficiently and more effectively the next time.“

Morik (1995) weist zurecht darauf hin, daß damit auch Phänomene abgedeckt seien, die üblicherweise nicht zum Lernen gezählt werden. Sie pointiert dies mit dem Verweis, daß man einen solchen Effekt erzielen könne, wenn man das gleiche Programm auf einem schnelleren Rechner ablaufen ließe.

Sieht man einmal von der Robotersteuerung ab, so beschäftigt sich die Künstliche Intelligenz zumeist mit nicht-motorischen Phänomenen. Wenn man sich dazu die Unterscheidung der KI von Wissen und Daten in Erinnerung ruft und mit Jantke (1995) Wissen als Daten plus Interpretation bezeichnet, so nähert man sich einer anderen Definition an, die sich folgendermaßen charakterisieren läßt: Lernen ist die Veränderung von Wissen aus Wissen und Erfahrung.

Erfahrung in diesem Sinne bedeutet, daß dem System Daten „gezeigt“ werden, die es entsprechend seiner Interpretationsmöglichkeiten „untersucht“. So können beispielsweise Cluster in der Datenmenge gefunden werden, die evtl. in einem weiteren Schritt zueinander in Relation gesetzt werden können. In dieser Form kann das System somit Wissen über die Daten gewinnen. Diese Erfahrung dient dann weiterhin dazu, das Wissen des Systems gezielt zu modifizieren, also Daten oder Interpretation oder beides.

Bei Langley und Simon (1995) findet sich folgende Definition: „Machine learning is the study of computational methods for improving performance by mechanizing the acquisition

of knowledge.“ Hier wird der Aspekt der Wissensaneignung stärker betont als in Simons früherer Aussage.

Für den vorliegenden Gegenstandsbereich ist von entscheidender Bedeutung, daß das angeeignete Wissen explizierbar ist. Diese Explikation ist die gesuchte Hypothese über kausale Zusammenhänge.

Damit deutet sich allerdings bereits an, daß der Fokus des Maschinellen Lernens und der vorliegenden Arbeit auf unterschiedlichem Gebiet liegen. Das von einem System des Maschinellen Lernens angeeignete Wissen soll dazu dienen, eine bestimmte, andere Leistung zu erbringen, z.B. eine Klassifikation zu ermöglichen. Mein Hauptaugenmerk liegt hingegen auf dem angeeigneten Wissen selber.

Trotzdem halte ich Ergebnisse aus dem Fachgebiet des Maschinellen Lernens für wertvoll in Hinblick auf das vorliegende Problem, da mit den dort entwickelten Systemen prinzipiell die Generierung expliziter Hypothesen möglich ist.

Es sei noch angemerkt, daß es auch Untersuchungen zu Systemen gibt, die aktiv in ihre Umwelt eingreifen, und exploratives Lernen betreiben. Solche Verfahren werden z.B. zur (retrospektiven) Computermodellierung wissenschaftlicher Entdeckungsprozesse benutzt, wie unter anderem von LANGLEY *et al.* (1983) und GRASSHOFF (1995) beschrieben. Hierbei generiert das lernende System Versuchsaufbauten, mit denen es sein Weltwissen vervollkommnen möchte.

Die vorliegende Arbeit bettet sich in ein Framework ein, in dem letztlich ebenfalls eine Wechselwirkung zwischen einem hypothesengenerierenden System und seiner Umwelt stattfindet, letztere repräsentiert durch detaillierte Simulationsmodelle epidemiologischer Forschung. Dort wäre auch die Entwicklung explorativ arbeitender Systeme möglich, indem das Analysewerkzeug ausgehend von gefaßten Hypothesen das der Simulation zugrunde liegende Modell selbständig modifiziert.

2.2 Abgrenzung zu anderen Bereichen

Eine Data Mining-Aufgabe besteht darin, in einer großen Datenmenge Regularitäten zu entdecken: die vorliegenden Daten sollen geclustert, funktionale Beziehungen sollen entdeckt und Ähnlichkeiten offengelegt werden. Eine Interpretation dieser Aufgabe aus Sicht des Maschinellen Lernens ist es, diese Regularitäten als Interpretationen auf diesen Daten zu begreifen.

Trotz der Tatsache, daß Maschinelle Lernverfahren einen statistischen Kern enthalten, sieht MORIK (1995) Unterschiede zu den klassischen statistischen Verfahren. Das Maschinelle Lernen geht in der Aufbereitung der Ergebnisse über die Statistik hinaus, indem direkt Hypothesen generiert werden. Die klassische Statistik ist dahingegen vor allem dazu geeignet, bereits vorhandene Hypothesen am Datenmaterial zu überprüfen bzw. zu quantifizieren.

Auch von der z.B. bei BIEHLER (1982) beschriebenen explorativen Statistik, die ebenfalls mit dem Anspruch auftritt, neue Phänomene in Daten entdecken zu können, unterscheidet sich das Maschinelle Lernen. Während bei der explorativen Statistik die Interaktion mit dem Menschen eine wichtige Rolle spielt, liegt der Fokus bei Systemen des Maschinellen Lernens wie auch im „klassischen“ Data Mining auf automatischen Verfahren, bei denen der Mensch nicht eingreift.

Auch wenn ich im folgenden mehrfach auf Konnektionistische Systeme eingehe, so zählt man nur symbolische Verfahren (vgl. Unterkapitel 2.4) zum Maschinellen Lernen im engeren Sinne. Genauso grenzt sich das Maschinelle Lernen von den Genetischen Algorithmen ab, die von Radtke (1998) bereits auf ihre Anwendbarkeit im Kontext der Analyse von Simulationen innerhalb der Umweltepidemiologie untersucht wurden.

2.3 Überwachtes vs. nicht-überwachtes Lernen

Eine Möglichkeit, Systeme zum Maschinellen Lernen zu unterscheiden, ist ihre Aufteilung in überwachte und nicht-überwachte Lernverfahren. Während bei ersteren eine Zielfunktion vorgegeben wird und das System dessen Realisierung zu lernen hat, ist bei letzteren keine solche Zielfunktion vorgegeben, sondern diese muß vom System erst antizipiert werden.

2.3.1 Überwacht

Bei einer überwachten Lernaufgabe wird dem System eine Menge von Beispielen aus einer Eingabemenge $\mathcal{E}$ und eine Klassifikation gegeben. Die Aufgabe besteht dann darin, eine geeignete Abbildung (auch Beschreibung genannt) zu finden, welche die Elemente aus $\mathcal{E}$ auf die entsprechenden Klassen abbildet. Eine Instanz, welche dem System hierzu die korrekte Klassifizierung mitteilt, wird teilweise Lehrer oder Orakel genannt.

Eine Klasse

Es wird eine Klasse $\mathcal{K}$ definiert, für die das System eine sogenannte *charakteristische Beschreibung* B finden soll. Dies ist eine Beschreibung, welche die Beispiele aus der Klasse $\mathcal{K}$ von den Beispielen unterscheiden soll, die nicht aus $\mathcal{K}$ sind.

Es sind zwei weitere Unterformen denkbar. Zum einen können dem System sowohl positive Beispiele, d.h. Beispiele aus $\mathcal{K}$, als auch negative Beispiele, also nicht aus $\mathcal{K}$, präsentiert werden. Die negativen Beispiele stehen dann für eine (möglicherweise unendliche) Zahl von Klassen, gegen die $\mathcal{K}$ abgegrenzt werden muß.

Zum anderen können nur positive Beispiele präsentiert werden. Im diesem Fall geht man von der *closed world assumption* aus, d.h. daß angenommen wird, daß die präsentierten Beispiele $\mathcal{K}$ gegen alle anderen möglichen Beispiele hinreichend genau abgrenzen. Unter dieser Annahme wird für alle Beispiele b, für die unter B nicht ausgesagt werden kann, daß sie aus $\mathcal{K}$ sind, ausgesagt, daß sie nicht aus $\mathcal{K}$ sind: $\forall b \in \mathcal{E} : \neg(b \in_B \mathcal{K}) \Rightarrow b \notin \mathcal{K}$.

Mehrere Klassen

Hierbei gibt der Lehrer eine ganze Reihe von Klassen $\mathcal{K}_1, \mathcal{K}_2, \ldots, \mathcal{K}_n$ vor. Wieder gibt es zwei Möglichkeiten: Das System kann für jede Klasse $\mathcal{K}_i$ eine *charakteristische Beschreibung* finden, die ein Beispiel aus $\mathcal{K}_i$ von jedem beliebigen anderen Beispiel unterscheidet. Bezogen auf den Fall mit nur einer zu lernenden Klasse hieße das, eine Beschreibung für positive Beispiele aus $\mathcal{K}_i$ zu lernen, wobei alle Beispiele aus anderen Klassen als negative Beispiele gelten.

Das System kann aber auch eine *diskriminierende Beschreibung* liefern, also ein Konzept entwickeln, welches Beispiele aus $\mathcal{K}_i$ von Beispielen aus $\mathcal{K}_j$ mit $j \neq i$ unterscheidet.

2.3.2 Nicht-überwacht

Bei einem nicht-überwachten Lernverfahren wird dem System kein Wissen über Klassenzugehörigkeiten mitgeteilt. Solche Ansätze werden gut durch die Begriffe Lernen aus Beobachtung und Konzeptuelles Clustering charakterisiert. Der letztere Begriff betont, daß es um einen Konzepterwerb geht, also darum, aus der Empirie eine Kategorisierung vorzunehmen. Dabei ist das Konzept der Kategorie vorab vorhanden, und es wird eine Belegung für dieses Konzept gesucht.

Letztlich geht es also darum, daß das System eine Eingabemenge $\mathcal{E}$ auf eine geeignete Art und Weise partitioniert, und gleichzeitig eine *charakteristische* oder *diskriminierende Beschreibung* für die gefundenen Partitionen findet. Das Ziel ist das Auffinden einer abstrakten Beschreibung der Eingabemenge.

2.4 Symbolische vs. Subsymbolische Systeme

Die KI insgesamt und nicht nur das Maschinelle Lernen zerfällt in zwei große Teile: die symbolische KI, die intelligentes Verhalten auf die Manipulation von Symbolen zurückführt, und die subsymbolische KI, die Fähigkeiten zu symbolischer Manipulation als Emergenzphänomen begreift. Das Maschinelle Lernen im engeren Sinne bezeichnet zumeist nur den symbolischen Zweig.

2.4.1 Symbolisch

Das Wort Symbol, wie es in der KI zumeist verwendet wird, bezeichnet nach DORFFNER (1991) eine *eindeutig identifizierbare und lokalisierbare Einheit*, die für „Etwas“ steht bzw. dieses „Etwas“ repräsentiert. Das Aussehen, also die Repräsentation ist dabei beliebig. In symbolverarbeitenden Systemen sind Symbole die kleinsten bedeutungstragenden Einheiten. Diese Symbole können allerdings in kleinere syntaktische Einheiten zerfallen.

Da zu jedem Symbol eine Interpretation gehört, hat auch jede Handlung auf den Symbolen eine Interpretation. Modifikationen auf den Symbolen sind eindeutig Modifikationen auf den Interpretationen zuzuordnen.

Die symbolische KI geht von der Hypothese aus, daß ein genügend leistungsfähiges symbolverarbeitendes System grundsätzlich eine hinreichende Voraussetzung für ein intelligentes System ist.[1] (Künstliche) Intelligenz wird im Forschungsgebiet der symbolischen KI letztlich auf die Manipulation von Symbolen zurückgeführt.

Auch das (klassische) Maschinelle Lernen ist in diesem Sinne stets Symbolmanipulation. Es werden Entscheidungsbäume mit expliziten logischen Tests aufgebaut oder Schlußregeln generiert, es findet eine symbolische Abstraktion statt.

[1] Für die starke symbolische KI ist dies eine hinreichende und notwendige Bedingung.

Symbolische KI-Systeme haben den Vorteil, daß ihr Verhalten durch die Notwendigkeit der Explizierbarkeit immer interpretierbar bleibt. Allerdings wird damit vorausgesetzt, daß alle kognitiven Vorgänge auf einer Grundlage formalisierbar sind. Wo eine solche Formalisierung nicht gelingt oder einen hohen Aufwand erfordert, findet die Interpretierbarkeit und damit die klassische KI selbst ihre Grenzen.

2.4.2 Subsymbolisch

Die subsymbolische KI gibt die Verankerung ihrer Repräsentation in einem eindeutig interpretierbaren Kontext auf. Sie versteht sich als eine Form der Modellierung assoziativer und intuitiver Prozesse, die mit der klassischen Symbolverarbeitung schlecht oder gar nicht handhabbar sind. Damit ist das Einzelphänomen nicht mehr an ein erkennbares *Etwas* gekoppelt. Aus diesem Grund spricht man häufig von einem Black Box-Charakter der subsymbolischen KI: das Gesamtsystem erfüllt eine bestimmte Funktion, das *Was* ist bestimmbar, aber auf welcher Grundlage das Verhalten des Systems entsteht bleibt unklar, das *Wie* ist höchstens im Nachhinein analysierbar.

Damit muß auch der Wissensbegriff der klassischen KI revidiert werden. Anstatt in symbolischen Daten, Schlußregeln und Interpretation liegt das Wissen in nicht symbolischer Form vor. Damit ist das Maschinelle Lernen für die subsymbolische KI anders zu definieren.

CARBONELL *et al.* (1983) formulieren es so, daß es um die Anpassung numerischer Parameter und Koeffizienten in bestimmten algebraischen Gleichungen geht, so daß das lernende System zum Schluß eine bestimmte Leistung zeigt. Diese Auffassung korrespondiert gut sowohl mit den bekannten künstlichen Neuronalen Netzen, bei denen es um die Anpassung von Verbindungsgewichten geht, als auch mit numerischen Optimierungsverfahren, wie z.B. der Learning Vector Quantisation von Kohonen (siehe Unterkapitel 4.4).

Die subsymbolische KI hat da ihre Stärken, wo es darum geht, Zielfunktionen zu approximieren, für die eine mathematisch exakte Formulierung nicht möglich oder zu aufwendig wäre. Eingeschränkt ist ihre Anwendbarkeit in Bereichen, in denen gerade keine Black Box gefragt ist, sondern eine explizite Formulierung der realisierten Funktion.

2.5 Hypothesensprache

Eine weitere Achse, entlang derer eine Charakterisierung von Verfahren des Maschinellen Lernens möglich ist, ist die Sprache, in der die generierten Konzepte ausgedrückt werden.

2.5.1 Attributierte Logiken

Die klassischen Lernverfahren arbeiten alle mit dieser Klasse von Repräsentationen, die aus Attribut-Wert-Bedingungen besteht. Sie baut auf der Propositionalen Logik auf. Die Aussage „(Farbe = Grün $\vee$ Farbe = Rot) $\wedge$ Form = Kreis“ ist eine (allerdings nicht allgemeingültige) Aussage in Konjunktiver Normalform, einem propositionalen Kalkül.

Im Unterschied zur Propositionalen Logik benötigen die Repräsentationssprachen von Maschinellen Lernsystemen Logiken mit Variablen. Die Grundidee ist dabei, Beispiele und Konzepte mit Wertebelegungen einer Menge von *Attributen* zu charakterisieren. Die Aussagen beziehen sich immer auf ein Objekt. Insofern würde das obige Beispiel in Attributierter Logik für ein beliebiges Objekt X richtig lauten: „$(X.\text{Farbe} = \text{Grün} \vee X.\text{Farbe} = \text{Rot}) \wedge X.\text{Form} = \text{Kreis}$."[2]

Attribute können grundsätzlich Boole'sche, numerische oder symbolische Werte annehmen. Teilweise kann es eine Ordnung auf den Wertebereichen geben. Von Algorithmus zu Algorithmus gibt es allerdings unterschiedliche Einschränkungen für mögliche Wertebelegungen von Attributen.

Es gibt, über das einfache vorgestellte Beispiel hinaus, verschiedene Ausdrucksformen für diese Logiken:

Entscheidungsbäume

Hierbei handelt es sich um eine weit verbreitete und mit Erfolg eingesetzte Repräsentation. Ein Entscheidungsbaum eignet sich zur Klassifikation von Beispielen, die einer endlichen Menge von Klassen angehören können. Diese Repräsentation benutzen z.B. CART, ID3, C4.5 (vgl. Unterkapitel 3.2).

Knoten sind mit Attributnamen beschriftet, und die ausgehenden Kanten mit möglichen Ausprägungen dieser Attribute. Die Klassifikation eines Objektes X erfolgt derart, daß von der Wurzel ausgehend in jedem Knoten derjenigen ausgehenden Kante gefolgt wird, deren Beschriftung der aktuellen Wertebelegung des entsprechenden Attributes von X entspricht.

Die Blätter sind zumeist mit Klassenbezeichnern versehen. Das zu klassifizierende Objekt gehört zu der Klasse des Blattes, zu dem der oben skizzierte Pfad führt. Es können aber auch Wahrscheinlichkeiten für mehrere Klassen angegeben sein (probabilistische Bäume). Bei Bäumen zur Regressionsanalyse stehen in den Blättern Werte der gewünschten Zielfunktion.

Ein Nachteil von Entscheidungsbäumen ist, daß sie sehr unübersichtlich werden können. Dies ist bei realen Data Mining Anwendungen auf großen Datenmengen häufig zu beobachten. Es gibt daher Algorithmen, zuerst einen Entscheidungsbaum aufbauen, um diesen dann in ein Regelsystem zu übertragen, vgl. beispielsweise QUINLAN (1993).

Entscheidungsregeln

Entscheidungsregeln sind Wenn-Dann-Regeln, in deren Bedingungsteil mögliche Wertebelegungen von Attributen in Form propositionaler Ausdrücke abgeprüft werden. Der Schlußteil ergibt dann die Klassifizierung. Diese Form der Repräsentation findet sich z.B. bei CN2 und RIPPER (vgl. Unterkapitel 3.3).

Man kann zwei Arten dieser Repräsentation unterscheiden. Bei geordneten Regeln werden diese in einer festgelegten Reihenfolge durchlaufen, und die erste, deren Bedingungsteil zutrifft, wird ausgewertet. Ein Vorteil ist die sehr kompakte Darstellung. Ein Nachteil ist, daß

[2] Aus mathematischer Sicht ist die Ausdrucksmächtigkeit Propositionaler und Attributierter Logiken streng genommen gleich.

der Bedingungsteil jeder Regel implizit eine Negation der Bedingungsteile der vorhergehenden Regeln enthält, was ihre Verständlichkeit einschränkt.

Bei ungeordneten Regeln tritt dieses Problem nicht auf: alle zu einer Klassifikation notwendigen Bedingungen tauchen im Bedingungsteil der jeweiligen Regel auf. Das wird mit einer größeren Komplexität erkauft. Im allgemeinen sind solche Listen aber verständlicher.

Graphen

Eine Attributierte Logik läßt sich auch in Form eines Graphen darstellen. Dazu gehören z.B. Ansätze, die auf baumbasierten Verfahren beruhen, identische Teilbäume allerdings wieder „verkleben“ (vgl. z.B. KOHAVI UND LI (1995)).

Mit solchen Graphen versucht man die intuitive Verständlichkeit von Entscheidungsbäumen zu erhalten, aber die Komplexität der entstehenden Struktur zu verringern. Damit die Verständlichkeit durch die Verklebungsoperation allerdings nicht verringert wird sind evtl. Einschränkungen auf den generierten Knoten notwendig. Dazu kann beispielsweise gefordert werden, daß auf einer Ebene des Baumes nur Tests auf einem Attribut erlaubt sind.

Probleme

Repräsentationsprachen auf Grundlage Attributierter Logiken haben, obwohl in diversen Systemen mit Erfolg eingesetzt, nach HOLSHEIMER UND SIEBES (1994) einige Nachteile. Zwei davon sind:

1. **Ausdrucksmächtigkeit:** Die in pseudo-propositionalen Sprachen ausdrückbaren Konzepte sind stark eingeschränkt. Beziehungen zwischen Objekten oder Attributen können mit ihnen nicht beschrieben werden.

   Als Beispiel soll das Konzept einer Klasse der Personen mit „gleichem Vor- und Nachnamen“ gelernt werden. Mit Attributierten Logiken arbeitende Systeme können hierzu keine Generalisierung entwickeln, sondern müssen alle Beispiele aufzählen.

2. **Hintergrundwissen:** Bereits vorhandenes Wissen ist nach Ansicht von Holsheimer und Siebes nur sehr schwer in das System integrierbar. Solches Wissen wird daher häufig in Form von Constraints, also generellen Bedingungen, modelliert, die allerdings zu restriktiv seien.

   Einige regelbasierte Systeme, wie RJ (vgl. Unterkapitel 3.3), erlauben allerdings die Angabe einer Ausgangsmenge von Regeln. Weiterhin hat sich NÚÑEZ (1991) mit der Möglichkeit befaßt, Hintergrundwissen in Form von IS-A-Hierarchien in einen ID3-basierten Algorithmus namens EG2 zu integrieren.

2.5.2 Logiken höherer Ordnung

Die beschriebenen Probleme von pseudo-propositionalen Repräsentationssprachen können z.B. mit Logiken Erster Ordnung (First Order Logics, FOL) überwunden werden. Eine ganze Reihe von Verfahren benutzt Sprachen, die auf FOL beruhen.

Dazu gehören in erster Linie die Systeme der Induktiven Logik Programmierung ILP (vgl. Unterkapitel 3.4). Ihr Ziel ist es, ein in einer FOL repräsentiertes Programm zu erstellen, welches zusammen mit vorhandenem Hintergrundwissen die Trainingsmenge als logische Folgerung ergibt.

Ausdrucksmächtigkeit vs. Komplexität

FOL-basierte Systeme sind in der Lage, einfache Konzepte für Probleme zu generieren, die mit Verfahren, die auf Attributierten Logiken beruhen, entweder gar nicht oder nur mit einer wesentlich höheren Komplexität der Beschreibung zu generieren wären. Dadurch sinkt der Berechnungsaufwand, da einfachere Hypothesen gefunden werden können.

Auf der anderen Seite können mit einer ausdrucksmächtigeren Repräsentation auch komplexere Beschreibungen konstruiert werden, was den Suchraum vergrößert. Das wiederum führt zu einem höheren Berechnungsaufwand. Daher werden häufig eingeschränkte Varianten der Repräsentationssprache benutzt. So kann die Anzahl der Prädikate in den Schlußregeln eingeschränkt werden, ein Prädikat darf nur eine bestimmte Anzahl von Argumenten enthalten, oder rekursive Beschreibungen werden ausgeschlossen.

Beim Einsatz von FOL-basierten Systemen sollte immer der Trade-Off zwischen Ausdrucksmächtigkeit und Komplexität im Auge behalten werden. Wenn die zu generierenden Hypothesen in pseudo-propositionalen Sprachen beschrieben werden können, sollte auf den Einsatz von Logiken höherer Ordnung verzichtet werden. Entscheidet man sich für ein FOL-basiertes Systemen, sollte großer Wert auf die Entwicklung von Heuristiken gelegt werden, die das Lernen anleiten (vgl. Unterkapitel 2.6).

2.5.3 Neuronale Netze

Künstliche Neuronale Netze bestehen aus untereinander stark vernetzten, einfachen Einheiten. Auf die symbolische Repräsentation von Wissen wird verzichtet. Das in einem Konnektionistischen System vorhandene Wissen wird lediglich in Form eines Verhaltens sichtbar, so z.B. bei einem Konnektionistischen Klassifikationssystem eine bestimmte Ausgabe bei einer bestimmten Eingabe beobachtet werden kann (vgl. Unterkapitel 4.2).

Damit ist die Formulierung expliziter Hypothesen nur durch eine sich an das Lernen anschließende Analyse des Netzes möglich. Dies gelingt aber nur sehr eingeschränkt, vgl. hierzu auch die Bemerkungen in der Einleitung zu Kapitel 3. Die Integration von Hintergrundwissen in ein Konnektionistischen System ist kaum möglich (vgl. hierzu z.B. HOLSHEIMER UND SIEBES (1994)).

2.5.4 Instanzenbasierte Verfahren

Wenn es nicht darauf ankommt, eine explizite Formulierung der Hypothese zu generieren, jedoch auf andere Aspekte des symbolischen Lernens nicht verzichtet werden soll, bieten sich Instanzenbasierte Lernverfahren wie IB an (vgl. Unterkapitel 3.5.4).

Bei ihnen wird keine Abstraktion im eigentlichen Sinne vorgenommen. Vielmehr werden bereits gesehene Beispiele aus der Eingabemenge gespeichert, und eine Klassifikation neuer Beispiele erfolgt anhand einer Ähnlichkeitsbeziehung zu den gespeicherten Exemplaren.

2.6 Wissensaneignung

Gemeinhin wird Maschinelles Lernen als Suche beschrieben, so z.B. bei MICHALSKI (1983), MORIK (1995) und KUBAT *et al.* (1998). KAINDL (1994) identifiziert sogar große Teile der KI mit Suchproblemen. Er befindet sich damit aber in Widerspruch zu ENZINGER *et al.* (1994), die darauf hinweisen, wie wichtig andere Problemlösungsstrategien sind. Diese Diskussion ist gerade für die Entwicklung effizienter Lernalgorithmen wichtig, die in die Lage versetzt werden sollen, den Lernprozeß mit Hilfe von Hintergrundwissen zu steuern.

Für den vorliegenden Gegenstandsbereich ist m.E. allerdings die Identifikation mit dem im folgenden beschriebenen Suchproblem gerechtfertigt, da es allein darum geht, in dem Raum möglicher Beziehungen von Attributen die für die Klassifikation kausalen Beziehungen zu finden. Eine z.B. durch eine Wissensbasis angeleitete Beschränkung des Lösungsraumes findet dabei zumindest derzeit nicht statt.

Von der Eingabemenge $\mathcal{E}$ ausgehend kann ein Maschinelles Lernsystem eine große Anzahl unterschiedlicher Beschreibungen entwickeln. Einige dieser Beschreibungen sind schlicht falsch, andere sind (teilweise) richtig. Von den richtigen Beschreibungen können einige ungesehene Beispiele besser klassifizieren als andere. Wenn wir also ein Maß für die Qualität einer Beschreibung finden, so läßt sich das Lernproblem als ein Suchproblem beschreiben: finde in der Menge aller möglichen Beschreibungen $\mathcal{D}$ die beste Beschreibung B.

Der durch $\mathcal{D}$ aufgespannte Raum dürfte zumindest für realistische Probleme im allgemeinen zu groß sein, um in ihm eine vollständige Suche durchzuführen. KUBAT *et al.* (1998) illustrieren das sehr schön: bei einer Attributierten Logik als Repräsentationssprache und nur zehn Attributen mit jeweils fünf möglichen Werten gibt es $5^{10} = 9.765.625$ unterschiedliche Belegungen. Jede Untermenge dieser Belegungen bildet ein mögliches Beschreibungskonzept, das sind $2^{9.765.625}$ Konzepte.

Um mit dieser Komplexität zu arbeiten, werden beim Maschinellen Lernen im allgemeinen zwei Techniken kombiniert: die Induktion und eine heuristisch gelenkte Suche.

2.6.1 Induktion

Die Schlußregel der Induktion ist folgende: gegeben seien die Aussagen „Sokrates ist ein Mensch“, „Menon ist ein Mensch“ und „Phaidon ist ein Mensch“. Dazu die Aussagen „Sokrates ist sterblich“, „Menon ist sterblich“ und „Phaidon ist sterblich“. Die Ableitung der Aussage „Menschen sind sterblich“ wäre ein induktiver Schluß.

Er ist, wie man leicht sieht, nicht wahrheitserhaltend. Aus den Beispielen ließe sich ferner ableiten „Menschen sind Männer“, ein Schluß, der spätestens bei der Begegnung mit einer Frau ad acta gelegt werden muß.

KUBAT *et al.* (1998) erläutern das Prinzip des induktiven Lernens am Beispiel eines Außerirdischen, der auf der Erde landet, und seine Sprachkenntnisse aufbessern will. Er fragt einen Eingeborenen danach, was ein *Vogel* sei. Dieser zeigt ihm daraufhin eine *Amsel.* Der Außerirdische möchte dieses Beispiel wahrheitserhaltend *generalisieren* und verlangt daher ein Gegenbeispiel.

Der Mensch zeigt ihm einen *Hund.* Daraufhin wagt der Außerirdische einen ersten Schluß und erklärt die Eigenschaft, *Flügel* zu besitzen, zu einer notwendigen Eigenschaft von *Vögeln.* Zur Überprüfung fragt er den Einheimischen, ob *Fliegen* auch *Vögel* seien. Da dieser das verneint, betrachtet der Außerirdische seinen Schluß als *Übergeneralisierung*, und fügt zur *Spezialisierung* hinzu, daß *Vögel gelbe Schnäbel* besitzen.

So funktioniert im wesentlichen das Maschinelle Lernen. Man kann drei Vorgehensweisen bei der Induktion unterscheiden:

1. **Spezialisierer:** Beginne mit einem möglichst allgemeinen Konzept und verfeinere dieses schrittweise.
2. **Generalisierer:** Beginne mit einem möglichst speziellen Konzept und erweitere dieses schrittweise.
3. **Versionen-Raum:** Beginne mit einem beliebigen Konzept. Generalisiere, wenn notwendig. Spezialisiere, wenn notwendig.

Der kritische Punkt in Systemen des Maschinellen Lernens ist, jeweils die passende nächste Operation auszuwählen bzw. eine passende Generalisierung und Spezialisierung zu finden.

2.6.2 Heuristische Suche

Die Aufgabe einer Heuristik ist es, zu entscheiden, welcher von einer Menge an möglichen Schritten der am meisten Erfolg versprechende ist. Vorausgesetzt ist die Existenz einer *Qualitätsfunktion* Q. Ich stelle im folgenden kurz nach KUBAT *et al.* (1998) zwei Heuristiken vor. Eine genauere Grundlegung zu Suchtechniken findet sich z.B. bei CHARNIAK UND MCDERMOTT (1985) und SCHEFE (1991).

Best-First

1. Betrachte den derzeitigen Zustand Z als den *besten* und bilde ein Menge *derzeitiger Zustände* $\mathcal{Z}$, die nur aus Z besteht.
2. Wenn Z eine gegebene Abbruchbedingung erfüllt, halte an. Z ist eine Lösung des Problems.
3. Sonst wende alle derzeit anwendbaren Operationen auf den derzeitigen Zustand an. Füge die so erzeugten Zustände Z_i der Menge der *derzeitigen Zustände* $\mathcal{Z}$ hinzu.
4. Wähle anhand von Q einen Zustand Z_j aus $\mathcal{Z}$ aus. Mache ihn zum Zustand Z des Systems und gehe zu Schritt 2.

Dieser Algorithmus ist relativ speicheraufwendig, da er sich alle generierten Zustände merkt. Dieses Problem wird mittels der *Beam Search* überwunden.

Beam Search

1. Betrachte den derzeitigen Zustand Z als den *besten*.
2. Wenn Z eine gegebene Abbruchbedingung erfüllt, halte an.
3. Falls die Menge *derzeitiger Zustände* $\mathcal{Z}$ mehr als n Elemente enthält, verwerfe alle bis auf die besten n bzgl. Q.
4. Wende alle derzeit anwendbaren Operationen auf Z an. Füge die so erzeugten Zustände Z_i der Menge $\mathcal{Z}$ hinzu.
5. Wähle anhand von Q einen Zustand Z_j aus $\mathcal{Z}$ aus. Mache ihn zum Zustand Z des Systems und gehe zu Schritt 2.

Eine häufig angewandte *Beam Search*-Methode ist der *Hill Climber*, bei dem $n = 1$ gilt. Der Name soll an einen Bergsteiger erinnern, der von seiner jeweiligen Position immer einen Schritt in Richtung eines höheren Punktes macht, um den Gipfel zu erreichen.

In dieser Metapher wird gleich ein Problem derartiger Methoden aufgezeigt: der Bergsteiger kann auf einem kleinen Hügel landen, statt auf einem hohen Gipfel. Heuristische Methoden ohne globale Sicht (oder zumindest einen Lookahead) können leicht in globale Extrema laufen. Eine Methode, das zu verhindern, ist das Simulated Annealing.

Simulated Annealing

Simulated Annealing (simuliertes Ausglühen) ist eine Technik, bei welcher der Suchraum mit einer gewissen Zufälligkeit durchschritten wird. Dazu wird nicht unbedingt immer die Operation gewählt, die den höchsten Qualitätsgewinn verspricht, sondern die zur Auswahl stehenden Operationen werden alle mit einer gewissen Wahrscheinlichkeit gewichtet.

Diese Wahrscheinlichkeit verteilt sich so, daß die Auswahl von Operationen mit höherem Gewinn in Hinblick auf Q die besseren Chancen haben.

Welchen Einfluß diese Wahrscheinlichkeiten auf die Wahl haben, wird aber wiederum von einer weiteren Funktion gesteuert, der *Temperatur*. Wenn die Temperatur hoch ist, werden alle Operationen mit ungefähr der gleichen Wahrscheinlichkeit gewählt. Ist die Temperatur niedrig, werden Operationen bevorzugt, die einen höheren Qualitätsgewinn versprechen. In den Extremfällen ist die Wahl also rein zufällig (maximale Temperatur), oder es handelt sich um ein reines Hill Climbing Verfahren (minimale Temperatur).

Die Grundidee ist, mit einer hohen Temperatur zu starten, um diese im Verlauf der Suche immer weiter zu erniedrigen.

Der Name Simulated Annealing wird in der Literatur inzwischen nicht mehr allein für Beam Search Verfahren benutzt. RUMMEL (1997) benutzt den Begriff beispielsweise für beliebige

Verfahren, durch die störende Einflußgrößen ausgeglichen werden. So kann eine höhere Komplexität von Ausdrücken bei der Güteberechnung dieser Ausdrücke mit einem „Straffaktor“ belegt werden. In diesem weiteren Sinne habe ich Simulated Annealing auch in dieser Arbeit benutzt.

2.7 Vorgehensweise

Die Vorgehensweise der meisten Algorithmen des Maschinellen Lernens läßt sich grob zwei Gruppen zuordnen: den Divide-and-Conquer-Methoden und den sogenannten AQ-ähnlichen Methoden.

2.7.1 Divide-and-Conquer

Wie in vielen Problembereichen der Informatik haben sich auch beim Maschinellen Lernen Verfahrensweisen herausgebildet, die das Gesamtproblem in mehrere kleine, einfachere Grundprobleme zerlegen. Dies läßt sich besonders schön bei der Top-Down Induction of Decision Trees (TDIDT) sehen.

Bei der Erzeugung von Entscheidungsbäumen ist ein Grundprinzip, durch Tests auf einzelnen Attributen die Eingabemenge in immer kleinere Mengen aufzuspalten, bis diese (möglichst) nur noch aus Exemplaren einer Klasse bestehen. Dabei wird zumeist ein Top-Down Verfahren ohne Lookahead gewählt. Aufgrund eines häufig informationstheoretisch begründeten lokalen Optimierungskriteriums wird ein Attributtest unter allen möglichen ausgewählt. In den so gebildeten Tochterknoten wird das Verfahren wiederholt (eine ausführlichere beispielhafte Beschreibung findet sich im Unterkapitel 3.2.1).

Eine Fallgrube bei diesem Vorgehen ist das sogenannte Overfitting, also eine Überanpassung an die Datenmenge. Es kann sein, daß ein Zweig aufgrund von Störfaktoren in der Datenmenge zustande kam und nur bestimmte Sonderfälle abdeckt. Ein solcher Zweig kann die Leistung eines Baumes zur Klassifizierung neuer Beispiele verschlechtern. Außerdem erhöht er aus Sicht eines Klassifikationsverfahrens unnötig die Komplexität des Baumes. Bei der im vorliegenden Fall gewollten Generierung von Hypothesen können allerdings auch in solchen überangepaßten Teilbäumen relevante Kausalbeziehungen zu finden sein.

In einem solchen Fall greift man zum Pruning. Pruning heißt wörtlich Beschneidung, und wie der Gärtner schlechte Triebe beschneidet, so werden im Maschinellen Lernen überangepaßte Zweige eines Entscheidungsbaumes entfernt. Dies kann einerseits in der Phase des Baumaufbaus passieren (Forward Pruning), andererseits kann aber auch der fertige Baum beschnitten werden (Post Pruning).

2.7.2 AQ-ähnlich

Diese Vorgehensweise hat ihren Namen von den ersten verbreiteten Algorithmen, welche diese Technik anwendeten (vgl. Unterkapitel 3.3.1). Die grundsätzliche Vorgehensweise ist nach KUBAT *et al.* (1998) wie folgt:

1. Die Beispiele aus der Eingabemenge $\mathcal{E}$ werden in die Menge der positiven Beispiele $\mathcal{P}$ und die Menge der negativen Beispiele $\mathcal{N}$ aufgeteilt. Beginne mit einer (evtl. leeren) Regelmenge $\mathcal{R}$.

2. Zufällig oder mittels einer Heuristik wird ein Beispiel aus $\mathcal{P}$ als sogenannter *Samen (Seed)* genommen.

3. Finde eine Menge von generalisierten, maximalen Regeln $\mathcal{R}_S$ zu diesem *Samen* mit $\mathcal{R}_S = \{R_1, R_2, \ldots, R_n\}$. Dabei beschreibt für maximale Regeln $\mathcal{N}$ die Grenze: Eine Regel R_i aus $\mathcal{R}_S$ darf kein Element aus $\mathcal{N}$ abdecken. Die so erhaltene Menge wird *Stern* genannt.

4. Wähle anhand eines *Präferenzkriteriums* eine beste Regel R_j aus.

5. Füge R_j zu $\mathcal{R}$ hinzu. Überdeckt $\mathcal{R}$ alle Beispiele aus $\mathcal{P}$, so halte an. Im anderen Fall entferne alle überdeckten Beispiel aus $\mathcal{P}$ und fahre mit Punkt 2. fort.

Die einzelnen Verfahren unterscheiden sich vor allem durch das Auswahlkriterium und das Verfahren, mit dem aus der Menge der verbleibenden positiven Beispiele eine Menge generalisierter Regeln erzeugt wird.

2.8 Problembereiche

Zum Abschluß dieser Betrachtungen möchte ich noch einige Problembereiche des Maschinellen Lernens skizzieren.

2.8.1 Fehlende Werte

Die dem Lernalgorithmus präsentierten Werte müssen nicht immer vollständig sein. Auch wenn in verschiedenen Datensätzen Werte für einige Attribute fehlen, so sollte ein System zum Maschinellen Lernen in der Lage sein, eine Klassifikation zu erstellen.

Zum Handling fehlender Werte sind nach QUINLAN (1986) z.B. folgende Vorgehensweisen denkbar:

1. Der von Igor Kononenko vorgeschlagene „Bayes-guess“ wobei fehlende Attributwerte anhand eines vorgeschalteten Bayes-basierten Klassifizierers geschätzt werden.[3]

2. Die Einführung eines neuen Attributwertes „unknown“, weil das Fehlen eines Wertes auch eine wichtige Information beinhalten kann. Es kann evtl. Gründe dafür geben, daß der Wert fehlt, diese Gründe können für die Klassifizierung wichtig sein.

3. Der „straightforward guess“, bei welchem der Wert über eine einfache Häufigkeitsverteilung der Attributwerte errechnet wird.

[3]Dabei wird mittels eines Bayes'schen Verfahrens (s. Unterkapitel 3.5.3) eine bedingte Wahrscheinlichkeit für die Klassifizierung in Bezug auf die Attributwerte ermittelt und der in dieser Klasse am häufigsten vorkommende Wert als Schätzung für den fehlenden Attributwert genommen.

Daß fehlende Werte ein schwieriges Problem darstellen, macht z.B. folgende Aussage aus (FRIEDMAN *et al.*, 1996, S. 718) deutlich: „Friedman estimated that about half the code in CART and about 80% of the programming effort went into missing values!“

Die vorliegende Arbeit ist eingebettet in ein Framework, bei dem nicht direkt auf empirischem Datenmaterial gearbeitet wird, sondern dieses vielmehr genutzt wird, um ein Simulationsmodell zu treiben. Die dort entstehenden, prinzipiell vollständigen Daten bilden die Arbeitsgrundlage für das hypothesenbildende System. Die vorgeschaltete Modellbildung kann somit als gute Datenvorverarbeitung für Systeme des Maschinellen Lernens angesehen werden, vgl. hierzu beispielsweise KÖSTER UND SONNENSCHEIN (1999).

Ich möchte daher das Vorschalten eines Simulationsmodells der oben genannten Liste möglicher Vorgehensweisen hinzufügen. Ich denke, daß ein validiertes Simulationsmodell mindestens zu vergleichbaren Ergebnissen führen wird. Hierzu wären allerdings genauere Untersuchungen notwendig.

2.8.2 Rauschen

Unter Rauschen versteht man im allgemeinen nicht-systematische Störgrößen auf Attributwerten. Sie können in zwei Situationen Probleme bereiten: beim Training und bei späteren Klassifikationen.

QUINLAN (1986) identifiziert für die Arbeit mit verrauschten Datensätzen in der Trainingsphase folgende Problembereiche:

1. Die Arbeit mit nicht adäquaten Attributwerten. Rauschen kann dazu führen, daß für die kausalen Beziehungen wichtige Attribute auf den ersten Blick wenig zur Klassifikation beitragen.
2. Ausführung des Prunings, also des Herausnehmens von überspezialisierten Bereichen, die zwar Sonderfälle richtig erkennen, aber für eine erfolgreiche Klassifikation ungesehener Beispiele wenig beitragen.

Ein interessanter Effekt ist, daß Klassifikationssysteme, die mit unverrauschten Datensätzen aufgebaut wurden, ein schlechteres Klassifikationsverhalten zeigen als Klassifikationssysteme, die auch während des Lernens mit Rauschen konfrontiert waren (vgl. HOLSHEIMER UND SIEBES (1994), QUINLAN (1986)). Eine genauere Diskussion findet sich z.B. auch bei BRATKO *et al.* (1996).

Bei der vorliegenden Problemstellung kann sich Rauschen vor allem über die Modellierung der Simulation einstellen, wobei jedoch auch hierzu nähere Untersuchungen notwendig sind. Zumindest ist eine Anforderung an das in dieser Arbeit vorgestellte System, mit verrauschten Datensätzen arbeiten zu können.

2.8.3 Größe der Datenbasis

Häufig wird die Anforderung, mit Datenmengen arbeiten zu können, die nicht mehr im Hauptspeicher eines Computers vorgehalten werden können, als Unterscheidungsmerkmal des Data

Mining von Klassifikationsverfahren des Maschinellen Lernens genannt (vgl. z.B. WROBEL (1998)). Zum Teil ist dies allerdings nur eine Frage des Standpunktes.

QUINLAN (1993) schreibt dazu auf S. 57: „When I started work on ID3 in the late 1970s, computers with virtual memory were uncommon and programs were usually subject to size restrictions. The training set of those early experiments were quite large – one had 30,000 cases described by 24 attributes – and exceeded my memory allowance on the machine I was using. Consequently, there was a need to explore indirect methods of growing trees from large datasets."

Fenster

Die von Quinlan gewählte Abhilfe bestand darin, eine *Fenster* genannte, zufällige Stichprobe aus den Daten auszuwählen, und mit dieser einen Baum aufzubauen. Dieser Baum wurde dann auf die anderen Datensätze angewandt. Eine Auswahl der Beispiele, die falsch klassifiziert wurde, wurde dem *Fenster* hinzugefügt und der Prozeß iteriert. Das Abbruchkriterium war, daß der Baum keines der nicht gezeigten Beispiele falsch klassifizierte. Das so entstandene *Fenster* war zumeist immer noch kleiner als die Gesamtmenge. Der Effekt wird darauf zurückgeführt, daß durch die Iteration nur die „interessanten" Fälle in den Lerndatensatz aufgenommen werden.

Dieses Verfahren wird, mit einigen kleineren Änderungen, auch heute noch beim C4.5-Verfahren (vgl. Unterkapitel 3.2.1) eingesetzt. Dabei erweisen sich solche Anpassungen auch noch aus anderer Perspektive als positiv, so verlangt C4.5 nicht mehr, daß alle Datensätze richtig klassifiziert werden, um Überanpassung zu vermeiden.

Auch bei anderen Verfahren des Maschinellen Lernens wird das *Fenster*-Verfahren in der einen oder anderen abgewandelten Form genutzt. Es ist allerdings bei Verfahren des automatischen Clustering, wie des k-means Clusterings (vgl. Unterkapitel 4.3), kaum anwendbar. Allerdings können hier unterschiedliche Attribute unabhängig voneinander bearbeitet werden.

Relevante Attribute

Ein anderes Problem bei realen Data Mining Aufgaben ist häufig die große Anzahl an Attributen. Dabei kann es vorkommen, daß viele dieser Attribute für die Lösung der Klassifikationsaufgabe irrelevant sind. Mit Techniken, eine Untermenge relevanter Attribute zu finden, befassen sich z.B. JOHN *et al.* (1994).

2.9 Bewertung

Nachdem in den letzten Unterkapiteln ein Abriß des Maschinellen Lernens gegeben worden ist, möchte ich noch einmal auf den anfangs bereits angesprochenen Unterschied zwischen der vorliegenden Arbeit und Arbeiten zum Maschinellen Lernen allgemein zurückkommen.

Systeme des Maschinellen Lernens, wie sie vor allem auch im Bereich des Knowledge Discovery in Databases eingesetzt werden, haben zumeist eine Klassifizierung von Daten als

Ziel. Dieses Ziel wird mit unterschiedlichen Verfahren und auf Grundlage unterschiedlicher Repräsentationen erreicht.

2.9.1 Hypothesenbildung vs. Klassifikation

In der vorliegendenden Arbeit geht es nicht darum, ein Klassifikationsinstrument für neue Fälle an die Hand zu bekommen. Ziel ist letztlich die Erstellung einer expliziten Hypothese über kausale Zusammenhänge. Damit sind auch nur solche Verfahren anwendbar, die einen in symbolischer Form vorliegenden Klassifikator entwickeln. In diesem Klassifikator sollten sich Ursache-Wirkungs-Beziehungen insofern wiederfinden lassen, als das ein Klassifikationssystem diejenigen Momente an den Lernobjekten herausarbeitet, die eine eindeutige Klassifikation ermöglichen. Dazu gehören genau solche Momente, die in kausaler Beziehung zur Klassenzugehörigkeit stehen.

Aber es gibt auch einen sehr wichtigen darüber hinaus weisenden Unterschied. Durch Maschinelles Lernen entwickelte Klassifikatoren sollen effektiv anwendbar sein. Das heißt insbesonders, daß sie ein gutes Vermögen zur Generalisierung haben. Dazu werden im allgemeinen Pruning-Techniken eingesetzt, die Überspezialisierungen verhindern sollen.

Bei der Analyse von Kausalzusammenhängen kann allerdings in solchen Spezialfällen eine wichtige Information stecken. In dem hier betrachteten Anwendungsgebiet ist die Fähigkeit zur Generalisierung dem präzisen Erhalten spezieller Informationen unterzuordnen.

2.9.2 Occam's Razor

Maschinelle Lernverfahren versuchen zumeist, eine in gewisser Hinsicht minimale Klassifikation aufzubauen. Dahinter steckt eine in der Literatur selten explizierte Annahme, die auf den Philosophen Wilhelm von Ockham zurückgeht. Sie ist als „Occam's Razor" bekannt und lautet angelehnt an BLUMER *et al.* (1987) wie folgt: seien zwei Erklärungen eines Phänomens gegeben. Dann ist die einfachere der beiden Erklärungen als gültig anzunehmen.

In der Welt des Maschinellen Lernens taucht diese Annahme in dem Ziel der meisten Algorithmen auf, die einfachste Hypothese zu finden, welche mit den Lerndaten übereinstimmt. Der Sinn einer solchen Heuristik ist jedoch stets unter Betrachtung der konkreten Anwendung zu diskutieren.

Erkenntnistheoretische Kritik

Aus erkenntnistheoretischer Sicht ist Ockhams Annahme zumindest problematisch. Eine Hypothese über einen Sachverhalt ist der Versuch, wesentliche Bestimmungen des Sachverhaltes auszudrücken. Diese Hypothese muß in sich und im Kontext der ihr zugrundeliegenden Theorie widerspruchsfrei sein.

Der Inhalt ist nur als Übereinstimmung der Hypothese mit den beobachteten Erscheinungen zu fassen. Da sich das Wesen der Dinge nur über die Erscheinungen mitteilt, die Hypothese aber wesentliche Momente erfassen soll, so bleibt Ockhams Annahme ihr äußerlich, und taugt daher nicht als Kriterium für die Wahrheit einer Hypothese.

Empirische Befunde

MURPHY UND PAZZANI (1994) berichten über Versuchsreihen, in denen sie unterschiedlich komplexe Entscheidungsbäume über Lerndatenmengen aufgebaut haben. Als Maß der Komplexität gilt dabei die Anzahl der Knoten. Danach wurde jeweils geprüft, welcher dieser Bäume bisher ungesehene Datensätze am besten klassifiziert d.h. mit der geringsten Fehlerrate.

Die Autoren kommen zu dem Schluß, daß Bäume, die etwas größer sind als die minimalen Bäume, die beste Klassifikationsgenauigkeit liefern. Dies steht allerdings nicht im Widerspruch zu Occam's Razor, sondern eher zu dessen vereinfachter Übertragung auf das Gebiet des Maschinellen Lernens.

Die Suche nach einem minimalen Konzept ist nach BLUMER *et al.* (1987) häufig NP-vollständig. Effektiv arbeitende Algorithmen benutzen im allgemeinen eine Annäherung an die eigentlich zu entwickelnde Theorie. Diese Annäherung ist als *Probably Approximately Correct (PAC) Learning* bekannt, vgl. hierzu VALIANT (1984) und KEARNS UND SHAPIRE (1992).

Die Betonung der Suche minimaler Konzepte im Bereich des Maschinellen Lernens ist somit als problematisch einzustufen. Dies gilt, wie von Murphy und Pazzani gezeigt, bereits für Klassifikationsaufgaben und um so mehr noch für das Anliegen dieser Arbeit. Aus diesem Grund sollte genau untersucht werden, inwieweit Verfahren des Maschinellen Lernens bei der Entwicklung von Hypothesen gerade wegen ihres Bestrebens, minimale Theorien zu entwickeln, Kausalzusammenhänge „übersehen“bzw. „vergessen“.

Kapitel 3

Überblick: Ansätze Maschinellen Lernens

*The first obvious fact about human learning is that it's horribly slow (*Simon*, 1983, S. 26).*

3.1 Einleitung

In diesem Kapitel werde ich einen Streifzug durch bereits publizierte Verfahren des Maschinellen Lernens unternehmen. Ich konzentriere mich dabei auf Lösungen aus dem symbolischen Zweig der KI, da Ansätze z.B. aus dem Bereich des Konnektionismus aufgrund der Schwierigkeiten bei der Interpretation der generierten Klassifikationsabbildung für das von mir behandelte Problem nicht in Frage kommen.

Es gibt zwar Untersuchungen zur Interpretierbarkeit Neuronaler Netze, aber zumindest die mir bekannten arbeiten entweder mit stark eingeschränkten Netztopologien und eignen sich somit nur für relativ eingeschränkte Problemklassen, oder haben mit prinzipiellen Schwierigkeiten bei der Interpretation komplexerer Strukturen zu kämpfen.

So beschränken sich Waschulzik *et al.* (1993) z.B. auf ein Single-Layer-Perceptron mit Neuronen angelehnt an das McCulloch-Pitts-Modell[1]. Dabei lernt in einem ersten Schritt ein Neuronales Netz einen Klassifikator, der dann in einem zweiten Schritt durch Analyse der entwickelten Gewichte interpretiert wird. Die Ausdrucksmächtigkeit von Single-Layer-Netzen ist allerdings beschränkt, vgl. hierzu die Arbeiten von Minsky und Papert (1969) oder (kürzer) Wasserman (1989). Waschulzik *et. al.* hatten zudem mit der Schwierigkeit zu kämpfen, daß Datenerhebungsartefakte überhaupt erst im zweiten Schritt entdeckt wurden.

Demgegenüber steht Vaughn (1996) stellvertretend für die Probleme beim Verstehen komplexer Netze. Ihr Analyseansatz für Multi-Layer-Perceptrons leitet die Interpretation direkt aus

[1]Das theoretische Modell ist genauer beschrieben in Waschulzik und Geiger (1990b), das SENN-Verfahren (Strukturierte Entwicklung Neuronaler Netze) in Waschulzik und Geiger (1990a). Grundlegende Erläuterungen zum McCulloch-Pitts-Modell finden sich in (Ritter *et al.*, 1991, S. 25).

den Aktivitäten der Hidden Layer ab, insofern diese Hinweise auf die wichtigsten Komponenten der Eingabewerte geben. Aus dieser (recht trivialen Erkenntnis) leitet sie Beziehungen der Ein- und Ausgabedaten ab, ohne allerdings eine entsprechende mathematische Fundierung zu geben. Der Ansatz ist somit nur für Problembereiche handhabbar, bei denen die Plausibilität der gefundenen Interpretation direkt überprüfbar ist. Dies dürfte vor allem bei großen Netzen ein recht mühsamer Prozeß sein.

Einen ganz anderen Weg gehen Craven und Shavlik (1996). Sie nehmen ein austrainiertes Neuronales Netz und die Lerndatenmenge, mit der es trainiert wurde. Aus beidem generieren sie einen Entscheidungsbaum. Da es sich hierbei aber nicht um eine Analyse des Netzes handelt, bleibt das Grundproblem der Interpretierbarkeit bestehen: man bekommt zwar einen besser verständlichen Klassifikator, kann aber nicht erkennen, an welcher Stelle z.B. Datenerhebungsartefakte zu einem fehlerhaft klassifizierenden Netz geführt haben.

Einzig kooperative Fuzzy-NN-Systeme gehen in die Betrachtung ein, da bei ihnen die Interpretation von Neuronen und Verbindungen als Fuzzy-Regeln bzw. Zugehörigkeitsfunktionen möglich ist, vgl. hierzu beispielsweise Kruse und Nauck (1996).

Zu jedem der im folgenden vorgestellten Verfahren des Maschinellen Lernens gibt es eine längere textuelle Beschreibung und eine kurze Zusammenfassung in tabellarischer Form. Ferner wird jeweils eine kurze Einschätzung der Systeme im Hinblick auf ihre Eignung für den Einsatz im vorliegenden Anwendungsgebiet gegeben. Im Vordergrund steht dabei die Frage: „Sind die Verfahren zur Bildung expliziter und kommunizierbarer Hypothesen in der Lage?“

Zu beachten ist weiterhin, daß kaum ein Algorithmus des Maschinellen Lernens mit Attributen in Form von Zeitreihen arbeiten kann. Zusätzlich muß daher eine geeignete Vorverarbeitung gefunden werden. Diese Frage werde ich gesondert im Kapitel 4 zum Clustering behandeln.

Die einzelnen Unterkapitel sind so geordnet, daß Algorithmen, die im weiteren Verlauf der vorliegenden Arbeit eine größere Rolle spielen, oder die grundlegende Konzepte für eine ganze Klasse von Verfahren enthalten, zu Beginn erläutert werden. Im Anschluß daran werden weitere Systeme genannt, die m.E. aus unterschiedlichen Gründen nicht für den Einsatz im vorliegenden Gegenstandsbereich in Frage kommen.

3.2 Baumbasierte Verfahren

Die im folgenden vorgestellten Verfahren arbeiten mit einer in Form von Bäumen dargestellten Attributierten Logik als Repräsentationssprache (vgl. Kapitel 2.5.1).

3.2.1 ID3-Familie

Die vielleicht am besten untersuchte Familie von Algorithmen zum Maschinellen Lernen ist die ID3-Familie von Quinlan, die auf Arbeiten von Hunt zum Konzepterwerb aufbaut, die in den frühen sechziger Jahren publiziert wurden. Vom selben Autor stammen die Erweiterungen C4.5 und C5, von Igor Kononenko die Variante Assistant.

ID3, Assistant, ID5R, C4.5, C5, MC4	
Verfahren:	Baumerzeugend. Nicht inkrementell, aber in Utgoff (1989) wird die inkrementelle Variante ID5R beschrieben. Die Attribute werden nach informationstheoretischen Überlegungen ausgewählt. C4.5 ist der Nachfolger von ID3 und als Referenzmodell weit verbreitet. Inzwischen ist die kommerzielle Weiterentwicklung C5 auf dem Markt. MC4 ist eine im akademischen Bereich freie Implementierung von C4.5.
Gegeben:	Menge von Beispielen in Attribut-Wert-Repräsentation. Die Attribute dürfen nur bestimmte Ausprägungen annehmen, im besonderen sind keine kontinuierlichen Attribute zugelassen. Der Nachfolger C4.5 kennt auch kontinuierliche Attribute, sie werden lokal (d.h. in den jeweiligen Knoten) diskretisiert.
Ziel:	Entscheidungsbaum zur Klassifikation neuer Objekte.
Suche:	Top-Down ohne Recovery.
Störungen:	Statistische Verfahren für Rauschen und fehlende Werte.
Wissen:	Explizit im Baum, Wenn-Dann-Regeln.
Literatur:	ID3: Quinlan (1983), Quinlan (1986), Holsheimer und Siebes (1994), kurz auch: (Morik, 1995, S. 258), Assistant: Kononenko (1993), ID5R: Utgoff (1989), C4.5: Quinlan (1993), C5.0: Quinlan (1998) MC4: Kohavi und Sommerfield (1996)

Tabelle 3.1: Eigenschaften der ID3-Familie

ID3

Die wohl am häufigsten zitierte Veröffentlichung, in der die Grundlagen von ID3 erläutert werden, ist Quinlan (1983). Interessant an diesem Text sind vor allem die ausführliche Schilderung des Beispiels „Matt in n Zügen“ nebst Betrachtungen über das Lernen aus wenigen Beispielen sowie Überlegungen zum automatischen Finden von Attributen.

Die Beschreibung in Quinlan (1986) ist allerdings eher geeignet, die Grundlagen des Verfahrens zu verstehen. Hier findet sich eine detaillierte Beschreibung des ID3-Algorithmus. Neben dem grundsätzlichen Vorgehen werden folgende Problembereiche behandelt (vgl. auch Unterkapitel 2.8):

1. Handling von Rauschen,
2. Handling unbekannter Attributwerte und
3. die Bestimmung eines Auswahlkriteriums für die Wurzel des Baumes.

Der beschriebene mathematische und informationstheoretische Hintergrund ist auch für andere Verfahren von oft maßgeblicher Bedeutung.

Eines der größten Handicaps von ID3 ist, daß der Algorithmus nicht in der Lage ist, kontinuierliche Attributwerte zu bearbeiten. Oftmals wird daher ein Preprocessing der Art durchgeführt, daß solche Attribute vorab diskretisiert werden[2].

Weiterhin generiert die Urfassung von ID3 strenge Regeln, d.h. daß der Baum solange weiter aufgebaut wird, bis in einem Blatt nur eine Klasse zu finden ist. Inkonsistente Beispiele würden in einem Blatt landen, daher wird vorausgesetzt, daß die Klassen paarweise disjunkt sind.

Das Suchverfahren ist ein top-down-Verfahren ohne lookahead und ohne recovery. Es garantiert, daß ein einfacher Baum gefunden wird, wenn auch nicht notwendigerweise der einfachste, da das Verfahren in lokale Gütemaxima hineinlaufen kann. Ein Baum wird wie folgt aufgebaut:

1. Ein Attribut wird als Wurzel genommen, und für jede mögliche Ausprägung wird ein Tochterknoten generiert.
2. Der Baum wird benutzt, um die Lernmenge zu klassifizieren. Wenn alle Datensätze in einem Tochterknoten nur einer Klasse angehören, wird dieser Knoten zu einem Blatt und mit der Klasse beschriftet. Sind alle Töchter Blätter oder fertig aufgebaute Teilbäume, terminiert der Algorithmus.
3. Für diejenigen Töchter, die keine Blätter sind, wird der Algorithmus rekursiv angewandt. Knoten in unteren Ebenen testen nur Attribute, die im entsprechenden Pfad noch nicht getestet wurden. Die Lerndatenmenge jedes Tochterknotens umfaßt die Exemplare, deren getestetes Attribut die entsprechende Werteausprägung hat.

Die Auswahl des jeweils zu testenden Attributes erfolgt aufgrund einer informationstheoretischen Heuristik. Es wird das Attribut ausgewählt, das für die Tochterknoten den höchsten Informationsgewinn bewirkt. Die Heuristik lautet im einzelnen:

Seien zwei zu erkennende Klassen definiert: P sei die Menge der positiven, N die Menge der negativen Beispiele.[3] Die Menge S von Beispielen enthalte p Exemplare aus P und n aus N. Dann ist die Informationsmenge in bits, die für die Entscheidung benötigt wird, ob ein beliebiges Beispiel s aus S zu P oder N gehört, definiert durch:

$$I(p,n) = -\frac{p}{p+n}\log_2\frac{p}{p+n} - \frac{n}{p+n}\log_2\frac{n}{p+n} \tag{3.1}$$

Nehmen wir jetzt an, die Wahl des Attributes A in der Wurzel teile die Menge der Beispiele S auf in die Menge $\{S_1, S_2, \ldots, S_k\}$. Falls nun S_i p_i Beispiele aus P und n_i Beispiele aus N enthält, so ist die entsprechend notwendige Informationsmenge für die Entscheidung, ob ein beliebiges Element aus S_i zu P oder N gehört, $I(p_i, n_i)$. Damit ist die Information, die unter der Wahl des Attributes A als Wurzel zur Klassifikation eines Elementes aus S notwendig ist, definiert als gewichteter Durchschnitt über alle k Unterbäume S_i:

[2]Dabei werden kontinuierliche Wertebereiche in Intervalle aufgeteilt. Diese Intervalle stellen dann nominale Werte für das diskretisierte Attribut dar.

[3]Die ursprüngliche ID3-Variante konnte nur zwei Klassen erkennen. Eine verallgemeinerte Fassung findet sich in Gleichung 3.3

$$E(A) = \sum_{i=1}^{k} \frac{p_i + n_i}{p + n} I(p_i, n_i) \tag{3.2}$$

Dann wird das Attribut A mit dem höchsten Informationsgewinn gewählt, d.h. das Attribut, für welches $E(A)$ minimal ist.

Ich möchte die Betrachtung an dieser Stelle abbrechen, obwohl das vorgestellte Maß die bekannte Eigenschaft hat, Attribute mit vielen Ausprägungen zu bevorzugen. Für die Diskussion von alternativen Maßen verweise ich auf Quinlan (1986).

ASSISTANT

Von Anfang an stand die Unfähigkeit von ID3, Attribute mit kontinuierlichem Wertebereich zu verarbeiten, im Mittelpunkt von Forschungsarbeiten. Bereits 1984 publizierte Kononenko seine Erweiterung Assistant, durch welche der Algorithmus in die Lage versetzt wurde, Attribute zum Zeitpunkt des Baumaufbaus zu diskretisieren (so arbeitet z.B. auch die $\mathcal{MLC}$++-Implementation von ID3).

Weiterhin wurden in Assistant erstmals Techniken zum Pruning des Baumes implementiert. Unter Pruning versteht man das Beschneiden des Baumes, d.h., daß solche Teilbäume abgeschnitten werden, die eine Überspezialisierung darstellen. Man unterscheidet Pre-Pruning Techniken (Pruning während des Aufbaus des Baumes) und Post-Pruning Techniken (hierbei wird der fertige Baum beschnitten). In Assistant sind beide Arten verwirklicht.

C4.5, C5, MC4

Der ID3-Nachfolger C4.5 aus Quinlan (1993) baut auf diesen Arbeiten auf und kann ebenfalls Attribute mit kontinuierlichem Wertebereich nutzen. Hier wird die Diskretisierung als Teil des Baumaufbaus angesehen, d.h., daß die Umwandlung in diskrete Attribute in den Knoten durchgeführt wird. Quinlan erhofft sich durch die Ausnutzung lokaler Eigenschaften (da nur noch die Attributwerte zur Diskretisierung herangezogen werden, die in einem Teilbaum noch vorkommen) ein verbessertes Klassifikationsverhalten. Allerdings hat sich dies nach Dougherty *et al.* (1995), einer umfassenden Untersuchung von Verfahren zur Diskretisierung kontinuierlicher Attribute, nicht bestätigt.

Ein weiterer Vorteil von C4.5 ist die Fähigkeit, Verzweigungen auf Mengen von Attributausprägungen statt auf einzelnen Werten durchzuführen. Das kann die Komplexität des Baumes verringern und den Aufbau weitgehend identischer Unterbäume („replication problem“) vermindern. Pruning-Techniken sind ebenfalls integriert.

Im Gegensatz zu ID3 erlaubt C4.5, dasselbe Attribut in einem Pfad mehrfach zu testen. Dies ist vor allem im Hinblick auf die lokale Diskretisierung von Attributen mit kontinuierlichem Wertebereich sinnvoll, da die entstehenden Intervalle in einem Teilbaum anders aussehen können als in einem im Pfad vorhergehenden Knoten.

C4.5 ist eigentlich kein einzelner Algorithmus, sondern eine Sammlung von Programmen zum Maschinellen Lernen (so sagt es auch der Titel des Buches). In vielen Veröffentlichungen wird

Verfahren:	Baumerzeugend, nicht inkrementell.
Gegeben:	Vektoren von Attributwerten (nominal, linear, strukturiert).
Ziel:	Entscheidungsbaum analog → C4.5, wobei an den Knoten Tests auf Multidimensionale Attribute (wie beispielsweise Zeitreihen) stehen können.
Wissen:	Explizit in Form eines Baumes.
Suche:	Analog → C4.5.
Störungen:	Analog → C4.5.
Bemerk.:	Es handelt sich um eine Ergänzung des bewährten C4.5-Systems zur Bearbeitung multidimensionaler Attribute.
Literatur:	RUMMEL (1997)

Tabelle 3.2: Eigenschaften von ADSEQ

C4.5 zwar synonym für das entsprechende baumerstellende Programm benutzt, tatsächlich zählen aber z.B. auch das Verfahren C4.5*rules*, welches C4.5-Bäume in Regeln übersetzt, und die Klassifikationsprogramme CONSULT (für Bäume) und CONSULTR (für Regeln) zum Paket.

Resumee

Nicht zuletzt, weil C4.5 in QUINLAN (1993) umfassend beschrieben ist, wird das Verfahren sehr häufig als Referenz oder als Grundlage von Erweiterungen (vgl. → ADSEQ) benutzt. Da das Ergebnis des Basisprogrammes, ein Entscheidungsbaum, relativ gut kommunizierbar und zumindest in einer graphischen Aufarbeitung auch von menschlichen Experten gut nachvollziehbar ist, habe ich sowohl ID3 als auch das in $\mathcal{MLC}$++ implementierte C4.5-Derivat MC4 in meiner Arbeit benutzt.

MC4 unterscheidet sich von C4.5 nur durch die Grundeinstellung von Lernparametern und ist im Gegensatz zu C4.5 im akademischen Bereich frei verfügbar (die Lizenz von C4.5 ist an das entsprechende Lehrbuch gekoppelt). Auf die Einbeziehung der neuesten Variante der Familie, C5, habe ich verzichtet, da es ein kommerzielles Produkt ist (allerdings gibt es eine frei verfügbare eingeschränkte Version). Außerdem unterscheidet sich nach Aussage von QUINLAN (1998) C5 v.a. in der Laufzeit von C4.5. Sowohl C4.5 als auch C5 werden aber von den $\mathcal{MLC}$++-Utilities unterstützt.

3.2.2 ADSEQ

Das ADSEQ-Verfahren (Algorithm for Data SEQuences) von RUMMEL (1997) ist eines der wenigen Systeme, die direkt auf strukturierten Attributen arbeiten können. Verfahren des symbolischen Maschinellen Lernens tun sich im allgemeinen eher schwer mit derartigen Attributen wie z.B. Zeitreihen. Da diese allerdings in realen Problembereichen häufig vorkommen, gibt es einige Ansätze zum Preprocessing derartiger Daten. Dazu sind nach RUMMEL (1997) zu zählen:

- Die Auswahl eines „most significant“ Attributes, (z.B. des letzten Meßwertes).
- Das Finden einer symbolischen Beschreibung (z.B. des Aussehens einer Zeitreihe).
- Funktionale Gruppierung interdependenter Attribute in einem einzigen Attribut (z.B. Mittelwertbildung über Zeitreihen).
- Aufhebung der Abhängigkeit (z.B. wird jeder Wert wiederholter Messungen als neues Attribut genommen).
- Aufhebung der Individuen (z.B. begründet jede einzelne einer ganzen Reihe von Messungen ein neues Individuum).

Bei keiner der genannten Methoden handelt es sich um einen wissensgesteuerten Ansatz, und alle sind mit Informationsverlust verbunden. Daher geht Rummel einen anderen Weg. Die zugrunde liegende Idee ist, vereinfacht gesagt, die möglichen Wertebelegungen solcher strukturierter Attribute (er nennt sie Multi-Events) an den Knoten des Entscheidungsbaumes lokal verschiedenen Klassen zuzuordnen. Während bei einem Attribut mit kontinuierlichem Wertebereich bei → C4.5 in den Knoten die Auswahl des passenden Nachfolgeknotens anhand einer geeigneten Diskretisierung gefunden wird, so geschieht dies im vorliegenden Fall über die Zugehörigkeit der Wertebelegung zur jeweiligen Klasse.

Als Klassifikationsverfahren benutzt Rummel dabei ein aus der Mustererkennung stammendes, subsymbolisches, unüberwachtes Clusteringverfahren, das k-means Clustering (vergleiche Unterkapitel 4). Das Clustering wird lokal, d.h. in den jeweiligen Knoten durchgeführt, um lokale Eigenschaften auszunutzen.

Eine Anwendung von Adseq in der Kardiologie wird in Dürer *et al.* (1997) vorgestellt. Hierbei handelt es sich um eine vergleichende Studie der Einsetzbarkeit von baumbasierten Verfahren und Neuronalen Netzen in der Medizin. Die Interpretierbarkeit von Bäumen wird als Vorteil angesehen. Demgegenüber hätte das verwendetet Neuronale Netz den Vorteil, nicht nur qualitative, sondern auch quantitative Aussagen liefern zu können.

Resumee

Adseq, Algorithm for Data SEQuences, ist eines der wenigen Verfahren des Maschinellen Lernens, das in der Lage ist, ohne weitere Vorverarbeitung mit in sich strukturierten Attributen zu arbeiten. Zugleich stützt es sich mit der Anbindung an → C4.5 auf eines der am besten untersuchten und erfolgreichsten traditionellen Verfahren ab.

Auf der anderen Seite haben Dougherty *et al.* (1995) für die Diskretisierung kontinuierlicher Attribute bereits gezeigt, daß keine signifikante Performance-Verbesserung durch lokale statt globale Diskretisierung erreicht wird, sondern daß die Art des Diskretisierungsverfahrens von wesentlich größerer Bedeutung ist.

Da das Clustering ein aufwendiger Prozeß ist (quadratisches Laufzeitverhalten), sollte untersucht werden, ob ein vor den symbolischen Verarbeitungsschritt geschaltetes subsymbolisches Verfahren wie das von Rummel gewählte k-means Clustering eine ähnliche Performance liefert.

Verfahren:	Baumerzeugend, nicht inkrementell.
Gegeben:	Menge von Objekten in Attribut-Wert-Repräsentation.
Ziel:	Entscheidungsbaum zur Klassifikation neuer Objekte mit zusätzlichen „Option Nodes“. Dabei kann an einem Knoten mehr als ein Attribut getestet werden.
Suche:	Greedy.
Störungen:	Sowohl Rauschen wie fehlende Werte werden behandelt.
Wissen:	Explizit im Baum.
Literatur:	KOHAVI UND KUNZ (1997)

Tabelle 3.3: Eigenschaften von OPTIONDT

3.2.3 OptionDT

Reguläre Entscheidungsbäume erlauben in jedem Knoten den Test genau eines Attributes. Dabei wird dieses Attribut anhand eines bestimmten Gütekriteriums ausgewählt, welches z.B. informationstheoretisch motiviert sein kann. Das Gütekriterium variiert von Verfahren zu Verfahren.

Die in einem Knoten getroffene Wahl ist dabei entscheidend für den gesamten Teilbaum unter ihm. Da die meisten Verfahren ohne lookahead arbeiten, kann eine „mißglückte“ Wahl des Attributes (wie sie z.B. aufgrund verrauschten Datenmaterials oder einer in Bezug auf die Verteilung der Klassen ungünstigen Stichprobe auftreten kann) die Performance des Systems wesentlich beeinflussen.

Verschiedene Verfahren sind daher diskutiert worden, das Verhalten von Entscheidungsbaumverfahren in kritischen Situationen zu verbessern. Sie laufen häufig auf sog. Voting-Verfahren hinaus, d.h., daß mehrere Exemplare von Bäumen, die z.B. mit unterschiedlichen Stichproben des Lerndatensatzes erzeugt werden, eine neue Instanz klassifizieren und letztlich die Klasse gewählt wird, für die sich eine Mehrheit der Bäume ausspricht[4].

KOHAVI UND KUNZ (1997) gehen einen anderen Weg: sie schlagen auf Arbeiten von Buntine beruhend vor, daß Entscheidungsbäume auf den ersten Ebenen statt einfacher Knoten sog. Optionsknoten einfügen können. Ein solcher Optionsknoten unterscheidet sich von einem gewöhnlichen Knoten in einem Entscheidungsbaum dadurch, daß nicht auf ein Attribut getestet wird, sondern auf mehrere verschiedene Attributen.

De facto hängt damit an einem derartigen Knoten nicht ein Unterbaum, sondern für jedes der gewählten Attribute ein Unterbaum, und in jedem Optionsknoten wird eine zu klassifizierende Instanz von jedem dieser Unterbäume klassifiziert. Die Klasse wird dann letztlich nach dem Mehrheitsprinzip aus der Klassifikation dieser Unterbäume ausgewählt (Buntine schlägt gewichtete Wahl vor).

[4]Ich habe leider im Rahmen dieser Arbeit nicht die Möglichkeit, die Verfahren genauer auszuführen, und möchte daher beispielhaft auf BREIMAN (1996) sowie besonders auf den in einer der nächsten Ausgaben von *Machine Learning* erscheinenden Artikel von BAUER UND KOHAVI (Unveröffentlicht) verweisen.

Verfahren:	Baumbasiert, nicht inkrementell.
Gegeben:	Menge von Beispielen in Attribut-Wert-Repräsentation.
Ziel:	Entscheidungsbaum zur Klassifikation oder Regression.
Suche:	Heuristischer Hill Climber.
Störungen:	Kein explizites Handling von Rauschen und fehlenden Werten.
Wissen:	Explizit im Baum.
Literatur:	Breiman *et al.* (1984)

Tabelle 3.4: Eigenschaften von Cart

Es wird vorgeschlagen, statt eines einfachen Knotens Optionsknoten einzufügen, wenn die von einem anderen Attributtest erreichte Qualitätssteigerung (gemessen an der Gütefunktion des Algorithmus) nur um einen bestimmten (variablen) Faktor schlechter ist als die des am besten abschneidenden Attributes.

Als Vorteil gegenüber Voting-Verfahren sehen Kohavi und Kunz an, daß die grundlegende Struktur – ein einziger Baum – erhalten bleibt, und die Klassifizierung damit besser kommunizierbar sei. Außerdem würden Wahlknoten nur eingefügt, wenn dies die Performance des Baumes erhöhen würde.

Resumee

Die OptionDT stellen für den vorliegenden Gegenstandsbereich insofern eine interessante Alternative dar, als das in einem Baum jeweils mehrere Hypothesen gleichzeitig dargestellt werden. Dabei kommt es nicht auf die (mehrheitlich bestimmte) Klassifikation von Beispielen an, sondern darauf, daß jeder Optionsknoten Informationen über verschiedene mögliche Kausalbeziehungen repräsentiert. Es wäre aber zu untersuchen, ob die Verständlichkeit und Kommunizierbarkeit nicht zu sehr leiden.

3.2.4 CART

Der Cart-Algorithmus (Classification And Regression Trees) von Breiman *et al.* (1984) ist ein Standardverfahren des Maschinellen Lernens mittels Bäumen. Sowohl ein Klassifikations- als auch ein Regressionsansatz sind in diesem Verfahren integriert. Ferner verfügt das System über Pruning-Techniken zur Verhinderung überspezialisierter Bäume.

Maschinelles Lernen wird als Problem der Statistik betrachtet, dementsprechend wird der zugrunde liegenden Mathematik in der Veröffentlichung ein großer Stellenwert beigemessen. Die verwendeten Bewertungsfunktionen werden ausführlich beschrieben. Sie sind in ähnlicher Form auch auf verwandte Verfahren anwendbar.

In Breiman *et al.* (1984) finden sich Beispiele je einer Anwendung aus der Medizin und der Physik (Massenspektographie). Bei Bentrup und Ray (1993) wird Cart zusammen mit anderen Verfahren zur Klassifikation von Schlafphasen eingesetzt.

Verfahren:	Lazy Decision Tree: Aufbau von Pfaden während der Klassifizierung.
Gegeben:	Menge von Objekten in Attribut-Wert-Repräsentation, bestehend aus der Trainingsmenge und einer neu zu klassifizierende Instanz. Attributwerte müssen diskret sein.
Ziel:	Klassifikator aus Algorithmus, Trainingsmenge und gecachten Pfaden früherer Klassifizierungen.
Suche:	Hill Climber.
Störungen:	Fehlende Werte werden nicht zur Klassifikation benutzt, kein Pruning, daher Probleme mit Rauschen.
Wissen:	Implizit im bereits aufgebauten Baum und dem Verfahren zum Klassifizieren neuer Instanzen.
Literatur:	FRIEDMAN *et al.* (1996)

Tabelle 3.5: Eigenschaften von LazyDT

Resumee

Eine Weiterentwicklung von CART wird als kommerzielles Produkt vertrieben. Mit → ID3-basierten Verfahren liegen mir andere ausgereifte baumbasierte Algorithmen in zumindest für den akademischen Bereich freien Versionen vor, daher habe ich auf eine weitere Analyse dieses Ansatzes verzichtet. Da CART aber von den $\mathcal{MLC}$++-Utilities unterstützt wird, wäre ggf. eine empirische Untersuchung ohne Änderungen am prototypisch implementierten System möglich.

3.2.5 LazyDT

Einige Algorithmen des Maschinellen Lernens verzichten auf den Aufbau einer expliziten Lösung in Form eines Klassifikators, so z.B. die Nearest Neighbour Verfahren oder Ahas → IB-Familie.

Solche Verfahren werden im allgemeinen als „lazy“ bezeichnet, da sie die eigentliche Arbeit so lange herausschieben, bis sie unumgänglich geworden ist, sprich: eine Klassifikation erstellt werden muß. Obwohl teilweise bezweifelt wird, daß bei derartigen Systemen überhaupt von Lernen die Rede sein kann (der Begriff des Lernens ist in der KI-Community fast so umstritten wie bei den Kollegen aus der Psychologie resp. der Kognitionswissenschaft), so gelten sie doch zumeist als Teil des Forschungsgebietes Maschinelles Lernen.

LazyDT ist ein Beispiel dafür, daß ein „lazy“ Vorgehen auch beim Aufbau eines Entscheidungsbaumes möglich ist, indem praktisch nur der Pfad für eine zu klassifizierende Instanz aufgebaut wird, und gespeicherte Informationen aus der Trainingsmenge dazu benutzt werden, diesen aufzubauen.

Verfahren:	Nicht inkrementelle Induktion von Entscheidungsbäumen.
Gegeben:	Menge von Objekten in Attribut-Wert-Repräsentation.
Ziel:	Entscheidungsbaum zur Klassifikation neuer Objekte mit höchstens zwei Layern. Das Verfahren ist garantiert Agnostisch PAC.
Suche:	Greedy.
Störungen:	Sowohl Rauschen wie fehlende Werte werden behandelt.
Wissen:	Explizit im Baum.
Literatur:	Auer *et al.* (1995)

Tabelle 3.6: Eigenschaften von T2

Resumee

Da es keine kommunizierbare Hypothese über die Klassenzugehörigkeiten gibt, ist das Verfahren für meine spezielle Problemstellung nicht geeignet, und hier nur beispielhaft für ähnliche Algorithmen erwähnt.

3.2.6 T2

T2 ist ein weiterer Algorithmus zur Induktion von Entscheidungsbäumen. Es werden Bäume mit höchstens zwei Layern generiert, was die Anwendung auf große Probleme nach Untersuchungen von Kohavi *et al.* (1997) stark einschränkt. Auer *et al.* (1995) heben dagegen hervor, daß ihr Verfahren garantiert „Agnostisch PAC“ sei (s.u.). Dabei handelt es sich um eine auf Überlegungen von Valiant (1984) zur Möglichkeit Maschinellen Lernens[5] aufgebaute Definition von Kearns und Shapire (1992) der Klasse von Agnostisch PAC Lernalgorithmen.

Agnostisch PAC bedeutet in diesem Zusammenhang, daß aus den Eingabedaten ungeachtet der Stichprobe garantiert und ohne a priori Annahmen über die zu generierende Theorie und die Lerndatensätze eine fast optimale Theorie (d.h. in diesem Fall ein Entscheidungsbaum) generiert wird, sofern die Stichprobe ausreichend groß ist.

Gegenüber PAC-Algorithmen hat ein Agnostisches PAC-Verfahren den Vorteil, ohne a priori Wissen über die Klassifikation der Beispiele auszukommen, d.h. im besonderen, daß keine Annahmen über Häufigkeitsverteilungen in der Trainingsmenge gemacht werden müssen.

Der zweite Unterschied zeigt sich in der Semantik des erreichten Konzeptes: die „Zielfunktion“ ist kein Ideal, dem sich beliebig genau angenähert werden kann (vereinfacht ausgedrückt in der Fehlerrate des Klassifikators gegenüber einer „Idealklassifizierung“, die dafür aber formal beschreibbar sein muß), sondern eine Wahrscheinlichkeitsfunktion.

[5] Eine Theorie des Lernens mit dem Ziel der Konstruktion von Theorien, die „Propably Approximately Correct (PAC)“ sind.

Verfahren:	Regelerzeugend, inkrementell.
Gegeben:	Vektoren von Attributwerten (nominal, linear, (hierarchisch) strukturiert).
Ziel:	Entscheidungsregeln mit einer logischen Formel als Voraussetzung. Repräsentationssprache ist VL1, eine logische Sprache mit Variablen.
Wissen:	Explizit in Form von Regeln.
Suche:	Heuristische irrevocable Beam Search.
Störungen:	Nur durch Pre- und Postprocessing, z.B. Rule Truncation.
Bemerk.:	Domain Knowledge in Form von Regeln nutzbar. Konstruktive Induktion von Features, die in der Ausgangsmenge nicht vorkommen.
Literatur:	Michalski *et al.* (1986), Holsheimer und Siebes (1994)

Tabelle 3.7: Eigenschaften der AQ-Familie

Resumee

Trotz der theoretischen Bedeutung (oder besser: der Bedeutung in der Theorie) habe ich in meinen Experimenten das T2-Verfahren nur am Rande beobachtet (es ist problemlos über die $\mathcal{ML}$C++-Utilities zu nutzen), da mir die anfangs gemachte Einschränkung auf kleine Problembereiche bei der zu erwartenden Komplexität epidemiologischer Hypothesen problematisch erscheint.

3.3 Regelbasierte Verfahren

In diesem Unterkapitel beschäftige ich mich weiterhin mit Systemen, die auf Attributierten Logiken arbeiten. Im Unterschied zum vorhergehenden Unterkapitel handelt es sich aber um regelerzeugende Verfahren (vgl. Unterkapitel 2.5.1).

3.3.1 AQ

Das ML-System AQ15 generiert Entscheidungsregeln mit einer logischen Formel in der Voraussetzung. Die Regeln werden in der in Dietterich und Michalski (1983) genauer beschriebenen Logiksprache VL1 repräsentiert. VL1 ist eine mehrwertige, attributierte Logik mit getypten Variablen. Ein *Selektor* ist dabei eine Bedingung, die ein Attribut einem Wert oder einer Disjunktion von Werten zuordnet. Ein *Komplex* ist eine Konjunktion von Selektoren. Der Bedingungsteil einer Entscheidungsregel ist ein Komplex.

Es findet eine heuristische Beam-Suche im Raum der möglichen logischen Ausdrücke statt, um passende Komplexe zu finden, die alle positiven und keine negativen Beispiele überdecken. Es werden also strenge Regeln generiert, was besagt, daß eine Regel alle positiven und keine negativen Beispiele überdeckt. Im Gegensatz dazu stehen probabilistische (wahrscheinlichkeitstheoretische) und possibilistische (unscharfe) Regeln. AQ kann nur durch Pre- und Postprocessing mit Rauschen und inkonsistenten Daten umgehen.

Werden mehrere gültige Regeln gefunden, so wird die signifikanteste (im Sinne eines benutzerdefinierten, anwendungsabhängigen Maßes) ausgewählt.

Eine interessante Eigenschaft des Systems besteht in der *konstruktiven Induktion* neuer *Features*. Unter Features versteht die Mustererkennung klassischerweise relevante Gruppierungen von Attributen, die so ein neues Attribut bilden, welches keine direkte Entsprechung in der Eingabemenge hat. AQ15 kann sowohl neue logische als auch neue numerische Variablen als Funktionen vorhandener Variablen definieren, wobei Anwendungswissens einbezogen wird, das in Form von Regeln abgelegt ist. Vom System selber generierte Regeln werden der bereits vorhandenen Regelmenge hinzugefügt.

Eine empirische Betrachtung des Verfahrens findet sich in Michalski *et al.* (1986). Der Algorithmus wird auf drei medizinische Problembereiche angesetzt: Lymphography, Prognose für die Heilung von Brustkrebs und Lokalisierung primärer Tumore. Die Daten kommen von der Uniklinik Ljubljana und werden häufig für solche Studien herangezogen.

Ein Vergleich mit der Performance des → ID3-basierten Systems Assistant soll die Wirkung der (dem Pruning in baumbasierten Verfahren vergleichbare) „Truncation“ von Regeln aufzeigen. Diese Truncation resultiert in einer signifikanten Verringerung der Anzahl und Komplexität der Regeln ohne einen Verlust an Genauigkeit. Dazu werden sukzessiv die Komplexe aus der Regelmenge entfernt, welche die wenigsten positiven Beispiele abdecken, da diese möglicherweise eine Überspezialisierung darstellen, bis die Vorhersagegenauigkeit der Regeln gegenüber neuen Fällen abnimmt.

Resumee

Durch die Fokussierung auf das Auffinden exakter Regeln ist das Handling von Rauschen schlecht in den eigentlichen Algorithmus integrierbar. Ist die Lernmenge inkonsistent, enthält also sich widersprechende Beispiele, so kann mit strenger Induktion überhaupt keine Regel generiert werden. AQ bietet in solchen Fällen drei Möglichkeiten: die inkonsistenten Beispiele werden alle als positive oder alle als negative Beispiele aufgefaßt, oder sie werden ignoriert.

Der auf AQ aufbauende Algorithmus → CN2 ist insgesamt ausgereifter, auf die weitere Untersuchung von AQ habe ich daher verzichtet.

3.3.2 CN2

Die Grundfassung von CN2 wird in Clark und Niblett (1989) beschrieben. Dabei wird CN2 explizit als „system designed for the efficient induction of simple, comprehensible induction rules in domains where problems of poor description language and/or noise may be present“ vorgestellt. Dies wird durch Änderungen an den Algorithmen der → AQ-Familie erreicht, durch welche Pruning-Techniken des → ID3-Verfahrens in ein regelerzeugendes System übernommen werden. Dadurch soll die Effizienz des baumbasierten Verfahrens mit der guten Kommunizierbarkeit von Wenn-Dann-Regeln kombiniert werden.

Als Ausgabe erzeugt CN2 eine geordnete Liste von Regeln, d.h. daß die erste zutreffende Regel gilt. Falls keine der generierten Regeln zutrifft, so wird auf die mehrheitlich vorhandene Klasse geschlossen. Die Regeln ähneln den Regeln von → AQ, allerdings besteht der Bedingungsteil

Verfahren:	Regelerzeugend, inkrementell.
Gegeben:	Menge von Beispielen in Attribut-Wert-Repräsentation, kontinuierliche Wertebereiche sind zugelassen.
Ziel:	Sortierte Liste von Wenn-Dann-Regeln. Dabei handelt es sich im Gegensatz zu → AQ nicht um Disjunktionen von Komplexen, sondern um „elementare" Komplexe. Die erste Version liefert eine geordnete Liste von Regeln, die spätere Version von CLARK UND BOSWELL (1991) erzeugt eine ungeordnete Liste.
Suche:	Beam Search, prinzipiell mit Möglichkeit zur Recovery.
Störungen:	Angelehnt an → ID3 werden Verfahren zum Handling von Rauschen und fehlenden Werten eingebaut.
Wissen:	Explizit in Regeln.
Bemerk.:	Basierend auf der AQ-Familie werden Vorgehensweisen von → ID3 übernommen.
Literatur:	CLARK UND NIBLETT (1989), CLARK UND BOSWELL (1991), BOSWELL (1990), HOLSHEIMER UND SIEBES (1994)

Tabelle 3.8: Eigenschaften von CN2

aus elementaren Komplexen, und nicht aus einer Disjunktion von Komplexen. Im Lernprozeß werden für jede gefundene Regel diejenigen (positiven und negativen) Beispiele aus der Menge der Beispieldatensätze entfernt, die bereits überdeckt werden.

Während des Suchprozesses werden gefundene Komplexe spezialisiert, indem konjunktive Selektoren hinzugefügt oder Disjunktionen in Selektoren entfernt werden (zur Def. von Selektoren → AQ). Dabei werden alle möglichen Spezialisierungen erzeugt. Aus so gefundenen Komplexen werden anhand zweier Tests, einer auf *Signifikanz* und ein weiterer auf *Genauigkeit*, diejenigen ausgewählt, die der Regelmenge hinzugefügt werden sollen.

Die Genauigkeit wird über die informationstheoretisch fundierte Berechnung der Entropie bewertet. Dazu ist zuerst die Menge E von Beispielen zu finden, die von einem gegebenen Komplex überdeckt werden, und die Wahrscheinlichkeitsverteilung $P = (p_1, p_2, \ldots, p_n)$ der Beispiele in E über die n Klassen zu berechnen. Danach ist die *Entropie* definiert wie folgt:

$$\textit{Entropie}\,(E) = -\sum_{i=1}^{n} p_i \log_2(p_i) \tag{3.3}$$

Je geringer die Entropie, um so höher die Genauigkeit. Dieses Maß bevorzugt Komplexe, die eine große Menge von Beispielen überdecken und wenig negative. Damit erzeugt CN2 im Gegensatz zu → AQ keine strengen Regeln, sondern possibilistische.

Die Signifikanz einer Regel ist gleichfalls ein wichtiges Kriterium, um nicht unnötig Regeln für einige wenige Sonderfälle in die generierte Menge aufzunehmen. Als Maß der Signifikanz gilt die Wahrscheinlichkeits-Rate, welche die durch die Regel abgedeckte relative Häufigkeit

der Klassen mit der in der Lernmenge insgesamt zu beobachtenden Häufigkeitsverteilung in Beziehung setzt. Es sei $F = (f_1, f_2, \ldots, f_n)$ die unter Anwendung der Regel zu beobachtende Verteilung und $E = (e_1, e_2, \ldots, e_n)$ die Grundverteilung. Dann ist die Wahrscheinlichkeits-Rate definiert durch:

$$WR = 2 \sum_{i=1}^{n} f_i \log(f_i/e_i) \tag{3.4}$$

In KONONENKO (1993) wird auch ein empirischer Vergleich des konzipierten Systems mit dem ID3-basierten System ASSISTANT, einer einfachen → AQ-Variante und → Bayes Tabellen in drei medizinischen Problemfeldern vorgenommen (den gleichen, die auch in MICHALSKI *et al.* (1986) benutzt werden). Jedes der benutzten Verfahren wird dabei grob beschrieben.

Verbesserungen

Eine Modifikation des CN2 Algorithmus' in zwei Punkten wird in CLARK UND BOSWELL (1991) vorgeschlagen:

1. Eine Laplace-Funktion ersetzt die ursprünglich vorgeschlagene Entropie-Funktion. Die Entropie-Funktion bevorzugt Regeln, die wenige Fälle abdecken. Ein ursprünglich eingeführter Signifikanz-Test fängt dieses Verhalten nur unzureichend ab.

   Die neue Funktion lautet:

   $$Laplace = (k_c + 1)/(k_{tot} + n) \tag{3.5}$$

   mit n Anzahl der Klassen, k_c Anzahl der Beispiele aus der Klasse c, für die die Regel gilt, und k_{tot} die Gesamtzahl der Fälle, für die die Regel gilt.

   Eine allgemeine Fassung der Regel wird in CESTNIK UND BRATKO (1991) diskutiert.

2. Als Ausgangsmenge wird nicht mehr eine geordnete Liste von Regeln generiert, sondern eine ungeordnete. Diese ist von menschlichen Experten leichter zu lesen, weil nicht immer die implizite Negation der Bedingungsteile der vorhergehenden Regeln beachtet werden muß. Dazu werden jeweils nur die überdeckten positiven und nicht auch die negativen Beispiele aus dem Lerndatensatz entfernt.

Resumee

Die von CN2 generierten Regeln sind besonders in der modifizierten Version (ungeordnete Menge von Regeln) einfach, verständlich und kommunizierbar. Das integrierte Handling von Rauschen (Aufbau probabilistischer Regeln) ist einfach und konsistent gelöst. Der Algorithmus, der bereits seit fast zehn Jahren bekannt ist, wird auch heute noch häufig in empirischen Untersuchungen eingesetzt. Auch in der vorliegende Arbeit wird CN2 benutzt.

Verfahren:	Regelerzeugend, nicht inkrementell.
Gegeben:	Menge von Beispielen in Attribut-Wert-Repräsentation, kontinuierliche Wertebereiche sind zugelassen.
Ziel:	Liste von Wenn-Dann-Regeln.
Suche:	Heuristisch gesteuerte Greedy Search.
Störungen:	RIPPER ist explizit für das Lernen in verrauschten Anwendungsgebieten und mit fehlenden Werten konzipiert.
Wissen:	Explizit in Regeln.
Literatur:	COHEN (1995)

Tabelle 3.9: Eigenschaften von RIPPER

3.3.3 RIPPER

RIPPER ist ein regelerzeugendes System, welches explizit für das Lernen in verrauschten Anwendungsgebieten und mit fehlenden Werten konzipiert worden ist. Es skaliert ungefähr linear mit der Anzahl der Trainingsbeispiele und ist daher nach Ansicht seiner Entwickler gut für die Arbeit mit großen Datenmengen geeignet. RIPPER setzt auf dem ebenfalls noch zu erläuternden IREP-Verfahren auf und ergänzt dieses durch ein optimierendes Postprocessing.

Suche (IREP)

Für den Lernprozeß von induktiven Lernverfahren wird der Trainingsdatensatz häufig zweigeteilt. Der erste Teil (Trainingsmenge) dient zum Aufbau einer Hypothese, bei der davon auszugehen ist, daß diese überangepaßt ist. Daher wird mit dem zweiten Teil der Trainingsdaten (Pruningmenge) ein Pruning-Schritt durchgeführt, der diese Überanpassung durch Entfernen von Regeln bzw. Teilbäumen rückgängig machen soll.

Ein Nachteil derartiger „overfit-and-simplify"-Verfahren ist, daß sie im allgemeinen sehr laufzeitaufwendig sind (im Bereich $O(n^4)$). COHEN (1995) schlägt daher die Modifikation eines als „incrementel error pruning" bekannten Verfahrens vor.

Zuerst wird mit einem „greedy" Verfahren eine Regelbasis erzeugt, indem nach jeder gefundenen Regel die damit überdeckten Beispiele aus der Lernmenge entfernt werden (vgl. → CN2). IREP benutzt hierzu eine propositionale Variante einer Strategie, die auch bei → FOIL Anwendung findet.

Eine Regel wird nur dann akzeptiert, wenn ihre (informationstheoretisch hergeleitete) Beschreibungskomplexität nicht über einem Faktor von k bits über der bisher kürzesten Regel liegt (Cohen gibt den Wert von k mit 64 an). Nachdem eine Regel gefunden wurde, wird diese sofort beschnitten, indem der am wenigsten signifikante Teil entfernt wird, sodaß das im folgenden betrachtete Gütemaß maximiert wird.

Wir betrachten ein Klassifikationsproblem mit zwei Klassen. Sei *PrunePos* die Menge der positiven Beispiele im Pruning-Datensatz, *PruneNeg* die der negativen. Sei ferner P (bzw.

N) die Anzahl der Beispiele in *PrunePos (PruneNeg)* und p (n) die Anzahl der der positiven (negativen) Beispiele, die von der betrachteten *Regel* überdeckt werden. Dann ist das Gütemaß definiert wie folgt:

$$v(Regel,\ PrunePos,\ PruneNeg) = \frac{p + (N - n)}{P + N} \tag{3.6}$$

Implementiert ist eine verallgemeinerte Variante, die auch mehr als zwei zu klassifizierende Klassen zuläßt.

Optimierung (RIPPER)

Nachdem eine Regelbasis gefunden wurde, wird diese von RIPPER einer Revision unterzogen. Dazu wird für jede der gefundenen Regeln R_i aus der Regelbasis $(R_1, R_2, \ldots, R_m)$ (in der Reihenfolge ihrer Erzeugung) geprüft, ob eine der folgenden Modifikationen die Fehlerrate der gesamten Regelbasis minimiert:

1. R_i wird ersetzt durch eine neu gebildete Regel R_i^r (Ersetzung),
2. R_i wird durch eine durch Hinzufügen von Bedingungen zu R_i gebildete Regel R_i^m ersetzt (Modifikation) oder
3. R_i bleibt, wie sie ist.

Dieses Postprocessing wird wiederum von der informationstheoretischen Heuristik geleitet, die auch den den Abbruch der Regelerzeugung steuert.

Resumee

Cohen gibt die Leistungsfähigkeit seines Algorithmus' als vergleichbar mit modernen baumbasierten Verfahren wie → C4.5 und dessen regelerzeugendem Postprocessing-Tool → C4.5*rules* an. In der Tat sind die von ihm veröffentlichten Befunde in Hinblick auf die Fehlerraten auch in schwierigen artifiziellen und natürlichen Problembereichen vielversprechend. Gleichzeitig zeigt IREP/RIPPER ein besseres Laufzeitverhalten als → C4.5.

Insgesamt halte ich RIPPER für ein Verfahren, daß schon aufgrund seiner guten Skalierbarkeit sowohl gegen moderne baumbasierte Algorithmen wie auch gegen das Referenzsystem für Regelerzeuger → CN2 getestet werden sollte, wie in der vorliegenden Arbeit geschehen.

3.3.4 RJ, RELAX, JoJo

Der Algorithmus RELAX ist ein generalisierendes, regelerzeugendes Lernverfahren. Ausgehend z.B. von in Form von Regeln bereitgestelltem Hintergrundwissen versucht RELAX (bzw. die heuristische Variante H-RELAX), die Regelbasis solange zu verallgemeinern, wie dadurch keine Gegenbeispiele, sondern nur positive Beispiele aus einer Lernmenge überdeckt werden. Dadurch wird eine minimale Regelmenge für die zu entwickelnde Hypothese generiert.

Verfahren:	Inkrementelles Erzeugen von Regeln.
Gegeben:	Menge von Objekten in Attribut-Wert-Relation.
Ziel:	Menge minimaler Regeln zum Überdecken positiver Beispiele. Es existiert auch eine FOL-basiert Variante JoJo-FOL.
Suche:	In RJ ist in der Lernprozedur JoJo sowohl das RELAX-Verfahren zum Generalisieren von Regeln als auch ein Spezialisierer analog zu → AQ und → CN2 (Beam Search) implementiert. RELAX arbeitet mit vollständiger Suche, die Erweiterung B-RELAX mit Beam Search.
Störungen:	Rauschen: Durch einen Bayes-Classifier falsch klassifizierte Objekte des Lerndatensatzes werden weggelassen. Fehlende Werte: Zwei Möglichkeiten: 1. Weglassen oder 2. Bestimmung der Wahrscheinlichkeit bestimmter Werte analog ASSISTANT (→ ID3).
Wissen:	Explizit in den erzeugten Regeln.
Literatur:	RJ: FENSEL (1993a), FENSEL *et al.* (1993), RELAX: FENSEL UND KLEIN (1991), FROG: FENSEL UND WIESE (1994), JoJo: FENSEL UND WIESE (1993), FENSEL (1993b), JoJo-FOL: WIESE (1996)

Tabelle 3.10: Eigenschaften von RJ, RELAX, JoJo

Der in FENSEL UND WIESE (1993) und FENSEL (1993b) beschriebene Algorithmus JoJo vereint top-down-Suche (Spezialisierung) und bottom-up-Suche (Generalisierung) in einem System. Es gibt einen Generalisierer analog zu RELAX, einen Spezialisierer und einen Scheduler, der jeweils den nächsten anzuwendenden Schritt (Generalisierung oder Spezialisierung) auswählt.

Für den Spezialisierer sind zwei Vorgehensweisen denkbar: zum einen kann ein zusätzlicher Attributtest eingefügt werden. Zum anderen kann die alte Regel aufgespalten werden in mehrere neue, die zusammengenommen die alten positiven Beispiele abdecken, aber keine negativen. Das letztere Verfahren wird präferiert, da es den Suchraum nicht zu stark beschneidet und somit Überanpassung vermeidet.

JoJo kann mit beliebigen genauen Regeln starten, da er sowohl Spezialisierungen wie Generalisierungen dieser Regeln überprüft. Er kann daher z.B. auch zum Pruning für Regeln eingesetzt werden, die von anderen Algorithmen erzeugt wurden. JoJo ist in das in FENSEL (1993a) und FENSEL *et al.* (1993) beschriebene RJ-System eingebunden. In RJ integriert sind ferner statistische Verfahren, wie Clusteranalyse, Test auf Kollinearität, Verfahren zum Schätzen unbekannter Werte und Bayes-Klassifikatoren.

Um die Ausdrucksmächtigkeit zu erhöhen, wurde eine → ILP-Variante von JoJo namens JoJo-FOL (für First Order Logic) entwickelt. JoJo-FOL ist in WIESE (1996) beschrieben. Dort findet sich auch ein Vergleich mit → FOIL.

Im FROG-Verfahren, vgl. FENSEL UND WIESE (1994), werden die beiden Suchrichtungen Generalisierung und Spezialisierung von JoJo ergänzt um eine dritte Dimension. Dazu wird in einem Suchschritt eine bestimmte Anzahl von Prämissen der Regeln ausgetauscht. FROG

Verfahren:	Generalized Rule Induction, Top Down (Spezialisierung).
Gegeben:	Menge von Objekten in Attribut-Wert-Repräsentation, Attributwerte diskret.
Ziel:	Menge der k besten Regeln, welche Beziehungen zwischen verschiedenen Attributwerten beschreiben, k benutzerdefiniert. Es wird mit Wahrscheinlichkeitsverteilungen statt mit deterministischen Klassifikationen gearbeitet.
Suche:	Sowohl Suche als auch Auswahl der besten Regeln werden mit einer informationstheoretischen Heuristik gesteuert.
Störungen:	Sowohl Rauschen wie fehlende Werte werden behandelt.
Wissen:	Explizit in der Regelmenge.
Literatur:	Smyth und Goodman (1992)

Tabelle 3.11: Eigenschaften von Itrule

„springt“ also im Suchraum anstatt in Einzelschritten vorzugehen.

Resumee

Eine Regelmenge, von der ausgehend generalisiert werden könnte, ist in meinem Anwendungsfall nicht vorhanden. Trotzdem wäre es sicherlich interessant zu sehen, wie ein kombiniertes System aus Spezialisierer und Generalisierer sich im Verhältnis zu reinen Spezialisiern verhält.

Da mir im Gegensatz zu vielen anderen interessanten Verfahren keine Implementierung vorlag, habe ich allerdings vorerst von der Arbeit mit diesen Verfahren abgesehen. Dieter Fensel hat mir allerdings in einer persönlichen Notiz zugesichert, das System für Tests im vorliegenden Gegenstandsbereich zugänglich zu machen. Dazu wären aber sicher kleine Abänderungen an dem im Rahmen dieser Arbeit prototypisch implementierten System notwendig.

3.3.5 ITRULE

Bei Itrule handelt es sich um einen weiteren regelerzeugenden Induktionsalgorithmus. Der Name leitet sich aus der Nutzung informationstheoretischer Überlegungen bei der Suche nach Regeln ab. Häufig werden informationstheoretische oder Entropie-basierte Gütekriterien nur bei der Auswahl von Regeln angewandt, während die eigentliche Generierung von Hypothesen ein vollständiges Durchschreiten des Suchraumes beinhaltet, bzw. diesen aufgrund anderer ad-hoc Überlegungen (wie z.B. der Konzeptgröße bei CN2) einschränkt. Itrule nutzt bereits beim Aufspannen des Suchraumes eine Heuristik informationstheoretisch begründeter Grenzen.

Ein interessanter Unterschied zu den meisten anderen Regelerzeugern liegt darin, daß Itrule nicht einfach eine Theorie für die in der Beispielmenge gegebene Klassifizierung sucht, sondern verallgemeinerte Konzepte (in gewissem Sinne also mögliche Klassifizierungen der Eingabemenge).

Verfahren:	Induktive Logik Programmierung, erzeugen logischer Formeln aus Beispielen
Gegeben:	Beschreibung von Relationen in Form von Beispielen
Ziel:	Prädikatenlogische Formulierung von Relationen
Suche:	je nach Verfahren
Störungen:	je nach Verfahren
Wissen:	Explizit in den Relationen.
Literatur:	Anwendungsübersicht: Bratko und Muggleton (1995), Bratko *et al.* (1998), Grundlagen von ILP: Muggleton (1993), Data Mining und ILP: Wrobel und Džeroski (1995)

Tabelle 3.12: Eigenschaften von ILP

Resumee

Ich habe Itrule nicht weiter untersucht, da es beim vorliegenden Gegenstandsbereich darauf ankommt, für eine gegebene Klassifizierung (Krank oder Gesund) brauchbare Konzepte zu finden, und das Laufzeitverhalten der getesteten Varianten bei den vorliegenden Problemgrößen eine weitere Einschränkung des Suchraumes vorerst nicht notwendig erscheinen läßt. Dies um so mehr, als aus den mir vorliegenden empirischen Ergebnissen nicht hervorgeht, ob Itrule gegenüber traditionellen Systemen tatsächlich Performance-Vorteile aufweisen kann, da keine vergleichenden Tests gegen andere Verfahren veröffentlicht wurden.

3.4 Logikprogrammierung

In diesem Unterkapitel werden Verfahren skizziert, die auf Logiken erster Ordnung arbeiten (vgl. Unterkapitel 2.5.2). Derartige Systeme verfügen damit über eine ausdrucksmächtigere Repräsentationssprache.

3.4.1 ILP

Unter der Induktiven Logik Programmierung versteht man nicht ein spezielles Verfahren des Maschinellen Lernens, sondern einen Ansatz, eine Klasse von Verfahren. Ich nehme es hier wegen seiner grundsätzlichen Bedeutung auf und habe mit → Foil einen typischen Vertreter genauer beschrieben. Zudem gibt es von dem auf Attributierter Logik basierenden System → JoJo eine auf FOL basierende Variante.

Bei der ILP geht es nach Muggleton (1993) darum, aus Hintergrundwissen und Beispielen Prädikatenlogische Formulierungen von Theorien über einen Gegenstandsbereich zu konstruieren. Dabei sind ILP-Systeme allein aus ihrer logischen Spezifikation heraus unterspezifiziert,

zusätzliche Bestimmungen werden z.B. aus wissenschaftstheoretischen Annahmen gewonnen (wie z.B. Occams Razor[6], also der Bevorzugung einfacher Theorien).

Gegenüber Verfahren, die auf pseudo-propositionalen Logiken aufbauen (wie bis auf die Neuro-Fuzzy-Systeme praktisch alle anderen hier dargestellten Systeme), besitzen ILP-Systeme den Vorteil einer leichten Integrierbarkeit von Hintergrundwissen in Form von z.B. Horn-Klauseln. Außerdem erlaubt die Mächtigkeit der zugrundeliegenden Logik die Formulierung von Relationen zwischen den Objekten. Dahingegen ist es in einer Attributierten Logik nicht möglich, ein Konzept wie beispielsweise „ist-Schwester-von“ aus gegebenen Daten zu konstruieren.

Der Unterschied wird auch schon in der Art der Eingabedaten deutlich: während Systeme, die mit Attributierten Logiken arbeiten, eine Reihe von Beispielen in Attribut-Wert-Darstellung sowie deren jeweilige Klassenzugehörigkeit benutzen, um Theorien über die Klassenzugehörigkeit anhand von Attributwerten zu entwickeln, so arbeiten ILP-Systeme im allgemeinen mit der Eingabe von Relationen (in denen selbstverständlich auch Attribut-Wert-Beziehungen ausgedrückt sein können).

Ein großer Nachteil (vor allem auch im Hinblick auf die im betrachteten Gegenstandsbereich vorkommenden Klassifikationsaufgaben) dürfte sein, daß nach Aussage von EMDE UND WETTSCHERECK (1996) die meisten ILP-Systeme nicht mit Attributen mit kontinuierlichem Wertebereich arbeiten können (eine Ausnahme ist → FOIL).

Ihr Vorschlag geht dahin, ein ILP-System mit Techniken des instanzenbasierten Lernens zu koppeln. Das so entstandene RIBL-System wurde als externes Modul zum Anwendungssystem MOBAL[7] entwickelt. MOBAL ist ein selbstlernendes wissensbasiertes System und wird z.B. zur Unterstützung von Ärzten in der Therapie eingesetzt.

Resumee

Meines Erachtens ist die höhere Ausdrucksmächtigkeit beim derzeitigen Stand für den vorliegenden Gegenstandsbereich nicht notwendig. Man sollte allerdings nicht außer acht lassen, daß eine spätere Erweiterung (wie sie durch den zweigeteilten Aufbau des Lernprozesses prinzipiell möglich ist) die Integration von z.B. epidemiologischen Wissensbasen[8] außerordentlich erleichtern würde, und eine komplett automatisierte Iteration von Generierung und Verwerfung von Hypothesen (mit zwischengeschalteter Simulation aufgrund automatisch entworfener Szenarien) ermöglichen könnte.

Zu beachten ist bei ILP-Systemen auch der Trade-Off zwischen der Vereinfachung der Berechnung durch die verbesserte Möglichkeit, Konzepte überhaupt auszudrücken, und der Vergrößerung des zu durchsuchenden Lösungsraumes[9].

Obwohl noch eher dem Forschungsbereich zuzurechnen, haben ILP-Systeme inzwischen in einigen Gebieten Anwendungsreife erreicht, wie BRATKO UND MUGGLETON (1995) sowie BRATKO *et al.* (1998) zeigen.

[6] Vgl. BLUMER *et al.* (1987).

[7] Vgl. MORIK (1995).

[8] In einer solchen Wissensbasis könnte z.B. Wissen über bekannte kausale Zusammenhänge oder bereits durchgeführte Studien zu finden sein.

[9] Insofern man die Konstruktion von Hypothesen als Suche in einem derartigen Raum begreift, vgl. hierzu die Diskussion in ENZINGER *et al.* (1994).

Verfahren:	→ ILP; hier: Erzeugen von Horn-Klauseln aus Relationen.
Gegeben:	Beschreibung von Relationen in Form von Beispielen 1. Nur positive und closed world assumption, d.h. alle nicht aufgezählen Fälle sind Gegenbeispiele 2. Aufzählen positiver und negativer Fälle.
Ziel:	Prädikatenlogische Formulierung von Relationen in Form von Horn-Klauseln.
Suche:	Heuristische irrevocable Beam Search. Obwohl der (→ AQ-ähnliche) Suchalgorithmus prinzipiell ein backup anbietet, ist er de facto greedy.
Störungen:	Sowohl fehlende Werte als auch Rauschen können behandelt werden. Das Verfahren benutzt Heuristiken (Komplexität der Beschreibung) zum Abbruch; Ziel ist also eher die meisten denn alle positiven Beispiele, und eher kaum eines denn keines der negativen Beispiele zu überdecken.
Wissen:	Explizit in Klauseln.
Literatur:	QUINLAN (1990), QUINLAN UND CAMERON-JONES (1993)

Tabelle 3.13: Eigenschaften von FOIL

3.4.2 FOIL

FOIL ist ein ML-System, welches aus Beispielrelationen Horn-Klauseln generiert. Es handelt sich also (im Gegensatz zu den mit Attributierten Logiken arbeitenden Systemen wie → CN2 oder → ID3) um ein der → Induktiven Logik Programmierung (ILP) zuzuordnendes Verfahren.

FOIL beruht dabei auf Ideen, die (u.a. von Quinlan) im Bereich von ML-Systemen mit Attributierter Logik entwickelt worden sind. Im besonderen ist FOIL in der Lage, jede Theorie auszudrücken, die auch → C4.5 ausdrücken kann. Die Ausdrucksmächtigkeit von FOIL wurde in QUINLAN UND CAMERON-JONES (1993) anhand von zu lernenden Prolog-Programmen untersucht.

Resumee

Auch hier gilt, wie allgemein bei → ILP-Verfahren, daß die erreichbare Ausdrucksmächtigkeit in meinem speziellen Anwendungsfall nicht vonnöten ist. Allerdings würde eine Einbeziehung von Systemen wie FOIL weitere Möglichkeiten zur Interaktion mit anderen Wissensbasen eröffnen, da nicht nur der Raum der möglichen Hypothesen vergrößert wird, sondern auch andere Plausibilitätstests (in Form z.B. eines Logik-Checks) durchführbar wären. Gegenüber den meisten anderen → ILP-Systemen besitzt FOIL dabei den Vorteil, auch Attribute mit kontinuierlichem Wertebereich verarbeiten zu können.

Nefclass, Nefcon, Nefprox: Fuzzy Neuronale Netze	
Verfahren:	Kooperative Neuro-Fuzzy-Systeme.
Gegeben:	Menge von Objekten in Attribut-Wert-Repräsentation.
Ziel:	Als (zumeist dreistufiges) Neuronales Netz realisierte Fuzzy-Systeme. Nefclass ist eine Klassifizierer, Nefcon ein System der Regelungstechnik und Nefprox eine verallgemeinerte Variante.
Störungen:	Sowohl Rauschen als auch fehlende Werte.
Wissen:	Explizit. Eingangsneurone entsprechen Eingabegrößen, Verbindungen zwischen Eingabe- und mittlerer Schicht unscharfen Zugehörigkeitsfunktionen. Die Neuronen der mittleren Schicht entsprechen den Fuzzy Regeln. Verbindungen zur Ausgabeschicht entsprechen ggf. den Erfüllungsgraden, während Ausgabeneurone eine notwendige Defuzzyfizierung von Stellgröße, Klassenzugehörigkeit, etc. realisieren.
Literatur:	Nefclass: Nauck und Klawonn (1995), Nauck und Klawonn (1996), Nauck *et al.* (1996), Hoferichter (1997); Nefcon: Kruse und Nauck (1996); Nefprox: Nauck (1997)

Tabelle 3.14: Eigenschaften von Neuro-Fuzzy-Klassifikatoren

3.5 Sonstige Verfahren

In diesem Unterkapitel werden Verfahren vorgestellt, die auf unterschiedlichen weiteren (impliziten und expliziten) Repräsentationen von Hypothesen arbeiten. Dazu gehören u.a. Systeme, die auf Wahrscheinlichkeitstabellen arbeiten, Neuro-Fuzzy Systeme und graphenbasierte Verfahren (vgl. Kapitel 2.5).

3.5.1 Neuro-Fuzzy Klassifikatoren

Zur Modellierung vager Ausdrücke und nicht-präziser Hypothesen bietet sich die Anwendung von auf Fuzzy-Beschreibungssprachen basierenden Systemen an. Im Bereich der selbstlernenden Fuzzy-Systeme gibt es Kombinationen von Neuronalen Netzwerkmodellen und Fuzzy-Systemen u.a. in der Regelungstechnik, so beschrieben von Kruse und Nauck (1996), und auch zum Clustering und zur Klassifikation, so zu finden bei Nauck *et al.* (1996). Weitere Anwendungsgebiete liegen in der Vorhersage der Aktienentwicklung, so zu finden bei Neuneier und Zimmermann (1996), oder in der medizinischen Diagnostik, so zu finden bei Zahlmann *et al.* (1997).

Fuzzy-Mengen bieten die Möglichkeit, umgangssprachliche Begriffe und unscharfe Zusammenhänge zu modellieren. Im Unterschied zur gewöhnlichen Definition von Mengen, bei der ein Objekt entweder vollständig oder gar nicht zu einer Menge gehört, läßt sich die Zugehörigkeit eines Objektes zu einer Fuzzy-Menge mit einer Zugehörigkeitsfunktion beschreiben.

Sei $\mathcal{M}_k$ eine „klassische“ Menge über einer Menge $\mathcal{O}$ von Objekten, sei ferner $o \in \mathcal{O}$. Dann

gilt für die Zugehörigkeit von o zu $\mathcal{M}_k$ eine Zugehörigkeitsfunktion $Z : \mathcal{P}(\mathcal{O}) \times \mathcal{O} \rightarrow [0,1]$ wie folgt:

$$Z(\mathcal{M}_k, o) = \begin{cases} 1 & \text{falls } o \in \mathcal{M}_k, \\ 0 & \text{sonst.} \end{cases} \tag{3.7}$$

Sei jetzt $\mathcal{M}_f$ eine Fuzzy-Menge über einer Menge $\mathcal{O}$ von Objekten, sei weiterhin $o \in \mathcal{O}$. Dann gilt für die Zugehörigkeit von o zu $\mathcal{M}_f$ eine beliebige Zugehörigkeitsfunktion von der Grundmenge $\mathcal{O}$ in das Einheitsintervall, $Z : \mathcal{P}(\mathcal{O}) \times \mathcal{O} \rightarrow [0,1]$.

Mit dieser Abbildung kann der Grad der Zugehörigkeit eines Objektes zu einer Fuzzy-Menge bezeichnen. Nehmen wir als Beispiel die Menge der leckeren Kuchen, so gibt es sicherlich nicht nur „leckere“ und „fade“ Kuchen, sondern man wird einen speziellen Kuchen „irgendwo“ auf einer Skala von „lecker“ bis „fade“ einordnen wollen.[10]

Handelt es sich bei der Grundmenge $\mathcal{O}$ jetzt noch um eine Menge, auf der sich sinnvoll eine Ordnung definieren läßt (was ich beim Geschmack von Kuchen nicht probieren möchte), so läßt sich jetzt eine exakte Formulierung der Zugehörigkeit von o zu $\mathcal{M}_f$ definieren, die dem Wert von o bzgl. der Ordnung einen Wert aus dem Einheitsintervall zuordnet.

So läßt sich über der Größe von Kuchen die Menge der „mittelgroßen“ Kuchen definieren. Üblicherweise beschränkt man sich bei der Definition von Zugehörigkeitsfunktionen auf einfache Funktionen, wie Dreiecks-, Trapez- oder Gaußfunktion, die sich durch wenige Parameter beschreiben lassen.

Nehmen wir als Beispiel für die Anwendung eines Fuzzy-Systemes die Regelungstechnik. Hier soll abhängig von mehreren Einflußgrößen durch Veränderung einer Stellgröße eine Ausgangsgröße einen bestimmten Sollwert annehmen. So können z.B. der Kurs und die Geschwindigkeit eines Schiffes abhängig von Wind und Wetter und anderen Einflußgrößen auf einem bestimmten Wert gehalten werden.

Die Einflußgrößen können zum einen durch das Aufstellen einer Menge von Differenzen- und Differentialgleichungen modelliert werden, was eine genaue Kenntnis der zugrunde liegenden Prozesse verlangt. Ein Fuzzy-Regler hingegen ist eine Regelbasis, welche die Einflußgrößen angenähert an eine umgangssprachliche Beschreibung modelliert.

Es ist nun wünschenswert, daß die Modellierung der Regler nicht von einem menschlichen Experten zu erfolgen hat, sondern durch ein selbstlernendes System erfolgt. Daher liegt es nahe, die autoadaptiven Fähigkeiten Neuronaler Netze zu koppeln mit der Einfachheit von Fuzzy-Systemen. Zum einen können Neuronale Netze dafür genutzt werden, die Regler zu modifizieren (man spricht dann von kooperativen Systemen), zum anderen lassen sich Fuzzy-Systeme aber auch direkt als Neuronale Netze darstellen (hybride Systeme).

Im Zusammenhang meiner Arbeiten sind allein solche hybriden Ansätze geeignet, bei denen die entstehenden System immer auch als reine Fuzzy-Systeme zu interpretieren sind, um der Black-Box-Problematik konnektionistischer Verfahren zu entgehen. Die von KRUSE UND NAUCK (1996) vorgeschlagenen Modelle wie NEFCLASS erfüllen diese Anforderung.

[10] Ein anderes Beispiel wäre die Frage, ob ein Gebäckstück ein *Teilchen* ist oder nicht (nach einer Anmerkung von Frank Köster).

Das dem NEFCLASS-System zugrunde liegende Modell eines Fuzzy-Perceptron ist von einem dreischichtigem Perceptron[11] abgeleitet. Dabei werden die Gewichte auf den Verbindungen ersetzt durch Fuzzy-Mengen. Ausgabe- und Propagierungsfunktion werden modifiziert. Die Aufgaben der Schichten im einzelnen:

1. **Eingabeschicht:** Wertrepräsentation der Einflußgrößen.
2. **Verbindungen:** Zugehörigkeitsgrade der Eingabeparameter zu den jeweiligen Fuzzy-Reglern. Im Unterschied zu klassischen Perceptrons gibt es gekoppelte Verbindungen, deren Stärke nur gemeinsam verändert werden kann. Somit ist die Interpretation einer Zugehörigkeitsfunktion über alle Neurone der inneren Schicht gleich.
3. **Regelschicht:** Fuzzy-Regler, die über einen Minimumsbildung den Erfüllungsgrad einer Regel aus den aus der Eingabeschicht propagierten Werten bestimmt.
4. **Verbindungen:** In einem Regelsystem wie NEFCON wird die Einflußgröße der einzelnen Regler auf eine einzelne Stellgröße gewichtet. Klassifizierungnetze wie NEFCLASS enthalten für jede Klasse einen Ausgabeknoten, die Verbindungen dienen hier nur der Zusammenfassung der Aktivierungsgrade der passenden Regler.
5. **Ausgabeschicht:** In einem Regelsystem wird die Stellgröße über eine Integration der Aktivierungen der Regler mit anschließender Defuzzyfizierung durchgeführt. Die einzelnen Neurone eines Klassifizierungssystem bestimmen aus den Aktivierungsgraden der Neurone der mittleren Schicht den Zugehörigkeitsgrad des präsentierten Beispiels zur von ihnen repräsentierten Klasse, ausgewählt wird schließlich die Klasse des Neurons, welches am stärksten feuert.

Ein wichtiger Unterschied zu herkömmlichen Neuronalen Netzen ist die Aufgabe der Homogenitätsanforderung (nur gleiche Neuronen und Funktionen) zugunsten der Interpretierbarkeit der Struktur.

Lernalgorithmen für Neuro-Fuzzy-Systeme beruhen zumeist auf Variationen des Backpropagation-Modells. Backpropagation heißt hierbei, daß ein beim überwachten Lernen eines Datensatzes aufgetretener Fehler von der Ausgabeeinheit zurück durch das Netz propagiert wird, um die Verbindungsfunktionen zu modifizieren. Im Gegensatz zu konnektionistischen Systemen wird allerdings eine einfache Heuristik statt eines Gradientenverfahrens eingesetzt, da die im Neuro-Fuzzy-System realisierten Funktionen im allgemeinen nicht differenzierbar sind.

Resumee

Während die anderen hier vorgestellten symbolischen Modelle einen Klassifikator ausgehend von scharfen Attributwerten erstellen und kontinuierliche Werte etwa diskretisiert werden, ist bei Neuro-Fuzzy-Systemen gerade die Angabe unscharfer Zugehörigkeitsgrade der Attribute des zu klassifizierenden Objektes zu verschiedenen Fuzzy-Menge von zentraler Bedeutung.

Die hybride Struktur des Systems aus subsymbolischer Lernkomponente und symbolischer Hypothesensprache lassen Verfahren wie NEFCLASS als interessante Möglichkeit zum Bilden

[11] Zur Einführung in die Theorie Neuronaler Netze vgl. z.B. DORFFNER (1991) und WASSERMAN (1989).

Oodg: Hoodg, Eodg	
Verfahren:	Grapherzeugend, nicht inkrementell.
Gegeben:	Menge von Objekten in Attribut-Wert-Repräsentation, für die Hoodg-Variante müssen die Ausprägungen diskret sein, Eodg kann auch mit kontinuierlichen Wertebereichen arbeiten.
Ziel:	Entscheidungsgraph zur Klassifikation neuer Objekte.
Suche:	Hoodg: Hill Climber, Eodg: Entropie-basierte Heuristik
Störungen:	Sowohl Rauschen wie fehlende Werte werden behandelt.
Wissen:	Explizit im Baum.
Literatur:	Kohavi (1993), Kohavi (1994), Kohavi und Li (1995)

Tabelle 3.15: Eigenschaften von Oodg

von Klassifikatoren erscheinen, sofern die Möglichkeit besteht, die Untersuchungsobjekte in Form unscharfer Mengen zu beschreiben.

3.5.2 OODG

Die Klasse der Oblivious Read Once Decision Graphs (Oodg, Unbewußte Entscheidungsgraphen mit einem Lerndurchlauf) ist eng mit den Entscheidungsbäumen verwandt. Es handelt sich um gerichtete Graphen mit einem ausgezeichneten Startknoten, der Wurzel. Knoten ohne ausgehende Kante sind Blätter. Die Klassifikation beginnt bei der Wurzel, von dort wird in jedem (inneren) Knoten ein Attribut getestet, bis in einem Blatt die Klasse festgestellt ist.

Oodg lassen sich als Bäume verstehen, die in unteren Ebenen wieder verwachsen sind. Wenn von mehr als einem Knoten einer Ebene ein Test auf identische Söhne verweist, so werden diese Söhne identifiziert, und die von den entsprechenden Knoten ausgehenden Kanten verweisen auf genau einen Sohnknoten. Charakteristisches Merkmal der Oodg ist dabei, auf jeder Ebene in jedem Knoten das gleiche Attribut zu testen.

Vorteile von Oodg gegenüber Bäumen ergeben sich vor allem bei Problemen, die in Bäumen zum Entstehen von identischen Teilbäumen („replication problem“) oder sehr vielen Teilbäumen („fragmentation problem“) führen würden. Ein Beispiel für das erstere sind disjunkte Konzepte wie $(A \vee B) \wedge (C \vee D)$, das letztere ensteht z.B., wenn Attribute mit vielen möglichen Wertebelegungen nahe an der Wurzel getestet werden.

Während unter Oodg allgemein die Klasse der Verfahren zur Konstruktion von Read-Once Entscheidungsgraphen zu verstehen ist, so ist der in Kohavi (1993) und Kohavi (1994) beschriebene Hoodg ein konkretes Konstruktionsverfahren mit Hill-Climber-Suchstrategie.

Die Auswahl des Attributtests in einem Knoten erfolgt nicht aufgrund eines lokalen Gütekriteriums, sondern global für jede Ebene. Dabei wird der Attributtest bevorzugt, der in der nächsten Ebene zu einem möglichst schmalen Graphen führt, d.h. möglichst viele Knoten verschmilzt. Das ist durch eine vollständige Suche über die möglichen Tests leicht zu ermitteln.

Verfahren:	(Inkrementelles) Erzeugen von Wahrscheinlichkeiten für Klassifikationen.
Gegeben:	Menge von Beispielen in Attribut-Wert-Repräsentation, Attributwerte nominal.
Ziel:	Tabelle mit bedingten Wahrscheinlichkeiten.
Wissen:	Implizit, Entropie.
Literatur:	LANGLEY *et al.* (1992), KONONENKO (1991), KONONENKO (1993)

Tabelle 3.16: Eigenschaften Bayes'scher Klassifizierer

Drei Probleme fallen bei HOODG besonders ins Auge. Zum einen hat der Algorithmus kein globales Qualitätsmaß. Er ist daher auch nur schlecht dazu in der Lage, irrelevante Attribute zu identifizieren. Zum anderen kann er nicht mit kontinuierlichen Attributen arbeiten, da auch hierfür eine globale Optimierungsstrategie und ein globales Gütekriterium erforderlich wären. Als letztes hat HOODG wie jeder Hill Climber das Problem, daß er in lokalen Gütemaxima hängen bleiben kann.

Diese Probleme versucht der in KOHAVI UND LI (1995) beschriebene Entropiebasierte Algorithmus EODG zu überwinden. Bei ihm wird zuerst ein Entscheidungsbaum mit einem Entropiemaß als Gütekriterium aufgebaut, der dann zu einem Entscheidungsgraphen verklebt wird. Eine Pruning-Phase schließt die Konstruktion des Graphen ab.

Resumee

Im vorliegenden Problembereich kann es durchaus zu Replikationsproblemen kommen, da die Clusterung der Zeitreihen nicht überwacht erfolgt, mehrere unterschiedliche Cluster also gleiche bzw. ähnliche kausale Zusammenhänge repräsentieren können. Daher bietet es sich an, die in $\mathcal{MLC}$++ implementierten OODG-Algorithmen zu testen.

3.5.3 Bayes

Bayes'sche Klassifizierer wurden Anfang der 70er Jahre im Bereich der Mustererkennung entwickelt. Bei dieser Methode des überwachten Lernens wird für jede zu erkennende Klasse eine Zusammenfassung von Wahrscheinlichkeiten gespeichert, gebildet aus den bedingten Wahrscheinlichkeiten für ein Attribut bei gegebener Klasse sowie den Grundwahrscheinlichkeiten der gegebenen Klasse.

Die zugrunde liegende Sprache hat in etwa die gleiche Ausdrucksmächtigkeit wie ein Perceptron. Für jede zu lernende Instanz wird die Tabelle der gespeicherten Wahrscheinlichkeiten für die entsprechende Klasse aufdatiert. Eine Instanz unbekannter Klasse wird klassifiziert, indem die möglichen Klassenbezeichner anhand einer Evaluierungsfunktion gewichtet werden, dabei wird die Klasse gewählt, deren bedingte Wahrscheinlichkeiten den höchsten Wert ergeben.

Sowohl diese Evaluierungsfunktion als auch die zugrunde liegende Tabellenstruktur gehen von voneinander unabhängigen Attributen aus. Dies ist für reale Probleme eine zumeist nicht

haltbare Annahme. Trotzdem zeigen Bayes'sche Klassifizierer auch in Bereichen, in denen diese theoretisch begründete Annahme verletzt wird, eine überraschend gute Performance (vgl. z.B. BRATKO *et al.* (1996)).

In KONONENKO (1993) findet sich eine vergleichende Studie über den Einsatz eines Bayes Classifiers und des ID3-basierten Systems ASSISTANT (→ ID3) in vier medizinischen Anwendungen. Die Performance des naiven Bayes-Ansatzes erweist sich dabei als überlegen. Es werden Lösungsansätze für zwei Problembereiche des Naiven Bayes angegeben:

- Behandlung kontinuierlicher Attributwerte statt Beschränkung auf nominale Attribute durch Einführung sogenannter Fuzzy Bounds und
- Modellierung interdependenter Attribute (der naive Bayes-Ansatz verlangt, daß die einzelnen Attribute voneinander unabhängig sind).

Interessant ist ebenfalls der Hinweis auf die Möglichkeit, den Bayes Classifier als Neuronales Netz darzustellen, welches erklärbar sei. Da mir allerdings die Hypothesensprache eines Bayes'schen System für das behandelte Problemgebiet nicht adäquat erscheint, habe ich diese Möglichkeit nicht weiter untersucht.

KONONENKO (1991) stellt weiterhin eine Erweiterung des sogenannten naiven Bayes vor, bei der die Interdependenz von Attributen explizit errechnet wird. Die so errechnete Korrelation wird benutzt, um die bedingten Wahrscheinlichkeiten der Tabelle zu gewichten. Die Leistungen des semi-naiven Ansatzes sollen in Bereichen, in denen die Attribute tatsächlich unabhängig sind, der Leistung des naiven Bayes entsprechen, in Bereichen mit interdependenten Attributen diese übertreffen (Kononenko führt Untersuchungen in vier medizinischen Bereichen durch).

Eine ganze Reihe von weiteren Algorithmen aus dem Bereich des Maschinellen Lernens basiert auf Ideen Bayes'scher Klassifizierer, so z.B. das in CHEESEMAN UND STUTZ (1996) beschriebene nicht-inkrementelle Verfahren AUTOCLASS zum automatischen Clustering gegebener Instanzen.

Ein naiver Bayes-Ansatz, wie er auch in $\mathcal{MLC}$++ implementiert ist, wurde von LANGLEY *et al.* (1992) analysiert. Unter der Annahme voneinander unabhängiger, gleichverteilter Attribute, eines monotonen konjunktiven Zielkonzeptes und der Abwesenheit von Rauschen bei den Attributwerten gelingt es mit dem vorgestellten Modell, die Leistungsfähigkeit eines naiven Bayes vorherzusagen.

Resumee

Mit einem Bayes'schen Klassifizierer kann keine explizite Hypothese über kausale Zusammenhänge aufgestellt werden, es werden lediglich Aussagen über Wahrscheinlichkeitsverteilungen generiert. Das schränkt seine Anwendbarkeit im vorliegenden Gegenstandsbereich stark ein.

Auch sind umfangreiche Tabellen bedingter Wahrscheinlichkeiten für einzelne Klassifikationsaufgaben zwar handhabbar, ihre Bedeutung ist aber kaum kommunizierbar. Ich habe daher auf eine weitere Analyse verzichtet.

Verfahren:	Instance Based Learning: Nearest-Neighbour Varianten.
Gegeben:	Menge von Beispielen in Attribut-Wert-Repräsentation.
Ziel:	Sammlung von Instanzen. Dazu gibt es eine Ähnlichkeits- und eine Zugehörigkeitsfunktion für die Klassifizierung neuer Instanzen. IB3-CI ist in der Lage, induktiv neue Features zu konstruieren.
Suche:	Greedy.
Störungen:	Sowohl fehlende Werte als auch Rauschen können behandelt werden. Allerdings hat IB2 Schwierigkeiten mit Rauschen.
Wissen:	Implizit in Instanzen und Ähnlichkeits- und Zugehörigkeitsfunktionen.
Literatur:	IB1, IB2, IB3: AHA *et al.* (1991); IB3-CI: AHA (1991).

Tabelle 3.17: Eigenschaften der IB-Familie

3.5.4 IB-Familie

Die Algorithmen IB1, IB2 und IB3 werden in AHA *et al.* (1991) näher beschrieben. Bei ihnen wird nicht durch die Menge der Lerninstanzen ein Klassifikator erzeugt, sondern die Klassifikation bisher nicht gesehener Instanzen erfolgt durch die in der Lernphase gesehenen Instanzen, eine Ähnlichkeitsfunktion zwischen Instanzen und eine Zugehörigkeitsfunktion. Es handelt sich also um eine Nearest Neighbour-Variante, wobei allerdings keine Abstraktion von den Instanzen geleistet wird. Die Algorithmen im einzelnen:

IB1: Diese Grundform ist praktisch ein Nearest Neighbour Algorithmus. Die Spannweite der möglichen Attributwerte wird allerdings normalisiert, die Instanzen werden inkrementell verarbeitet und es gibt eine einfache Handhabung fehlender Attributwerte.

IB2: Diese Variante merkt sich nur Instanzen, die nicht richtig klassifiziert wurden. Dadurch wird der Speicherbedarf minimiert, allerdings sinkt in verrauschten Situationen die Klassifikationsgenauigkeit.

IB3: Um die Schwäche von IB2 bei verrauschten Daten auszugleichen, wird bei ID3 für jede gespeichert Instanz die „Güte“ gespeichert, d.h. der Algorithmus merkt sich, wie viele ungesehene Exemplare von einer gegebenen Instanz richtig und wie viele falsch klassifiziert wurden. Daraus wird die Brauchbarkeit in der Zukunft extrapoliert. Der Algorithmus bestimmt daraus die Instanzen, die gut zur Klassifikation geeignet sind, und solche, die Ergebnis von Rauschen sind. Letztere werden im Laufe der Zeit ersetzt. IB3 überwindet viele Probleme von IB2 und verbindet eine gute Klassifikationsgüte mit geringem Speicherplatzbedarf.

Die IB3-Variante IB3-CI (beschrieben in AHA (1991)) ist im Gegensatz zu den ursprünglichen Verfahren in der Lage, Features (das sind in etwa aggregierte neue Attribute) zu konstruieren und damit die Diskrepanz zwischen der Sprache, in der die zu lernenden Objekte vorliegen, und einer adäquaten Problembeschreibungssprache zu verringern. IB3-CI ist in der Lage,

Verfahren:	Graphenorientierte Analyse von Kausalbeziehungen in Zeitreihen.
Gegeben:	Menge von Beispielen in Attribut-Werte-Relation, wobei die einzelnen Attribute mehrfach für unterschiedliche Zeitpunkte vorhanden sind.
Ziel:	Kausalmodell in Form eines gerichteten beschrifteten Graphen mit Eigenschaften als Knoten und kausalen Abhängigkeiten als Kanten.
Suche:	Vollständig, zweistufig: 1. Discovery – Generieren von Hypothesen und Überprüfung ihrer Wahrscheinlichkeit durch Testen von Kovarianz und zeitlicher Abhängigkeit. 2. Study – Design statistischer Modelle für Beziehungen mit der höchsten Wahrscheinlichkeit unter Ausnutzung von Domain Knowledge.
Wissen:	Explizit im Graphen.
Bemerk.:	Anwendung auf eine medizinische Datenbank.
Literatur:	BLUM (1982), HOLSHEIMER UND SIEBES (1994)

Tabelle 3.18: Eigenschaften von RADIX, RX

durch konstruktive Induktion eine anwendungsspezifische Generalisierung der gelernten Instanzen vorzunehmen.

Resumee

Aufgrund der Tatsache, daß keine explizite Beschreibung der Klassifikation erzeugt wird, eignen sich instanzenbasierte Verfahren nicht für das von mir bearbeitete Problemfeld, da nicht die Klassifikation das Ziel ist, sondern eine maschinell generierte Darstellung der Kausalbeziehungen zwischen Attributwerten und Klassenzugehörigkeit.

3.5.5 RADIX, RX

Das Anfang der 80er Jahre von BLUM (1982) entwickelte RX-System ist in der Lage, Datenbanken mit Einträgen mit Zeitbezug zu durchsuchen und einen gerichteten azyklischen Graphen mit möglichen kausalen Abhängigkeiten aufzubauen.

Zeitbezug bedeutet hierbei, daß in der Datenbank zu einem Patienten das gleiche Attribut (wie z.B. die Körpertemperatur) mehrfach für unterschiedliche Zeitpunkte eingetragen sein kann. Die Einträge in der Datenbank lassen sich also in einem dreidimensionalen Raum mit den Achsen Patient, Attribut und Zeit anordnen.

Die Knoten des Graphen repräsentieren Eigenschaften. Diese Eigenschaften können entweder direkt in der Datenbank abgelegte Attribute, wie die Körpertemperatur zu einem bestimmten

Zeitpunkt, sein. Sie können aber auch, evtl. unter Benutzung von Domain Knowledge, aus anderen Attributen ableitbar sein, wie etwa eine Diagnose.

Die Kanten stellen kausale Beziehungen dar, wobei die Aussage „A steht in kausaler Abhängigkeit zu B“ im Sinne von Blum bedeutet:

1. **Korrelation:** A ist mit B korreliert, genau dann wenn für ein Beispiel $\mathcal{P}$ aus der Datenbank gilt: falls $\mathcal{P}$ die Eigenschaft A hat, dann hat $\mathcal{P}$ die Eigenschaft B. Die kausalen Beziehungen sind darüberhinaus zeitlich geordnet:
2. **Zeitliche Vorgängigkeit:** A ist oBdA gefolgt von B, d.h., A ist die *Ursache* und B die *Wirkung*. Außerdem gilt:
3. **Nicht-Zufälligkeit:** A und B sind korreliert, falls es keine (bekannte) dritte Eigenschaft C (Confounder) gibt mit: C steht in kausaler Beziehung mit A und C steht in kausaler Beziehung mit B.

Jede der Kanten ist mit zusätzlichen Informationen zu den Relationen annotiert:

1. **Intensität:** Die erwartete Veränderung der Wirkung bei einer gegebenen Veränderung der Ursache (statistisch). Sie ist nicht als einzelne Größe, sondern als diskretisierte Verteilung angegeben.
2. **Häufigkeit:** Die Häufigkeitsverteilung bei den untersuchten Fällen. Auch hier wird die gesamte Verteilung angegeben und nicht ein aggregierter Wert. Die Verteilung ist diskretisiert.
3. **Richtung:** Positive oder negative Korrelation. Dieses Merkmal ist redundant.
4. **Validität:** Die Sicherheit, mit der eine Beziehung ausgesagt wird, ist in 10 Stufen angegeben, wovon das System selber nur Sicherheiten von 1 bis 5 vergeben kann, höhere Stufen müssen beispielsweise aus der Literatur verifiziert sein.

Bei dem Graphen handelt es sich somit letztlich um eine modifizierte, annotierte Variante eines Bayes'schen Believe Network.

Ziel des gesamten Systems ist es, neue kausale Beziehungen in einem Modell aufzufinden, wenn es mit

1. einem Lerndatensatz in Form einer medizinischen Datenbank und
2. einer Wissensbasis in Form eines Graphen

gestartet wird. Bei RADIX/RX handelt sich um ein zweistufiges Modell. Im ersten Teil, dem Discovery-Modul, werden durch nichtparametrisierte, pseudo-zeitinvariante statistische Tests Hypothesen über kausale Beziehungen aufgestellt und ihre qualitativen und quantitativen Eigenschaften errechnet. Zur Verringerung der numerischen Komplexität (bei n Eigenschaften können $n(n+1)$ Kombinationen berechnet werden) können einige Eigenschaften von vorneherein als Ursachen, andere als Wirkungen gekennzeichnet werden. Zeitinvarianz wird über

Verfahren:	Nicht-inkrementelles Conceptual Clustering.
Gegeben:	Eine Menge von Beobachtungen in offener Prädikatenlogik.
Ziel:	Eine Hierarchie von Begriffsdefinitionen, die neue Beobachtungen vorhersagen.
Literatur:	(MORIK, 1995, S. 261), MICHALSKI (1983)

Tabelle 3.19: Eigenschaften von STAR

unterschiedliche Zeitverschiebungen modelliert. Diese Verschiebung ist nicht heuristisch gesteuert.

Im zweiten Teil, der Study-Phase, wird die Validität der wahrscheinlichsten Hypothesen des Discovery-Moduls getestet. Dabei wird die Domain Knowledge benutzt, um Confounder zu identifizieren. Der Einfluß von Confoundern wird durch statistische Verfahren und Ausschnittmodellierung minimiert. Die gefundenen möglichen kausalen Beziehungen werden auf statistische Signifikanz untersucht. Je nach Ergebnis wird ihre Validität erhöht oder die Kante entfernt.

RX hat erfolgreich medizinische Hypothesen aus einer rheumatologischen Datenbank generiert, die Performance leidet aber doch unter der Tatsache, daß die Suche nicht durch eine Wissensbasis angeleitet ist. Der Nachfolger RADIX verfügt daher über ein wissensgetriebenes Suchmodul.

Resumee

Obwohl RX in der Lage ist, mit zeitrelatierten Daten zu arbeiten, so ist die Arbeitsweise doch kaum direkt auf die vorliegende Problemstellung übertragbar. Der Schwerpunkt der zu untersuchenden Abhängigkeiten liegt nicht auf dem Verhältnis der Zeitreihen untereinander, sondern auf Mustern, die eine vorgegebene Klassifizierung des untersuchten Individuums zur Wirkung haben. Ich habe daher auf eine weitergehende Untersuchung verzichtet.

3.5.6 STAR

STAR ist ein frühes Clustering-Verfahren, d.h. daß nicht eine Klassifizierung anhand eines vorgegebenen ausgezeichneten Attributes „Klassenzugehörigkeit“ von gegebenen Instanzen erzeugt wird, sondern daß die Aggregation, also das Finden von Klassen, zum Induktionsschritt dazugehört[12].

Ich habe die STAR-Methode zum einen aufgenommen, weil bei dem von mir vorgeschlagenen System zur Erzeugung von Hypothesen aus Zeitreihen sowohl das Conceptual Clustering als auch die überwachte Generierung von Klassifikationen eine Rolle spielt (die Zeitreihen werden ungeachtet der Klassenzugehörigkeit der Individuen geclustert). Auch wenn ich mich aufgrund der Problemstellung für ein anderes Verfahren aus dem Bereich Mustererkennung entschieden

[12] Vgl. z.B. die Klassifikation von Algorithmen des Maschinellen Lernens in CARBONELL *et al.* (1983)

habe, so streift MICHALSKI (1983) in seinem einführenden Text viele der zugrunde liegenden Probleme.

Zum anderen halte ich es aber für vorteilhaft, längerfristig auch auf der zweiten Ebene, also der Generierung von Hypothesen, die Nutzung von STAR-ähnlichen Methodiken zu untersuchen. Dadurch wäre es z.B. möglich, ohne Kenntnis von Krankheitszuständen epidemiologische Zusammenhänge zu finden. Dies könnte insbesonders bei Krankheiten mit langer Inkubationszeit, bei denen der tatsächliche Krankheitszustand des untersuchten Individuums unentdeckt geblieben sein kann, zu besseren Ergebnissen führen als das derzeit von mir verwendete System der überwachten Akquirierung von Hypothesen.

Wichtig ist bei allen Konzept-Cluster-Verfahren, eine geeignete Kombination wahrheitserhaltender Deduktion aus bereitgestelltem Hintergrundwissen sowie bereits induziertem Wissen und (wiederum hintergrundwissengesteuerter, aber trotzdem unsicherer) Induktion nützlicher neuer Fakten zu finden.

3.6 Bewertung

Auffällig bei den vorgestellten Systemen ist, daß Attribute mit kontinuierlichem Wertebereich häufig problematisch sind. Dies trifft für die meisten ILP-Systeme in besonderen Maße zu. Aber auch Verfahren, die mit einer Attributierten Logik als Hypothesensprache arbeiten, diskretisieren kontinuierliche Werte zumeist vorab. Eine Ausnahme stellt z.B. C4.5 dar.

Von allen vorgestellten Modellen ist einzig ADSEQ in der Lage, mit in sich stark strukturierten Attributen zu arbeiten, wie es die in meinem Fall vorliegenden Zeitreihen darstellen.

Andere Systeme wie RADIX/RX oder das in SEITZ UND UHRMACHER (1998) beschriebene OASES-System modellieren zwar zeitliche Faktoren in ihren Modellen. Zeit ist dabei allerdings i.a. eine Ordnungsdimension, auf der andere beobachtbare Erscheinungen angeordnet sind. Die Klassifikation erfolgt dann auf Grundlage zeitlich geordneter Ereignisse.

Eine solche Modellierung von Zeit kommt für die benötigte Analysekomponente epidemiologischer Studien nicht in Frage. Zwar wird vermutet, daß in den Zeitreihen kausale Abhängigkeiten versteckt sind, die letztlich den Ausbruch einer Krankheit verstärken oder hemmen.

Allerdings wird dabei nicht nur von diskreten, zeitlich geordneten kausalen Ereignissen ausgegangen. Dies kann sicherlich auch der Fall sein, wenn z.B. eine kurzzeitige hohe Belastung mit einem Schadstoff eine Krankheit direkt auslöst. Viel wichtiger ist allerdings die Entdeckung von Kausalketten, die z.B. über den langfristigen Schadstoffeintrag in Zusammenhang mit anderen Faktoren die Erkrankung begünstigen.

Häufig wird bei Klassifikationen mit zeitlicher Dimension auch auf einzelnen Zeitreihen gearbeitet, so z.B. bei den Untersuchungen von KUBAT *et al.* (1993) zur Performance verschiedener Verfahren (u.a. mehrschichtige Perceptrons, ein TDIDT-Ansatz und LVQ) bei der Klassifikation von EEG-Signalen.

Allein das ADSEQ-Verfahren von Rummel kommt letztlich für das von mir betrachtete Anwendungsgebiet in Frage. Die Kombination unterschiedlicher Paradigmen des Maschinellen Lernens scheint für die Bearbeitung von Zeitreihen einige Vorteile zu bieten. Gegenüber dem von Rummel beschriebenen System weiche ich allerdings in zwei entscheidenden Punkten ab:

1. **Lokales vs. globales Processing strukturierter Attribute:** Für die Diskretisierung von Attributen mit kontinuierlichem Wertebereich haben DOUGHERTY *et al.* (1995) bereits gezeigt, daß der Ansatz von C4.5 mit lokaler Diskretisierung nicht unbedingt zu verbesserter Performance führt. Ich habe daher das Clustering in eine getrennte erste Phase (ACTS – Adaptive Clustering of Time Series) verlegt. Dies bietet aus meiner Sicht folgende wesentliche Vorteile:

   (a) Ein typisches Adaptives Clusteringverfahren wie k-means Clustering (vgl. Unterkapitel 4.3) zeigt ein quadratisches Laufzeitverhalten. Da bei meinen Rohdaten strukturierte Attribute nicht die Ausnahme, sondern die Regel sind, ein Clustering aber beim Aufbau des Baumes für jedes noch nicht im entsprechenden Pfad getestete Attribut neu durchgeführt werden müßte, wäre der Laufzeitaufwand für eine große Anzahl von Attributen enorm. Dazu kommt, daß ein typischer Baumalgorithmus selber im worst case-Szenario ein quadratisches Laufzeitverhalten hat.

   (b) Die Verständlichkeit und Kommunizierbarkeit leidet darunter, daß ein und dasselbe Attribut je nachdem, in welchem Knoten ein Test stattfindet, in unterschiedliche Klassen zerfallen kann. Bei der Bewertung der Plausibilität z.B. eines aufgebauten Entscheidungsbaumes muß also in jedem Knoten der Zerfall eines Attributes in die möglichen Klassen bedacht werden. Bei regelerzeugenden Systemen wäre das Verständlichkeitsproblem eher noch größer, da es nicht unbedingt eine Ordnung auf den Tests gibt (wie dies beim Baum durch den Pfad der Fall ist).

2. **Auswahl des symbolischen Verfahrens:** Die Einbindung in den Lernschritt eines Algorithmus ist zwar sehr elegant. Allerdings ist es (zumindest beim derzeitigen Stand der Forschung) praktisch nicht möglich, vorherzusagen, welcher Algorithmus einer Klasse von Verfahren in einem gegebenen Problembereich die beste Performance zeigt. Da es sich bei meiner Arbeit um einen prototypischen Test auf Machbarkeit handelt, die Einbindung eines adaptiven Clusterings in mehr als einen Algorithmus den Rahmen einer Diplomarbeit allerdings sprengen dürfte, halte ich es für sinnvoller, subsymbolischen und symbolischen Schritt zu trennen. Dies bietet wiederum zwei Vorteile:

   (a) Es wird dadurch möglich, mit relativ geringem Aufwand eine ganze Reihe von Algorithmen testen zu können.

   (b) Genauso wäre es denkbar, statt des von mir prototypisch implementierten subsymbolischen Verfahren auch andere (subsymbolische) Algorithmen zum Vorab-Clustering der Zeitreihen einzusetzen.

Ich habe daher darauf verzichtet, einen einzigen symbolischen Ansatz auszuwählen (wie Rummel es mit C4.5 getan hat), um in diesem die Verarbeitung von strukturierten Attributen exemplarisch zu integrieren. Stattdessen habe ich mich darauf konzentriert, die Anbindung an eine ganze Reihe von symbolischen Algorithmen sicherzustellen, mit denen ausführliche empirische Test auch auf Realdatensätzen ausführbar sind.

In einer ersten Auswahl habe ich mich auf folgende symbolische Verfahren konzentriert:

Bei den baumbasierten Verfahren ist die in $\mathcal{MLC}$++ implementierte, um das Handling kontinuierlicher Wertebereiche ergänzte ID3-Variante als Referenzsystem zu betrachten. Mit dem

C4.5-Derivat MC4 steht gleichzeitig einer der modernsten anwendungsnahen baumbasierten Klassifizierer zur Verfügung.

Von den eher forschungsorientierten Ansätzen halte ich vor allem OPTIONDT und HOODG für geeignet.

Der von mir ausgewählte und später noch zu erläuternde subsymbolische Ansatz für das Clustering der Zeitreihen läßt grundsätzlich eine Kodierung zu, die als unscharfe Zugehörigkeit zu den sich herausbildenden Klassen gedeutet werden kann, daher beziehe ich auch NEFCLASS in meine Untersuchungen ein.

Von den regelbasierten Ansätzen erscheinen mir CN2 und zu Vergleichszwecken RIPPER die am ehesten geeigneten Verfahren zu sein. Es bleibt darauf hinzuweisen, daß derzeit i.a. die baumbasierten Verfahren bei der Anwendung auf reale Probleme „die Nase vorn haben“.

Zum Schluß dieses Kapitels bleibt folgende Frage offen: wie komme ich von den Zeitreihen, die mir vorliegen, überhaupt erst einmal zu Attributen, wie sie von den symbolischen Verfahren ausgewertet werden können?

Kapitel 4

Überblick: Clustering

*The second distinctive feature about human learning is that there's no copy process (*SIMON*, 1983, S. 27).*

4.1 Einleitung

Es scheint so, als ob sich das Problem nur verlagert hätte, die Lösung allerdings noch genauso fern läge wie am Anfang der Arbeit. Es ist daher sinnvoll, sich einmal vor Augen zu führen, wie sich die Problemstellung verändert hat.

Das eigentlich zu lösende Problem besteht darin, aus einer Menge von Objekten, die in Attribut-Wert-Repräsentation vorliegen und (mindestens) zwei verschiedenen Klassen angehören, die Attribute herauszufinden, die die jeweilige Klassenzugehörigkeit begründen. Die Attribute können symbolische Kategorien sein oder einem kontinuierlichem Wertebereich angehören, liegen zumeist aber in Form von Zeitreihen vor, also in sich wiederum strukturierten Attributen.

Es hat sich nun gezeigt, daß die klassischen Methoden des symbolischen Maschinellen Lernens, die aufgrund ihrer interpretierbaren Hypothesensprache für unser Vorhaben besonders geeignet schienen, auf derartigen Wertebereichen von Attributen nicht arbeiten können. Möchten wir ihre Vorteile trotzdem nutzen, so müssen die Zeitreihen vorab (bzw. im Prozeß des Lernens) in eine besser handhabbare Form gebracht werden. Wir brauchen also eine symbolische Beschreibung der Zeitreihen (symbolisch im Sinne der Problembeschreibungssprache der Algorithmen, die wir anzuwenden wünschen).

Für ein solches Vorhaben stellt nun die klassische Mustererkennung eine ganze Reihe von Verfahren zur Verfügung, die getreu der von mir dargelegten Taxonomie von Techniken des Maschinellen Lernens zumeist den subsymbolischen, nicht überwachten Clustering-Methoden zuzurechnen sind.

Ich verzichte auf die Nutzung der mir vorliegenden Information über die Klassenzugehörigkeit aus zwei Gründen. Zum einen möchte ich einen Zirkelschluß vermeiden. Nachfolgende symbolische Algorithmen könnten Hypothesen aufgrund von Artefakten entwickeln, die durch die

überwachte Klassifikation begründet sind. Solche Hypothesen könnten trivialerweise gefunden werden.

Zum anderen ist ja noch nicht bekannt, welche Zeitreihe wirklich in einer Kausalbeziehung zur Krankheit steht. Eine Clusterung von Zeitreihen aufgrund der Klasenzugehörigkeit der Individuen würde für Attribute, die in keinem kausalen Zusammenhang zur Klassenzugehörigkeit stehen, eine „künstliche“ Partitionierung befördern.

Ich sehe allerdings keinen Grund, der dagegen spricht, das Wissen über die Klassenzugehörigkeit den hypothesengenerierenden Verfahren zugänglich zu machen. Das könnte z.B. durch die Entwicklung einer Ordnung auf den gefundenen Partitionen geschehen (vgl. Unterkapitel 8.2.2). Dies muß m.E. jedoch nach der eigentlichen Clusterung erfolgen.

4.2 Konnektionistische Systeme

GUO UND GELFAND (1992) stellen ein baumbasiertes Klassifikationsverfahren vor, bei dem kleine mehrschichtige Feedforward-Netze in den Knoten die Extraktion nicht-linearer Features, also (verkürzt gesagt) Aggregationen interessanter Attribute, übernehmen.

Dies soll den bekannten Nachteil baumbasierter Verfahren entgegenwirken, bei Betrachtung einzelner Attribute den Lösungsraum immer nur entlang von Hyperflächen senkrecht zu den durch die Attribute gebildeten Achsen aufzuteilen, und gleichzeitig die Verständlichkeit von Entscheidungsbäumen erhalten (vgl. das vorgestellte Modell der OPTIONDT im Unterkapitel 3.2.3).

Die Autoren sehen einen Vorteil ihres Ansatzes in der Tatsache, daß die entstehenden Bäume gegenüber klassischen baumbasierten Verfahren kleiner seien und niedrigere Fehlerraten hätten, und führen hierzu Tests gegen CART durch. Ein Nachteil sei die längere Laufzeit.

Wenn ich mich auch aufgrund bereits geäußerter Gründe gegen eine hybride Lösung entschieden habe, so lenkt die Arbeit von Guo und Gelfand den Blick doch auf die Möglichkeit, Neuronale Netze in einer ersten Stufe einzusetzen. Im Gegensatz zu ihrem Ansatz, in jedem Knoten die Testmenge in zwei Klassen aufzuspalten, die im Hinblick auf die vom Baum zu leistende Klassifizierung den höchsten Informationsgewinn erreichen,[1], geht es mir jedoch um das Auffinden von Clustern in der Testmenge. Ich untersuche daher andere Netztopologien.

Allgemeines

Der Lernvorgang eines Neuronalen Netzes soll ein spezielles Ergebnis erzielen. Ein Netz zur Voraussage von Aktienkursen soll eine Funktion für Aktienkurse entwickeln[2], ein assoziativer Speicher soll Konzepte lernen. Der Lernvorgang soll in jedem Fall nicht völlig zufällig sein, sondern sich in Richtung einer vorab definierten Performance-Verbesserung bewegen. Geht

[1] Das zu optimierende Maß ist für Guo und Gelfand das Gini-Kriterium, vgl. z.B. (BREIMAN *et al.*, 1984, S. 121).

[2] Interessant ist dabei die Aussage von R. Kruse auf der KI-98, daß zumindest die ihm bekannten Systeme derzeit damit durchaus ihre Schwierigkeiten hätten. Das deutet fälschlicherweise darauf hin, eine derartige, rational zu fassende Funktion gäbe es gar nicht.

man von einem Hebb'schen Lernalgorithmus aus, d.h. daß der Lernerfolg durch Verstärkung von Verbindungsstrukturen erreicht wird, so lassen sich nach DORFFNER (1991) zwei Formen eines solcherart geleiteten Lernens unterscheiden:

1. direktes Feedback durch einen Lehrer bei supervised learning, oder
2. Festlegung der Art, wie sich die Aktivierung herausbilden kann.

Der zweite Fall ist wichtig, wenn nur die Art des Modells, nicht aber die Ausprägung feststeht. Genau dies ist m.E. im vorliegenden Anwendungsbereich der Fall.

Competitive Learning

Eine Möglichkeit, eine Lernstrategie festzulegen, ist der „the winner takes it all"-Ansatz bei der Musterkategorisierung mit Hilfe des *Competetive Learning*. Die Grundarchitektur eines Netzes für dieses Verfahren vereint assoziative und kompetitive Vernetzungsstrukturen.

Es handelt sich um mehrschichtige Assoziationsnetzwerke mit positiven (also exhibitorischen) Feedforward-Verbindungen. Die erste Schicht dient der Eingabe der Muster. Die Neurone der anderen Schichten sind in Cluster aufgeteilt, die Neurone eines Clusters sind lateral (untereinander) mit negativen (inhibitorischen) Synapsen verbunden.

Zur Vermeidung von Mißverständnissen spreche ich im folgenden von Clustern von Neuronen und zu lernenden Kategorien in den Eingabedaten. Jedes der Neurone eines Clusters steht für eine mögliche Kategorie von Mustern in der Eingabe, auf die es mit starker Aktivierung reagiert. Es gibt dabei keine a priori-Zuordnung von Neuronen zu solchen Mustern. Die Anzahl der maximal zu unterscheidenden Kategorien ist also durch die Anzahl der Neurone festgelegt, allerdings kann auch eine geringere Anzahl an Konzepten gelernt werden.

Existieren allerdings Kategorien in den Eingabedaten, so werden sich die Neurone eines Clusters nach dem *sparse pattern learning theorem* (vgl. z.B. DORFFNER (1991)) mit hoher Wahrscheinlichkeit auf jeweils eine solche Kategorie spezialisieren. Die lateralen inhibitorischen Verbindungen dienen dabei dazu, andere Neurone davon „abzuhalten", die gleichen Kategorien zu lernen.

Da die letztliche Kategorisierung stark von den (im Hinblick auf die Klassifikation) zufälligen Anfangsgewichten abhängig ist, existieren zumeist so viele Cluster, daß Kategorien entlang verschiedener Dimensionen entdeckt werden können (häufig sind ja auch mehrere Kategorisierungen denkbar).

Der Lernprozeß im Einzelnen:

1. Ein binäres Eingabemuster wird angelegt. Die entstehenden Aktivierungen werden durch das Netz propagiert. Diese Propagierung wird iteriert, durch die inhibitorischen Verbindungen der Neurone kommt es zu Konzentrationen der Aktivierungen. Dieses wird fortgesetzt, bis ein Neuron maximale und praktisch alle anderen Neurone minimale Aktivierung erreicht haben.

2. Innerhalb eines Cluster lernt nur der so gefundene Gewinner, d.h. nur die zu ihm führenden Verbindungen werden verstärkt. Setzt man die Δ-Regel als Lernregel voraus, so läßt sich diese wie folgt modifizieren:

$$\Delta w_{ij} = \begin{cases} 0 & \text{falls Neuron } i \text{ verliert,} \\ \mu(\frac{c_{ik}}{n_k} - w_{ij}) & \text{sonst.} \end{cases} \tag{4.1}$$

   hierbei ist n_k die Anzahl der aktiven Neurone im Muster k der darunter liegenden Schicht ist. c_{ik} ist genau dann 1, wenn das in der vorhergehenden Schicht direkt verbundene Neuron i aktiv ist, sonst 0. μ ist der Lernparameter. Im Klartext bedeutet dies, daß Verbindungen zwischen aktiven Neuronen benachbarter Schichten verstärkt und andere abgeschwächt werden. Somit steigt die Wahrscheinlichkeit, daß das gleiche Muster wieder das gleiche Neuron gewinnen läßt, während diese Wahrscheinlichkeit für andere Muster sinkt.

3. Der Algorithmus wird über alle möglichen Eingabedaten iteriert.

Resumee

Das vorgestellte Verfahren eignet sich sehr gut zur Entdeckung von Mustern in Eingabedaten. Vier Eigenschaften halte ich allerdings in Hinblick auf den untersuchten Gegenstandsbereich für problematisch:

1. Das einzige Kriterium für die Ähnlichkeit zweier Muster ist die Überlagerung der Aktivitäten zweier Neurone in unterschiedlichen Clustern. Auf einer solchen Kategorisierung läßt sich allerdings sinnvoll keine unscharfe Zugehörigkeitsfunktion definieren, was aber beispielsweise die Anwendung eines Neuro-Fuzzy Systems ermöglichen würde, um die gefundenen Cluster in Beziehung zu einer beobachteten Gesundheitssituation zu setzen.

2. Es stellt sich das Problem der Kommunizierbarkeit der entdeckten Muster. Überspitzt formuliert: was nützt es einem Epidemiologen, wenn er von einem System erfährt, daß ein bestimmtes Attribut in kausaler Beziehung zum Ausbruch von Krankheiten stehen könnte, er aber nicht erkennen kann, in welcher?

3. Kategorien können weiterhin unentdeckt bleiben, wenn Muster sich zwar in Gruppen teilen, diese aber „gleichverteilt“ sind und sich die Muster nicht ausreichend phänomenologisch unterscheiden.

4. Wie allgemein bei Neuronalen Netzen gilt auch hier, daß Hintergrundwissen kaum integrierbar ist. Dies ist bei später möglichen Erweiterungen von Nachteil, falls die Interaktion mit epidemiologischen Wissensbasen realisiert wird.

Ich halte das *Competetive Learning* daher für den Einsatz im vorliegenden Anwendungsgebiet für wenig geeignet. Noch erwähnen möchte ich zwei weitere Modelle der nicht-überwachten Mustererkennung mit konnektionistischen Systemen, die aber auch nicht alle der erwähnten Probleme überwinden: die Adaptive Resonanz Theorie (ART) und Feature Maps.

ART

Die Adaptive Resonanz Theorie (ART) zeichnet sich gegenüber dem zuvor vorgestellten Verfahren vor allem dadurch aus, daß sie eine Überanpassung dadurch zu vermeiden sucht, daß gefundenen Mustern prototypische Eingabedaten zugeordnet werden. Weicht der so gefundene Prototyp zu stark vom Eingabewert ab, so wird das für die Erkennung dieses Musters verantwortliche Neuron gehemmt, so daß andere Neurone die Chance bekommen, die Eingabe zu klassifizieren. Obwohl dieser Aspekt für unser Modell von Vorteil wäre, so werden nicht alle der beschriebenen Probleme dadurch gelöst.

Die ART findet sich unter anderem in DORFFNER (1991) beschrieben.

Feature Maps

In Kohonens Feature Maps werden die Neurone in einer Nachbarschaftstopologie angeordnet. Hierbei lernt nicht nur das Neuron mit der höchsten Aktivität, sondern (abgestuft) lernen auch seine Nachbarn. Dieses Verfahren ist durchaus in der Lage, Ähnlichkeitsbeziehungen auszudrücken. Allerdings geschieht dies in Form einer topologischen Abbildung (daher auch der Name „Map“ für das Modell): es wird eine „Landkarte“ im Sinne einer Reduzierung des Merkmalsraumes aufgebaut. Ausführlich haben RITTER *et al.* (1991) die Feature Maps beschrieben.

Ich halte eine derartige Abbildung für wenig geeignet, epidemiologisch verwertbare Ähnlichkeitsbeziehungen auf den vorliegenden Zeitreihen aufzubauen. Eine topologische Beziehung dürfte nicht ausreichen, die denkbaren Arten von „Ähnlichkeit“ widerzuspiegeln. Auch dieses Modell habe ich daher verworfen.

4.3 k-means Clustering

Das z.B. in KAUFMAN UND ROUSSEEUW (1990) beschriebene k-means Clustering (k-fache Mittelwertclusterung, zum Teil auch verallgemeinertes Lloyd-Verfahren genannt) hat letztlich auch RUMMEL (1997) bei seinem ADSEQ-Algorithmus ausgewählt. Der k-means Clustering Algorithmus ist eine iterative Modifikation des allgemeinen Codebook-Ansatzes wie folgt:

Gegeben sei ein Menge $\mathcal{V}$ von Vektoren in einem n-dimensionalen Raum A_n. Auf $\mathcal{V}$ muß sich sinnvoll eine Distanzmetrik $Met\colon A_n \times A_n \to \mathbb{R}$ definieren lassen.

Verfahren

1. Ein *Codebook* $\mathcal{M}$ ist eine Menge von (nicht in der Eingabemenge gegebenen) Vektoren $m_i, i = 1, \ldots, k$, die $\mathcal{V}$ unter einer einfachen Nearest Neighbour-Annahme in k Cluster $\mathcal{C}_i$ zerlegen, so daß jedem Vektor v_j aus $\mathcal{V}$ ein Cluster C_b so zugeordnet wird, daß die Distanz $Met\,(v_j, m_j)$ minimal ist, d.h.:

$$b = \arg \min_{i=1,\ldots,k} Met\,(v_j, m_i) \tag{4.2}$$

2. Für jeden Cluster C_i wird das *Zentrum* $z(C_i)$ ermittelt, mit

$$z_i = z(C_i) = \frac{1}{|C_i|} \sum_{v_{i_j} \in C_i} v_{i_j} \tag{4.3}$$

3. Das *Codebook* $\mathcal{M}$ wird durch die Menge $\{z_1, \ldots, z_k\}$ ersetzt, die das neue *Codebook* bildet. Daraufhin wird die Clusterung gemäß Schritt 1 neu bestimmt.

4. Der Prozeß wird mit Schritt 2 fortgesetzt, bis

   (a) die *Codebook*-Vektoren konvergieren, oder

   (b) die *Distortion* D als Gütemaß sich nicht weiter vermindert.

Die *Distortion* D ist ein Maß der „Unreinheit" eines Clusters, d.h. der aufsummierten Abweichung der einem Cluster C_i zugeordneten Vektoren v_{i_j} vom jeweiligen *Codebook*-Vektor m_i. Sie läßt sich wie folgt berechnen:

$$D_i = \frac{1}{|C_i|} \sum_{j=1}^{k} \sum_{v_{i_j} \in C_i} Met\left(v_{i_j}, m_i\right) \tag{4.4}$$

Beide Abbruchkriterien können die Qualität der Clusterbildung nur verbessern. Falls keine Verbesserung möglich ist, bleiben die *Codebook*-Vektoren unverändert.

Es kann sein, daß ein Cluster C_j leer bleibt, d.h. daß kein Vektor aus der Eingabemenge dem entsprechenden *Codebook*-Vektor zugeordnet wird. In diesem Fall eröffnen sich drei Möglichkeiten:

1. Entferne den entsprechenden *Codebook*-Vektor m_j aus $\mathcal{M}$ und reduziere somit k.

2. Suche den Cluster C_i mit der größten Anzahl an zugeordneteten Vektoren aus der Eingabemenge $\mathcal{V}$, d.h.

$$i = \arg \max_{i=1,\ldots,k} |C_i| \tag{4.5}$$

   Teile diesen Cluster in zwei Teile durch Berechnung eines weiteren *Codebook*-Vektors c_{new} für C_i. Ersetze m_j durch c_{new}.

3. Suche den Cluster C_i mit der größten *Distortion* D, d.h.

$$i = \arg \max_{i=1,\ldots,k} D_i \tag{4.6}$$

   Teile diesen Cluster in zwei Teile durch Berechnung eines weiteren *Codebook*-Vektors c_{new} für C_i. Ersetze m_j durch c_{new}.

Das k-means Clustering hat den Vorteil, daß es garantiert in einer endlichen Anzahl von Iterationen konvergiert.[3] Da es ein unüberwachtes Verfahren ist, geht kein Wissen über die Verteilung der Klassen in die Clusterung ein. Somit wird vermieden, daß ein im zweiten Schritt folgendes Induktionsverfahren eine Klassifikation aufgrund von Wissen findet, das an dieser Stelle bereits in das Lernverfahren eingeht (wie in der Einleitung dieses Kapitels beschrieben).

[3] Vgl. (RUMMEL, 1997, S. 38).

Adaptives Clustering

Bei einem adaptiven k-means Clustering wird der Wert von k nicht vorab festgelegt, vielmehr wird der Wert aus $\mathcal{V}$ bestimmt und es werden nur eine untere Schranke l und eine obere Schranke u mit $l \leq u$ für k angegeben.

Dann wird das skizzierte Verfahren für alle Werte von k mit $l \leq k \leq u$ durchgeführt. Die Auswahl des besten Wertes von k erfolgt anhand einer *Fitting*-Funktion. Es gilt einerseits eine Überanpassung der Cluster und damit einhergehende schlechte Fähigkeiten zur Generalisierung zu verhindern. Andererseits bedeutet eine große Anzahl von Clustern, daß die Cluster übergeneralisieren, was wiederum die „Ausdrucksmächtigkeit" einschränkt.

Zwei Kriterien zur Auswahl von k möchte ich vorstellen (vgl. RUMMEL (1997)):

1. **Informationstheoretisch:** Ein Maß, welches sich auf Überlegungen zum möglichen Informationsgewinn abstützt: sei zuerst die *Entropie* eines Clusters C_i analog zur Gleichung 3.3 definiert, wobei $E = C_i$ sei.

   Dann ist die in einer Clusterung $\mathcal{M}$ enthaltene Information wie folgt definiert:

   $$Info_{\mathcal{M}}(\mathcal{V}) = -\sum_{i=1}^{k} \frac{|C_i|}{|\mathcal{V}|} Entropie(C_i) \tag{4.7}$$

   Ferner ist der durch diese Clusterung erreichte Informationsgewinn zu errechnen mit:

   $$Gain\,(\mathcal{M}) = Entropie\,(\mathcal{V}) - Info_{\mathcal{M}}(\mathcal{V}) \tag{4.8}$$

   Dann gilt es, den Cluster mit dem höchsten Informationsgewinn gegenüber dem Informationsinhalt von $\mathcal{V}$ zu finden. Sei $\mathcal{M}_k$ das für k gefundene *Codebook*. Dann gilt:

   $$k_{best} = \arg \max_{k=l,\ldots,u} Gain(\mathcal{M}_k) \tag{4.9}$$

   Der Nachteil dieser Methode liegt darin, daß die Klassenverteilung, wenn schon nicht in die Konstruktion der Cluster, so doch in die Auswahl der besten Cluster eingeht, da diese Information in Gleichung 4.7 benutzt wird.

2. **Distortion:** Eine andere Möglichkeit ist es, Gleichung 4.4 zu nutzen, um die Clusterung mit der niedrigsten Abweichung der Vektoren von den *Codebook*-Vektoren zu bestimmen:

   $$k_{best} = \arg \min_{k=l,\ldots,u} D_i \tag{4.10}$$

   Da bei einer größeren Anzahl von *Codebook*-Vektoren die *Distortion* im Durchschnitt absinken wird, ist es notwendig, daß ein großes k bei dieser Methode mit einer Strafe belegt wird. Dies kann als eine Form des Simulated Annealing (vgl. 2.6.2) aufgefaßt werden.

Resumee

Die Wahl der Metrik *Met* beeinflußt die Clusterbildung. Die Auswahl einer Metrik bietet somit Möglichkeiten der Modellierung von Hintergrundwissen. Nachteilig ist allerdings, daß eine automatische Auswahl einer Metrik nicht trivial ist, so daß für ein vollautomatisches Verfahren eine geeignete Heuristik zu entwickeln wäre.

Die Laufzeit des Verfahrens liegt im Bereich $O(n^2)$ im worst case, n sei die Anzahl der Vektoren in $\mathcal{V}$ (vgl. für eine detailliertere Analyse (RUMMEL, 1997, S. 56)).

Ein adaptives k-means Clustering mit *Distortion* als Maß für das Auffinden des besten Wertes von k erscheint mir als probates Clustering-Verfahren im hier betrachteten Anwendungsgebiet. Eine unscharfe Formulierung von Zugehörigkeiten ergibt sich aus der Distanz eines Vektors v zu allen möglichen *Codebook*-Vektoren.

Ein großes Problem bei der Nutzung des Verfahrens liegt sicher darin, daß die benutzten Metriken nicht translationsinvariant bzgl. der Zeit sind. Zwei Zeitreihen, die ein menschlicher Experte als ähnlich einstufen würde, weil sie beide genau einen charakteristischen Peak aufweisen, allerdings zu unterschiedlichen Zeitpunkten, werden vom k-means Clustering nicht ohne weiteres als ähnlich erkannt werden. Dieses Problem tritt allerdings nicht nur bei diesem Algorithmus auf.

4.4 Learning Vector Quantisation

Die Learning Vector Quantisation (LVQ) geht auf Kohonen zurück und wird u.a. in RUMMEL (1997) beschrieben. LVQ ist im Gegensatz zum k-means Clustering ein überwachtes Lernverfahren. Im folgenden beschreibe ich kurz eine LVQ-Variante zur Approximation einer idealen Clusterverteilung.

Wie beim k-means Clustering wird einem Beispiel v aus der Menge der Eingabevektoren $\mathcal{V}$ der *Codebook*-Vektor m_i zugewiesen, zu der unter einer Metrik *Met* der Abstand minimal ist. Allerdings haben die *Codebook*-Vektoren im Unterschied zum vorhergehenden Verfahren selber eine Klassenzugehörigkeit, und während wir vorher zum Ziel hatten, die *Distortion* innerhalb eines Cluster zu minimieren, wollen wir jetzt Cluster aus Vektoren mit der gleichen Klassenzugehörigkeit finden.

Ziel ist also, die Rate falscher Klassifikationen in den Clustern zu minimieren. Diese ist definiert als die durchschnittliche Anzahl von Vektoren aus C_i, deren Klassenzugehörigkeit von der Klassenzugehörigkeit des zugehörigen *Codebook*-Vektors abweicht. Sei $label(v)$ die Klassenzugehörigkeit eines Vektors v aus der Menge $\mathcal{M} \cup \mathcal{V}$ und t ein diskretes Zeitmaß für die Iterationsschritte.

Dann werden die *Codebook*-Vektoren wie folgt neu berechnet:

1. Die Menge $\mathcal{V}$ der Eingabevektoren wird wie beim k-means Clustering unter einer Metrik *Met* und einem *Codebook* $\mathcal{M}$ in k Cluster zerlegt.

2. Danach wird für alle Vektoren $m_i(t)$ aus $\mathcal{M}$ zum Zeitpunkt t ein neuer Vektor $M_i(t+1)$ berechnet. Dazu wird iteriert $\forall v \in \mathcal{V}, v \in C_i$:

$$m_i(t+1) = \begin{cases} m_i(t) + \alpha_i(t)(v - m_i(t)) & \text{falls } label(v) = label(m_i), \\ m_i(t) + \alpha_i(t)(v - m_i(t)) & \text{sonst.} \end{cases} \tag{4.11}$$

wobei die Lernrate $\alpha_i(t) \in [0,1]$ wie folgt definiert ist:

$$\alpha_i(t) = \min\left(1 - \delta, \frac{\alpha_i(t-1)}{1 + s(t)\alpha_i(t-1)}\right) \tag{4.12}$$

mit

$$s(t) = \begin{cases} +1 & \text{falls } label(v) = label(m_i), \\ -1 & \text{sonst.} \end{cases} \tag{4.13}$$

Die Lernrate α verändert sich also mit jedem Update der *Codebook*-Vektoren, wobei sie größer wird, wenn ein Vektor falsch klassifiziert wird, sonst sinkt sie. Die min-Funktion wird als obere Grenze der Lernrate benutzt mit $\delta > 0$ nahe 0.

3. Der Prozeß wird fortgesetzt, bis eine hinreichend gute Aufteilung von $\mathcal{V}$ erreicht ist.

Resumee

Im allgemeinen wird sich die Lernrate α im Laufe der Zeit 0 annähern, so daß bei einer genügenden Anzahl von Iterationen eine Konvergenz erreicht wird. Falls die Lage der Vektoren aus $\mathcal{V}$ allerdings in keinem Verhältnis zu ihrer Klassenzugehörigkeit steht, wird der Algorithmus nicht konvergieren, sofern kein Simulated Annealing durchgeführt wird.

Die ursprünglichen Werte der *Codebook*-Vektoren spielen eine genauso große Rolle für den Erfolg des Verfahrens wie ihre gewählte Klassenzugehörigkeit. Das gleiche gilt für den Wert der Lernrate α. Beides ist abhängig von den zu partitionierenden Daten. Daher erfordert LVQ ein großes Maß an Interaktion mit dem Benutzer.

Dazu kommt noch ein relativ hoher Laufzeitaufwand und der von mir aufgrund bereits genannter Gründe abgelehnte Einsatz des Wissens über die Klassenzugehörigkeit, so daß ich die LVQ-Methode für ungeeignet für den Einsatz im vorliegenden Problembereich halte.

4.5 Dynamisches Programmieren

Zum Schluß möchte ich noch einen symbolischen Ansatz von BERNDT UND CLIFFORD (1996) vorstellen. Ziel des *Dynamic Time Warping* genannten Verfahrens ist es, Muster in Zeitreihen zu finden. Solche Muster sind Ausschnitte mit einem spezifischen „Aussehen“ und werden auch von menschlichen Experten zur Beschreibung von Phänomenen benutzt.

Sobald identifiziert, können solche Muster dann zur Beschreibung von Zusammenhängen zwischen verschiedenen Zeitreihen dienen, wie z.B. dieser: „Einer Spitze in der Beutetierpopulation folgt zumeist eine Spitze in der Population von Räubern.“ Das vorgestellte Verfahren läßt sich also gut auf unsere Problemstellung übertragen.

Berndt und Clifford erläutern ihr System in Analogie zum Verstehen menschlicher Sprache, wo es auch darauf ankommt, gesprochene Worte dadurch zu erkennen, daß diese mit einer unscharfen Funktion auf bereits bekannte, abgespeicherte Worte abgebildet werden. Dies geschieht häufig darüber, daß eine diskretisierte Zeitreihe des gesprochenen Wortes mit Schablonen für bekannte Worte verglichen wird, wobei die Erkennung trotz einer gewissen zeitlichen und aussprachebedingten Varianz gelingen soll.

Die Technik des *Dynamic Time Warping* benutzt einen als *dynamisches Programmieren* bezeichneten Ansatz, welcher die Zeitreihen und Schablonen über die Minimierung einer Distanzmetrik in Beziehung setzt. Dabei ist die Zeitachse der Schablone in einem weiten Gebiet stauch- bzw. streckbar, so daß eine Schablone für mehrere verschiedene Zeitreihen gültig ist.

Wellenlinie

Seien eine Zeitreihe Z und eine Schablone S gegeben, $Z = z_1, z_2, \ldots, z_i, \ldots, z_n$ sowie $S = s_1, s_2, \ldots, s_j, \ldots, s_m$. Dann können diese beiden Folgen in Form eines $n \times m$ Rasters aufgetragen werden, wobei jeder Punkt (i, j) einer Zuordnung zwischen einem Element z_i der Zeitreihe und einem Element s_j der Schablone entspricht. Ein Pfad W_l ist eine Folge $W_l = w_{l_1}, w_{l_2}, \ldots, w_{l_p}$ mit $w_{l_k} = (i, j)_{l_k}$. Eine *Wellenlinie* DTW ist ein ausgezeichneter Pfad, der die Elemente aus Z und S derart verbindet, daß der Abstand bzgl. einer Abstandsfunktion

$$\delta : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$$

minimiert wird. Dabei gilt, daß bei Übereinstimmung zwischen Z und S ohne Zeitdifferenz jeweils $i = j$ gilt.

Für δ sind grundsätzlich viele Distanzmetriken denkbar. Durch die Wahl der Metrik kann implizit eine Ordnung auf den Ähnlichkeitsbeziehungen zwischen Schablone und Zeitreihe festgelegt werden. Denkbar sind auch Metriken, die relative Abstände benutzen.

In BERNDT UND CLIFFORD (1996) werden als einfache Differenzmetriken

$$\delta(i, j) = |z_i - s_j| \tag{4.14}$$

und

$$\delta(i, j) = (z_i - s_j)^2 \tag{4.15}$$

vorgeschlagen.

Sei $\mathcal{P}$ die Menge aller Pfade auf dem Raster. Eine *Wellenlinie* ist damit formal definierbar als der Pfad über das Raster mit einem minimalen aufsummierten Abstand zwischen Schablone und Zeitreihe, also:

$$DTW(Z, S) = \arg \min_{W_l \in \mathcal{P}} \sum_{k=1}^{l_p} \delta(w_{l_k}) \tag{4.16}$$

Dabei ist der Suchraum, hier also der Raum der zu betrachtenden möglichen Pfade, über folgende Bedingungen einschränkbar:

1. **Monotonie:** Die Punkte einer Wellenlinie müssen in der Zeit geordnet sein, also gilt für zwei aufeinanderfolgende Punkte $w_{l_{k-1}}$ und w_{l_k}:

$$i_{l_{k-1}} \leq i_{l_k} \text{ sowie } j_{l_{k-1}} \leq j_{l_k}. \tag{4.17}$$

2. **Kontinuität:** Zwei Punkte einer Wellenlinie liegen höchstens ein Element auf dem Raster auseinander, d.h.:

$$i_{l_k} - i_{l_{k-1}} \leq 1 \text{ und } j_{l_k} - j_{l_{k-1}} \leq 1. \tag{4.18}$$

3. **Fenster:** Eine zusätzliche Bedingung kann sein, daß die Punkte in ein Zeitfenster fallen, also keine beliebige Dehnung bzw. Stauchung der Schablone zugelassen wird. Dann muß gelten:

$$|i_{l_k} - j_{l_k}| \leq \omega \text{ mit } \omega \in \mathbb{N}. \tag{4.19}$$

4. **Steigungsbedingung:** Die Steigung kann eingeschränkt werden, um Verzerrungen der Schablone zu minimieren.

5. **Grenzbedingungen:** Anfangs- und Endpunkte der möglichen Wellenlinie können vorgegeben werden.

In einem dynamischen Verfahren wird jetzt nicht über alle möglichen Pfade der minimale ausgewählt, sondern es wird ein Backtracking-Verfahren benutzt: die aufsummierte Distanz $\gamma(i,j)$ für einen Punkt (i,j) auf dem Raster ist eindeutig definiert durch den Abstand von Schablone und Zeitreihe in diesem Punkt, also $\delta(i,j)$, und $\gamma(g,h)$ für den benachbarten Punkt (g,h) mit der minimalen aufsummierten Distanz aller Nachbarpunkte. Für einen gegebenen Punkt (i,j) läßt sich $\gamma(i,j)$ dann wie folgt darstellen:

$$\gamma(i,j) = \delta(i,j) + \min[\gamma(i-1,j), \gamma(i-1,j-1), \gamma(i,j-1)] \tag{4.20}$$

Diese Werte können in einer Tabelle aufgetragen werden, so daß zur Auswahl des besten Pfades, also der *Wellenlinie*, in der Tabelle jeweils der Nachbarpunkt mit der kürzesten aufsummierten Entfernung genommen wird.

Musterauswahl

Nach dem Auffinden einer *Wellenlinie* interessiert uns jetzt, wie gut eine Schablone durch Stauchung und Streckung mit einer Zeitreihe zusammenpaßt. Ein solches Maß sollte es ermöglichen, die einer Zeitreihe ähnlichste Schablone zu finden. Auch sind letztlich mehrfache Anwendungen auch unterschiedlicher Schablonen gegenüber einer Zeitreihe mit unterschiedlichen Startzeitpunkten anhand ihrer jeweiligen Übereinstimmung zu ordnen.

In einem ersten Schritt ist dazu eine Normierung der gefundenen aufsummierten Distanz notwendig. Deren Größe ist z.B. stark von der Amplitude der Zeitreihe abhängig. Eine einfache Variante wäre die Normalisierung anhand von Ankerpunkten, z.B. des Wertes des Anfangspunktes der Zeitreihe.

Wenn wir jetzt an einer relativen Fitness interessiert sind, und die passende Schablonen nicht aufgrund ihrer normalisierten aufsummierten Distanz auswählen wollen, so ist in einem

zusätzlichen zweiten Schritt noch die Bildung eines relativen Maßes notwendig. Dies kann z.B. bereits dadurch erfolgen, daß die Distanz mit der Distanz zu einer Grundlinie gewichtet wird, die dem Durchschnitt aller Zeitreihen entspricht.

Dadurch erhält man letztlich für jede Zeitreihe und jede dagegen getestete Schablone ein Ähnlichkeitsmaß, anhand dessen die ähnlichste Schablone ausgewählt werden kann. In einem nachfolgenden Schritt könnten dann Beziehungen zwischen den derart durch Schablonen beschriebenen Zeitreihen untersucht werden.

Resumee

Ich halte das *Dynamic Time Warping*-Verfahren für vielversprechend im Hinblick auf den untersuchten Gegenstandsbereich. Die gefundenen Repräsentationen von Merkmalen in den Zeitreihen kommen dem Menschen entgegen, wenn den einzelnen Schablonen beispielsweise symbolische Namen zugeordnet werden.

Die Autoren sehen allerdings bei großen Datenmengen Probleme in der Performance und schlagen hierzu noch weitere Preprocessing-Schritte vor.

Einen weiteren Nachteil sehe ich in der Tatsache, daß die Schablonen z.B. von menschlichen Experten vorgegeben werden müssen, und sich nicht automatisch aus den Zeitreihen erzeugen lassen. Dies ermöglicht zwar einerseits die Integration von Hintergrundwissen. So ist es z.B. bei einem für den Einsatz in der Beobachtung von Aktienkursen vorgesehenen System von Vorteil, charakteristische Verlaufsformen von Börsenkursen als Schablonen zu nehmen. Andererseits wird die Verwendbarkeit in solchen Problembereichen eingeschränkt, in denen dieses Hintergrundwissen nicht vorliegt oder vorliegendes Wissen schwer in Form von Schablonen zu fassen ist.

Ich habe mich aufgrund dieser Punkte letztlich gegen die Nutzung des Verfahrens entschieden. Die weitere Forschung auf diesem Gebiet sollte aber bei der Weiterentwicklung eines Systems zur automatischen Klassifikation von Zeitreihen beobachtet werden.

4.6 Bewertung

Von allen vorgestellten Systemen erscheint mir das k-means Clustering das für eine prototypische Implementierung im vorliegenden Gegenstandsbereich am besten geeignete zu sein.

Es konvergiert garantiert bei einem akzeptablen Laufzeitverhalten. Das Ergebnis ist m.E. gut kommunizierbar, vor allem, wenn man die Zuordnung der einzelnen Zeitreihen zu den Prototypen visualisiert.

Es ist sowohl eine disjunkte Zerlegung in nominale Attribute möglich, die für die meisten symbolischen Systeme des Maschinellen Lernens geeignet ist, als auch eine sinnvolle Kodierung in Form von unscharfen Zugehörigkeitsfunktionen für Neuronale Fuzzy-Systeme.

Die Einbindung von Hintergrundwissen kann in eingeschränkter Form über die Wahl der Metriken geschehen. Gleichzeitig ist ein naives Vorgehen möglich, indem einfache Standard-Metriken eingesetzt werden. Darunter leidet allerdings die Aussagekraft der Prototypen.

Konnektionistische Verfahren leiden einerseits an einem teilweise recht hohen Laufzeitaufwand für das Training. Andererseits tritt auch an dieser Stelle das bereits mehrfach angesprochene Problem der Interpretierbarkeit in den Vordergrund.

Das LVQ-Verfahren scheidet vor allem aus wegen der Notwendigkeit, die Klassenzugehörigkeit im Clusteringprozeß zu benutzen. Zunächst einmal könnte dadurch ein logischer Zirkel entstehen, da ich in einem zweiten Schritt wiederum nach Faktoren suche, welche die jeweilige Klassenzugehörigkeit erklären können. Als weiteren Nachteil sehe ich das unsichere Konvergenzverhalten an, wenn die gerade untersuchte Zeitreihe nicht in kausaler Abhängigkeit zur Klassenzugehörigkeit steht.

Das ansonsten sehr interessante Verfahren der dynamischen Programmierung leidet meiner Meinung nach stark am Problem einer effizienten Programmierung. Das halte ich für einen großen Nachteil, da durch Simulationen prinzipiell eine große Menge an auszuwertenden Daten erzeugt werden kann.

Ich habe mich daher zur Implementierung eines Algorithmus' des k-means Clustering entschieden.

Teil II

Praxis

Kapitel 5

Implementierung

*The fruits of knowledge growing on the tree of data are not easy to pick (*Frawley *et al., 1992, S. 67).*

5.1 Einleitung

In diesem Kapitel stelle ich die von mir realisierte Implementierung eines prototypischen Verfahrens vor, das zur Generierung von Hypothesen über kausale Zusammenhänge dient. Dazu sollen in einer Menge von Informationen, die über *Objekte* einer Eingabemenge vorliegen, diejenigen Bestandteile identifiziert werden, die eine gegebene *Klassifikation* der Eingabe begründen können. Die Informationen über die Objekte liegen in Form von *Attribut-Wert-Paaren* vor.

Die im Zusammenhang mit dem hier vorgestellten System stehenden Untersuchungen sind im Rahmen von Arbeiten motiviert, welche die Erstellung eines Softwarewerkzeugs zur Unterstützung bei der Planung und Durchführung umweltepidemiologischer Studien zum Ziel haben.

Insbesonders ist mit diesem Bezug die Aufbereitung realer Daten definiert, die dort auf detaillierten Simulationsmodellen beruht. Dies schwächt im besonderen solche Probleme ab, die in fehlenden oder fehlerhaft erhobenen Daten sowie nicht-äquidistanten Erhebungsfunktionen begründet liegen (vgl. Köster und Sonnenschein (1999)).

5.1.1 Konzept

Bei dem vorgestellten Verfahren handelt es sich um ein zweistufiges, hybrides System aus einer subsymbolischen und einer symbolischen Komponente (vgl. Abbildung 5.1). Die symbolische zweite Komponente stellt die eigentliche Hypothesengenerierung dar. Sie setzt allerdings einen vorhergehenden subsymbolischen Verarbeitungsschritt voraus, mit dem für die einzelnen Zeitreihen symbolische Repräsentationen gefunden werden.

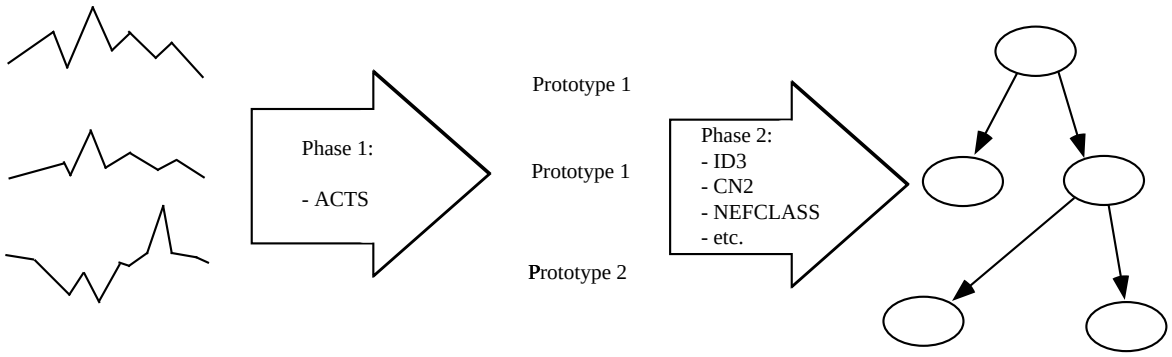

Abbildung 5.1: *Das vorgeschlagene System teilt sich in eine subsymbolische erste Verarbeitungsphase und eine symbolische zweite.*

Die im Kapitel 3 vorgestellten symbolischen Lernverfahren können im allgemeinen nicht direkt auf Attributen arbeiten, die in sich stark strukturiert sind, wie die vorliegenden Zeitreihen. Eine Möglichkeit zur Integration von Attributen mit derartigem Wertebereich bietet allein das ADSEQ-Verfahren (vgl. Unterkapitel 3.2.2).

Da im Rahmen der vorliegenden Arbeit prinzipiell die Frage untersucht werden soll, ob Data Mining Verfahren aus dem Bereich Maschinelles Lernen zur Generierung von relevanten epidemiologischen Hypothesen in der Lage sind, ist hier der Weg einer strikten Trennung der Aufbereitung der Zeitreihen und ihrer Analyse gegangen worden.

Das eröffnet einerseits die Möglichkeit, in der zweiten Phase unterschiedliche Verfahren auf ihre Eignung hin zu überprüfen. Andererseits können so auch unterschiedliche subsymbolische Verfahren in das System integriert werden.

5.1.2 Aufbau

Im folgenden Unterkapitel 5.2 wird die Realisation der ersten Phase des vorgelegten Konzeptes genauer beschrieben. Hierbei handelt es sich um das im Rahmen der Arbeit implementierte ACTS-System (Adaptive Clustering of Time Series).

Das zweite Unterkapitel 5.3 beschreibt, wie in der zweiten Phase Programme des symbolischen Maschinellen Lernens und ein Neuro-Fuzzy Verfahren zur Bildung von Hypothesen genutzt werden können.

5.2 Erste Phase

Zwischen den im betrachteten Gegenstandsbereich vorliegenden Zeitreihen ließe sich nach vielen unterschiedlichen Beziehungen suchen. Letztlich gewünscht ist allerdings eine Reduktion der Zeitreihen auf eine Repräsentationsform, mit der Verfahren des symbolischen Maschinellen Lernens arbeiten können.

Diese Reduktion ist komplexitätsreduzierend, da sie nicht jeder Zeitreihe ein einzelnes Symbol zuordnet, sondern mehrere Zeitreihen auf ein und dasselbe Symbol abbildet. Sie soll dabei insofern informationserhaltend sein, als daß diese Abbildung nicht zufällig ist. Vielmehr soll dabei eine Zuordnung vorgenommen werden, die im Sinne der Suche nach Kausalzusammenhängen funktional ist.

5.2.1 Anforderungen

Die Abbildung soll also die Zeitreihen, die auf den Ausbruch der zu untersuchenden Krankheit in der gleichen spezifischen Weise wirken, auf ein Symbol abbilden. Zeitreihen, die einen unterschiedlichen Effekt auf den Ausbruch der Krankheit haben, sollen auf unterschiedliche Symbole reduziert werden.

Ich stehe damit vor dem logischen Problem, die Zeitreihen aufgrund einer kausalen Beziehung einzuteilen, die noch gar nicht bekannt ist, sondern erst durch das vorgestellte System zu generieren ist.

Schlimmer noch: der in der zweiten Stufe arbeitende Algorithmus sollte keine kausalen Abhängigkeiten finden, die in der ersten Stufe hineingesteckt wurden. Dies wäre ein logischer Zirkelschluß und würde mich dem eigentlich verfolgten Ziel nicht näherbringen (vgl. die Diskussion im Unterkapitel 4.1).

Das in der ersten Stufe zu lösende Problem ist also, ohne Kenntnis der Kategorisierung der Daten eine Aufteilung der Zeitreihen vorzunehmen, die zumindest der Möglichkeit nach kausale Zusammenhänge ausdrückt.

Ich gehe davon aus, daß bestimmte Eigenschaften der Zeitreihen in kausaler Beziehung zum Ausbruch der Krankheit stehen. Das kann beispielsweise die individuelle Exposition gegenüber einem Schadstoff sein, die über einen längeren Zeitraum aufgezeichnet ist. Es handelt sich also um ein phänomenologisches Problem: der vermutete kausale Zusammenhang spiegelt sich im „Aussehen“ der Zeitreihen wider.

5.2.2 Vorgehen

Ich suche also eine Partitionierung der Zeitreihen so, daß Zeitreihen, die auf eine bestimmte Art und Weise ähnlich sind, in einer Partition liegen, während solche, die sich nicht ähnlich sind, in unterschiedlichen Partitionen liegen.

Von den im Überblick zu Clusteringverfahren (Kapitel 4) vorgestellten Verfahren ist das k-means Clustering im Acts-System prototypisch implementiert, um dieses Ziel zu erreichen. Die geforderte phänomenologische Ähnlichkeit wird dabei über die notwendigen Distanz- resp. Ähnlichkeitsmetriken realisiert.

Weil diese Metriken, im besonderen auch für die Integration von Hintergrundwissen, bedeutsam sind, sind sie in einem eigenen Kapitel 6.1 beschrieben.

Direkte Zuordnung

Das k-means Clustering liefert eine Abbildung aus der Menge von Zeitreihen $\mathcal{T}$ in die Menge der Codebook-Vektoren $\mathcal{P}$ mit $|\mathcal{P}| = k$.

$$Cluster : \mathcal{T} \rightarrow \mathcal{P} \tag{5.1}$$

Jede Zeitreihe T aus der Eingabemenge $\mathcal{T}$ ist genau dem Cluster $\mathcal{P}_j$, zugeordnet, zu dessen Codebook-Vektor P_j sie die größte Ähnlichkeit respektive geringste Distanz bezüglich der verwendeten Metrik hat. P_j gilt als sogenannter Repräsentant dieser Klasse. In der von der symbolischen Komponente generierten Hypothese nimmt P_j eine stellvertretende Funktion für alle Zeitreihen in $\mathcal{P}_j$ ein.

Die Codebook-Vektoren $P_0, P_1, \ldots, P_{k-1}, P_i \in \mathcal{P}$ nenne ich *Prototypen*. Jeder Vektor V aus $\mathcal{V}$ wird im Laufe des Clusterings genau einem Prototypen P_j zugeordnet. Die gesuchte komplexitätsreduzierende Abbildung

$$Proto : \mathcal{T} \rightarrow \mathbb{N} \tag{5.2}$$

ist dann definiert als:

$$Proto(V) = j \Leftrightarrow Cluster(T) = P_j \tag{5.3}$$

Damit ist $j, 0 \leq j \leq k-1$ eine symbolische Repräsentation des zugrunde liegenden Protypen.

Zugehörigkeitsfunktionen

Ein wenig anders ist der Fall gelagert, wenn im zweiten Schritt ein Neuro-Fuzzy System die Bildung der Hypothese übernehmen soll. Um Neuro-Fuzzy-Klassifikatoren nutzen zu können, ist es notwendig, eine Darstellungsform zu finden, die Zugehörigkeitsfunktionen über den Zeitreihen entspricht. Da ein auf Ähnlichkeitsmaßen basierendes Klassifikationsverfahren benutzt wird, ist es prinzipiell möglich, für jede Zeitreihe das Maß der Zugehörigkeit zu jedem Cluster als Distanz zu dem Codebook-Vektor jedes Cluster unter der verwendeten Metrik anzugeben.

In diesem Fall ist also nicht eine direkte Zuordnung zu einem der k Cluster gefordert, vielmehr wird für jede Zeitreihe T unter der verwendeten Metrik *Met* ihre Zugehörigkeit zu allen Prototypen berechnet.

Die realisierten Abbildungen lauten also:

$$Fuzzy_j : T \rightarrow \mathbb{R} \tag{5.4}$$

mit $j, 0 \leq j \leq k-1$

$$Fuzzy_j(T) = Met(T, P_j) \tag{5.5}$$

Es ist hierbei nicht kritisch, ob *Met* eine Ähnlichkeits- oder Distanzmetrik ist. Dies gilt zum einen, da beide Metriken sich ineinander überführen lassen (vgl. RUMMEL (1997)). Zum anderen lassen sich die Zugehörigkeiten bzgl. einer Metrik zudem als Zugehörigkeiten zu einer durch ihr jeweiliges Pendant beschriebenen (unscharfen) Komplementärklasse auffassen.[1]

[1] Zur Definition des Mengenbegriffes von Fuzzy-Systemen vgl. z.B. KRUSE (1996).

5.2.3 Eigenschaften

ACTS ist in der Lage, Spuren von Individuen aus einer Datei einzulesen, ein adaptives k-means Clustering durchzuführen, und Dateien für die Weiterverarbeitung mit verschiedenen induktiven Lernverfahren zu erzeugen. Darüber hinaus werden zur Visualisierung der gefundenen Partitionierungen weitere Dateien erzeugt. Das Verhalten des Systems ist vollständig über Umgebungsvariablen steuerbar. Es ist aber auch möglich, ACTS interaktiv zu benutzen.[2]

Eine kurze Übersicht über die benutzten Umgebungsvariablen findet sich in Anhang D.2. Beispiele für die Formate aller Ausgabe- und Eingabedateien finden sich im Anhang C.

Die zum Clustering verwendete Metrik kann einerseits global für alle Zeitreihen gleich vorgegeben werden. Andererseits ist es zur Integration von Hintergrundwissen über die Art der Zeitreihen auch möglich, in einer Steuerdatei für jede Zeitreihe eine der installierten Metriken auszuwählen.

Clustering

Obwohl die im Verlauf des k-means Clustering erzeugten Codebook-Vektoren letztlich artifiziell sind, d.h. daß sie nicht aus der Eingabemenge stammen, so muß das Verfahren mit einer ersten Variante von Vektoren starten. RUMMEL (1997) weist auf die Möglichkeit hin, diese Startvektoren aus der Eingabemenge zu nehmen.

Für die Initialisierung des k-means Clustering benutze ich daher jeweils die ersten Zeitreihen der Eingabedaten. Die Zeitreihen werden so ausgewählt, daß bei geradem k die Anzahl der Zeitreihen von kranken und gesunden Individuen gleich ist und ansonsten die Anzahl von kranken Individuen um eins größer ist. Dies geschieht aus pragmatischen Erwägungen und *kann* bei stark unterschiedlichem Aussehen der Zeitreihen von gesunden und kranken Individuen die Konvergenz beschleunigen.

Für das adaptive Clustering benutzt ACTS ein auf der Berechnung der Distortion (vgl. Gleichung 4.4) beruhendes Auswahlverfahren. Da sich bei einer größeren Anzahl an Clustern die Distortion im Durchschnitt verringert, wird ein Simulated Annealing durchgeführt (vgl. Unterkapitel 2.6.2).

Hierzu wird die Gesamt-Distortion mit einem Faktor von $l/5$ belegt, mit l Anzahl der Cluster. Wie allgemein üblich, wurde dieser Faktor experimentell ermittelt. Er führt im Durchschnitt zur Bildung einer Anzahl von Clustern, die unter der erlaubten maximalen Anzahl liegt.[3] Für Experimente auf Realdatensätzen ist sicherlich eine genauere experimentelle Ermittlung von möglichen Funktionen des Simulated Annealing erforderlich.

Wie bei der Beschreibung des k-means Clustering ausgeführt, können während des Clustering-Prozesses einzelne Partitionen leer bleiben. ACTS bietet hierzu zwei Möglichkeiten. Standardmäßig werden leere Cluster entfernt. Somit wird k im entsprechenden Lauf verringert. Es können aber auch die Cluster mit den meisten Einträgen aufgeteilt werden.

[2] Wenn die Umgebungsvariable PROMPTLEVEL den Wert „all“ hat, wird jede (relevante) Option abgefragt.
[3] Die gefundene Anzahl von Clustern ist abhängig von der verwendeten Metrik und den Eingabdaten.

Erste, nicht repräsentative Tests ließen keine Verbesserung erkennen, die den zusätzlichen Aufwand für die Aufteilung der Cluster rechtfertigen würde. Weder veränderte sich die Anzahl der letztlich gefundenen Partitionen, noch wurden in der zweiten Phase unterschiedliche Hypothesen generiert. Aber auch an dieser Stelle sollten bei Vorliegen von Realdaten weitere Untersuchungen durchgeführt werden.

5.2.4 Eingabe

ACTS benötigt als Eingabe eine Datei, in der die Attribute und Zeitreihen für jedes Individuum beschrieben sind. Diese Eingabedateien haben die Endung `*.db`. Ich benutze hierbei das von RADTKE (1998) vorgeschlagene Format, um eine leichte Integration verschiedener Komponenten sicherzustellen.

So können einerseits mit ACTS gefundene Cluster dem von Radtke skizzierten Analysesystem auf Grundlage genetischer Algorithmen als zusätzliche Information zur Verfügung gestellt werden. Andererseits ist es möglich, für die Darstellung von Zeitreihen in ACTS Klassen aus der Arbeit von Radtke einzubeziehen.

Bei dieser Eingabedatei handelt es sich um ein *flat file*, in dem die Daten für die einzelnen Individuen hintereinander beschrieben sind. Für die Arbeit mit großen Realdatensätzen, die nicht mehr im Hauptspeicher vorgehalten werden können, sind an dieser Stelle sicherlich Änderungen des Systems notwendig.

Dabei ist es von Vorteil, daß die einzelnen Attribute unabhängig voneinander partitioniert werden können. Allerdings benötigen Clustering-Verfahren prinzipiell Zugriff auf alle Daten, so das es wünschenswert ist, zumindest für jedes Attribut die Daten aller Individuen im Hauptspeicher vorliegen zu haben.

Falls für einzelne Attribute unterschiedliche Metriken verwendet werden sollen, so kann dies in einer Datei mit der Endung `*.class` festgelegt werden. Der erste Teil des Dateinamens entspricht dem der entsprechenden `*.db` Datei.

5.2.5 Ausgabe

ACTS erzeugt sowohl Dateien für die Nutzung mit Verfahren des Maschinellen Lernens, mit denen in der zweiten Phase Hypothesen generiert werden sollen, als auch Dateien für GNUPLOT, mit denen die erzeugten Prototypen visualisiert werden. Des weiteren gibt es eine Datei mit statistischen Informationen über das durchgeführte Clustering.

Maschinelles Lernen

Die erzeugten Steuerdateien für Maschinelle Lernverfahren entsprechen zum einen dem von QUINLAN (1993) vorgeschlagenen Format. In diesem Fall sind in Dateien mit der Endung `*.names` die Anzahl und Namen der Attribute aufgezählt. Dateien mit der Endung `*.data` enthalten die symbolischen Attribute.

Zum anderen bietet ACTS die Möglichkeit, Dateien für die Verwendung mit dem Neuro-Fuzzy-System NEFCLASS zu erzeugen. Diese Dateien folgen der in HOFERICHTER (1996) veröffentlichten Konvention. Sowohl die Definition der Attribute wie ihre Ausprägung sind in einer Datei mit der Endung `*.dat` vermerkt.

Einige symbolische Lernverfahren können nicht auf Attributen mit kontinuierlichem Wertebereich arbeiten. Daher bietet ACTS die Möglichkeit, die statischen Attribute zu diskretisieren. Auch hierzu wird ein adaptives k-means Clustering durchgeführt, die verwendete Metrik ist dabei definiert als $|x - y|$. Dateien für NEFCLASS enthalten immer die Originalwerte.

Visualisierung

Für eine Überprüfung der generierten Hypothesen ist eine Visualisierung der erzeugten Prototypen notwendig. Dies geschieht bei ACTS auf zwei Wegen. Zum einen können die erzeugten Prototypen in Form einer durch GNUPLOT verarbeitbaren Datei ausgegeben werden. Zum anderen können ebenfalls zur Verarbeitung mit GNUPLOT bestimmte Dateien erzeugt werden, welche die zu einem Cluster gehörenden Zeitreihen übereinander plotten. Das fördert ein intuitives Verständnis der Bedeutung der verwendeten Metrik.

Die Dateien mit den Prototypen haben die Endung `*.lin.#1.#2` für lineare und `*.disc.#1.#2` für diskrete Attribute. `#1` steht für die Nummer des Attributes und `#2` für die Nummer des Prototypen entsprechend der `*.names`-Datei. Die Dateien mit den Zeitreihen haben die Endung `*.tr.disc.#1.#2` respektive `*.tr.lin.#1.#2`. Die Datei mit dem Namen `ExperimentProto5.tr.lin.0.1` enthält also die aus `ExperimentProto5.db` erzeugten Zeitreihen für das lineare Attribut Nr. 0, Prototyp 1.

Im Normalfall wird eine Benutzerin von ACTS diese Dateien nicht direkt verwenden, sondern mit Ansteuerungsdateien für GNUPLOT arbeiten, die vom System generiert worden sind. Dateien mit der Endung `*.a4.tr.gnuplot` erzeugen ein POSTSCRIPT-Dokument, bei dem für jeden Prototypen auf je einer A4-Seite alle zugehörigen Zeitreihen übereinander geplottet werden. `*.a4.pr.gnuplot` erzeugt eine entsprechende Datei mit den Prototypen.

Es ist auch möglich, Dateien im ENCAPSULATED POSTSCRIPT-Format zu erstellen. Diese Dateien haben an Stelle des `a4` ein `eps` im Namen. Hierbei wird für jeden Prototypen eine Datei mit der zusätzlichen Endung `.eps` erzeugt.

Statistik

ACTS gibt im Normallfall eine kurze Statistik auf dem Bildschirm aus, bei der für jedes Attribut die Anzahl der gebildeten Prototypen und die zahlenmäßige Verteilung der Zeitreihen auf die Cluster angezeigt wird. Dazu wird jeweils die Distortion angegeben.

In der Datei mit der Endung `*.stat` ist zusätzlich noch die Distortion jedes einzelnen Clusters angegeben. Werden statische Attribute diskretisiert, so finden sich hier die gefundenen Werteintervalle, deren zahlenmäßige Aufteilung und die Distortion. Weiterhin wird der gefundene Codebook-„Vektor“ (eindimensional) ausgegeben.

5.2.6 Realisierung

ACTS ist in C++ realisiert. Dabei wird auf der $\mathcal{MLC}$++-Bibliothek aufgesetzt. Das Klassenkonzept der Bibliothek ist in LONG (1994) beschrieben. Allerdings ist die Dokumentation gegenüber der Implementierung veraltet.

Die Nutzung von $\mathcal{MLC}$++ in ACTS betrifft vor allem die Auswertung von Umgebungsvariablen, das über die Umgebungsvariable LOGLEVEL in seiner Ausführlichkeit geregelte Ausgabeverhalten und die Behandlung von Fehlerzuständen. Auf die in $\mathcal{MLC}$++ realisierten Lernverfahren wird im subsymbolischen Verarbeitungsschritt nicht zurückgegriffen. Ferner sind Klassen aus der Arbeit von RADTKE (1998) in die Softwarearchitektur einbezogen, die den Zugriff auf unbearbeitete Zeitreihen erlauben.

Alle Beziehungen zwischen Klassen von ACTS sind als *has-a*-Relationen realisiert. Es gibt derzeit keine Vererbungshierarchien. Daher sind in der `Prototype`-Klasse sowohl diskrete als auch lineare Prototypen implementiert. Da nur wenige in dieser Klasse implementierte Methoden zwischen den beiden Arten unterscheiden müssen, habe ich auf Vererbung verzichtet. Ein Ausbau zu einer Architektur mit Vererbung an dieser Stelle sollte, sofern notwendig, bei späteren Erweiterungen allerdings leicht möglich sein.

Die verwendeten Klassen im Überblick (vgl. Abbildung 5.2):

1. `QuinlanDB`: Diese Klasse bildet den Kern von ACTS. Es gibt für jeden Programmlauf genau ein Objekt dieser Klasse. Es handelt sich um ein integriertes Datenbankobjekt, welches einerseits ein Objekt mit den unbearbeiteten Zeitreihen enthält, und andererseits sowohl Objekte mit Listen von Prototypen vorhält als auch Objekte, welche Informationen über die symbolische Repräsentation der Zeitreihen beinhalten.

   Es ist eine Methode vorhanden, die eine Datei mit „Rohdaten“ einliest und auf den Zeitreihen dann ein k-means Clustering durchführt. Es ist ebenfalls möglich, die statischen Attribute zu diskretisieren.

   Weiterhin existieren diverse Ausgabemethoden. In `QuinlanDB` sind Methoden für die Erzeugung der Steuerdateien für symbolische Lernverfahren, Statistiken, Steuerdateien für GNUPLOT sowie die Ausgabe von Clustern von Zeitreihen direkt realisiert. Die Ausgabe von Prototypen wird letztlich auf Objekte mit Listen von Prototypen übertragen.

2. `Classifier`: Auch von dieser Klasse existiert genau ein Objekt. Mit diesem Objekt werden die Klassifikationsverfahren realisiert. Dazu stellt es Methoden mit den implementierten Metriken zur Verfügung.

   Wird eine Zeitreihe mit einem Prototypen verglichen, so kann gleichzeitig die Distortion für die gefundene Zuordnung und eine Liste mit unscharfen Zugehörigkeiten zurückgeliefert werden.

   Damit die mathematischen Grundfunktionen für das k-means Clustering an einer Stelle zusammengefaßt und somit leichter abzuändern sind, stellt das `Classifier`-Objekt auch Methoden zur Diskretisierung der statischen Attribute sowie zum Simulated Annealing zur Verfügung.

   Das Objekt kann entweder mit einer Metrik für alle Attribute initialisiert werden, oder eine Datei mit der Angabe der Metriken für jedes Attribut einlesen.

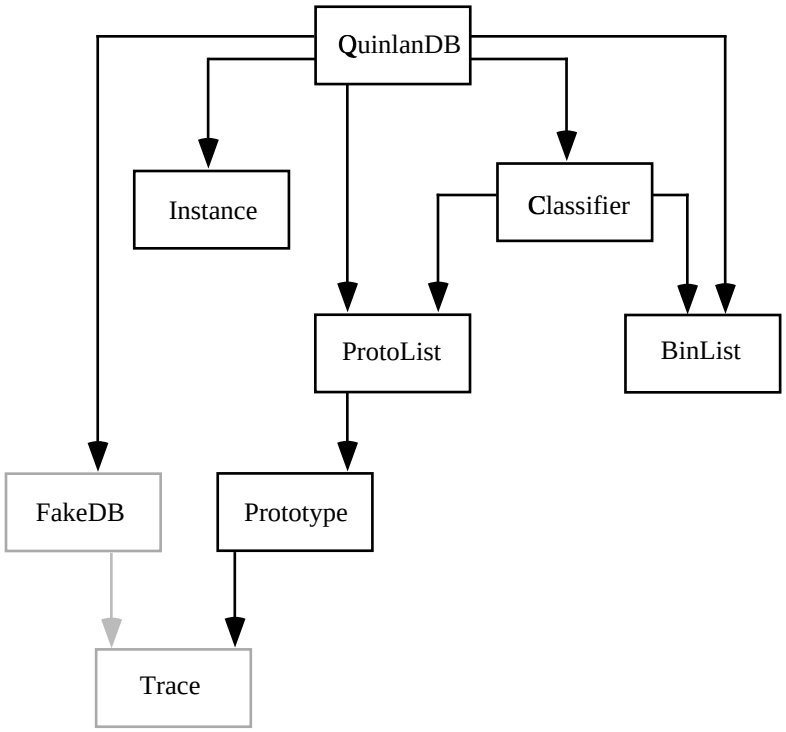

Abbildung 5.2: *Vereinfachtes Schema der Beziehungen der Klassen von* Acts. *Grau dargestellt sind Klassen aus der Arbeit von* Radtke *(1998).*

3. `ProtoList`: Diese Klasse stellt eine Liste von prototypischen Zeitreihen dar. Neben der Verwaltung der Prototypen wird die Ausgabe der Prototypen zu einem Attribut mit einer Methode gesteuert. Für jedes Attribut existiert nach Abschluß des Clusterings ein solches Objekt.

4. `Prototype`: In dieser Klasse findet sich die Realisierung einer prototypischen Zeitreihe. Neben den üblicherweise zu findenden Zugriffsfunktionen finden sich einige Methoden, die die Neuberechnung von Codebook-Vektoren erleichtern. Ferner sind Objekte dieser Klasse in der Lage, sich selber in eine Datei zu schreiben.

5. `Instance`: Hierbei handelt es sich um eine Klasse, deren Instanzen für die einzelnen Individuen die Klassenzugehörigkeit und eine Liste mit den symbolischen Attributwerten enthalten. Für jedes Individuum der Ausgangsdatenmenge existiert ein solches Objekt.

6. `AttList`: Diese Klasse realisiert eine Liste symbolischer Attributwerte mit den zugehörigen Zugriffsmethoden. Sie wird für das k-means Clustering zum Zwischenspeichern von Werten benutzt.

7. `BinList`: Diese Klasse realisiert Listen von Intervallen für die Diskretisierung von statischen Attributen. Das Objekt der Klasse `QuinlanDB` kann Instanzen dieser Klasse enthalten. In diesem Fall gibt es für jedes statische Attribut genau ein solches Objekt.

Das gesamte Verhalten des Systems wird durch das ACTS Hauptprogramm gesteuert. Hier werden die beiden zentralen Objekte der Klassen `Classifier` und `QuinlanDB` erzeugt. Ferner werden Umgebungsvariablen hauptsächlich hier ausgewertet. Sind diese nicht gesetzt, so werden entweder Default-Werte angenommen, oder es wird interaktiv beim Benutzer nachgefragt.[4] Das Setzen der entsprechenden Parameter erfolgt dementsprechend ebenfalls hier.

5.3 Zweite Phase

In der mit symbolischen resp. Neuro-Fuzzy Verfahren arbeitenden zweiten Phase wird aus den so erzeugten Abbildungen der Zeitreihen auf symbolische Repräsentationen die eigentliche Hypothese generiert.

5.3.1 Symbolische Verfahren

Die vorgestellte $\mathcal{MLC}$++-Bibliothek enthält Implementierungen mehrerer Verfahren des Maschinellen Lernens, die auf dem von ACTS generierten Dateityp arbeiten können. Des weiteren können mit Hilfe der in KOHAVI UND SOMMERFIELD (1996) beschriebenen $\mathcal{MLC}$++-Utilities Dateien für weitere Systeme generiert werden, so daß ein weites Spektrum an induktiven Lernverfahren für die Weiterverarbeitung zur Verfügung steht.

Die vollständige Integration von subsymbolischer und symbolischer Komponente in ein Gesamtsystem wäre möglich und wurde testweise für das ID3-Verfahren auch realisiert. In der derzeitigen Phase, in der verschiedene Modelle des Maschinellen Lernens auf ihre Tauglichkeit für das vorliegende Problem getestet werden sollen, halte ich eine derart enge Einbindung allerdings nicht für wünschenswert.

Stattdessen wird die Phase der Hypothesenbildung über das zu $\mathcal{MLC}$++ gehörende Programm INDUCER realisiert. INDUCER ist ein ebenfalls vollständig über Umgebungsvariablen oder interaktiv steuerbares System, das den Zugriff auf die verschiedenen in der Bibliothek implementierten Lernverfahren ermöglicht.

Darüber hinaus ist es möglich, mit weiteren, sogenannten externen Lernverfahren direkt zu arbeiten. Diese werden teilweise von $\mathcal{MLC}$++ zur Verfügung gestellt. Dabei handelt es sich vorwiegend um Verfahren, die im akademischen Bereich frei verfügbar sind. Einige der unterstützten Systeme sind allerdings nur in kommerziellen Varianten erhältlich.

[4] Durch die Umgebungsvariable PROMPTLEVEL wird gesteuert, ob alle Parameter interaktiv abgefragt werden, oder ob evtl. vorhandene Default-Werte ohne Rückfrage übernommen werden sollen (s.a. Anhang D.2).

Die im Verlauf der vorliegenden Arbeit hauptsächlich benutzten Algorithmen sind (vgl. Kapitel 3):

1. In $\mathcal{MLC}$++ implementiert:
   - ID3
   - MC4
   - HOODG
   - OPTIONDT
2. In $\mathcal{MLC}$++ integriert:
   - CN2
3. Externe Verfahren:
   - RIPPER

Die Einbindung dieser Verfahren über das Programm INDUCER hat weitere Vorteile. So ist für Verfahren, die keine Arbeit auf kontinuierlichen Wertebereichen ermöglichen, eine Diskretisierung möglich. Ferner sind Verfahren zur *Feature Subset Selection* integriert, also der Auswahl einzelner Attribute, wie in JOHN *et al.* (1994) beschrieben. Bisher wurden diese Möglichkeiten allerdings nicht systematisch getestet. Eine Dokumentation von INDUCER findet sich in KOHAVI UND SOMMERFIELD (1996). Für Dokumentationen zu den einzelnen Verfahren verweise ich auf Kapitel 3.

5.3.2 Neuro-Fuzzy Verfahren

Die Integration eines Neuro-Fuzzy Klassifikationsverfahrens wird über eine nach UNIX portierte Variante des NEFCLASS-Systems realisiert (siehe Unterkapitel 3.5.1). Das Tool XNFC ist eine TCL/TK basierte Graphische Oberfläche zu NEFCLASS.

Eine Beschreibung der Software findet sich in HOFERICHTER (1997). Durch die GUI ist das Programm allerdings relativ intuitiv bedienbar, wenn man mit den dahinterliegenden Konzepten vertraut ist.

In XNFC wird das Training des Neuronalen Netzes sehr schön als Modifikation der Zugehörigkeitsfunktionen visualisiert. Das System arbeitet in der installierten Version nicht sehr stabil, das sollte allerdings keinen prinzipiellen Einfluß auf die durchzuführenden Tests haben.

Kapitel 6

Metriken

*In entgegengesetztem Sinne wie ähnlich gebraucht man unähnlich (*Aristoteles, *Metaphysik, S. 1018a).*

6.1 Einleitung

In diesem Kapitel werden die in Acts implementierten Metriken vorgestellt. Diese sind für das k-means Clustering (vgl. Unterkapitel 4.3) von entscheidender Bedeutung: anhand der gewählten Metrik wird die Partitionierung gebildet. In jedem Cluster sind die einzelnen Zeitreihen zu finden, die bzgl. eines *Ähnlichkeitsmaßes* oder eines *Distanzmaßes* die geringsten Unterschiede aufweisen.

6.1.1 Distanz vs. Ähnlichkeit

Es lassen sich Ähnlichkeits- und Distanzmaße unterscheiden. Beide Formen lassen sich ineinander überführen, sind also von ihrer Ausdruckskraft her betrachtet identisch. Nach Kaufman und Rousseeuw (1990) und Rummel (1997) gelten die folgenden Anforderungen:

Ähnlichkeitsmaße

Ähnlichkeitsmaße sind zumeist Abbildungen in das Einheitsintervall $[0 \dots 1]$. Ein höherer Wert der Abbildung impliziert eine größere Ähnlichkeit, das Ähnlichkeitsmaß zweier gleicher Zeitreihen muß daher den Wert 1 haben.

Seien $s(X, Y)$ eine Ähnlichkeitsmetrik, X, Y Zeitreihen. Dann gilt:

Reflexivität: $s(X, X) = 1$

Symmetrie: $s(X, Y) = s(Y, X)$

Identität: $s(X, Y) = 1 \Leftrightarrow X = Y$

Distanzmaße

Distanzmaße verringern sich mit zunehmender Ähnlichkeit der betrachteten Zeitreihen. Im allgemeinen sind sie nicht negativ, bilden also in das Intervall $[0 \ldots \infty]$ ab.

Seien $d(X, Y)$ eine Distanzmetrik, X, Y, Z Zeitreihen. Dann gilt:

Reflexivität:	$d(X, X) = 0$
Symmetrie:	$d(X, Y) = d(Y, X)$
Identität:	$d(X, Y) = 0 \Leftrightarrow X = Y$
Dreiecksungleichung:	$d(X, Y) + d(Y, Z) \geq d(X, Z)$

6.1.2 Allgemein

Die Identitätsbedingung ist aus unserer Sicht insofern problematisch, als daß die Zeitreihen nur Attributausprägungen an Individuen sind. Damit darf aber aus der Identität der Zeitreihen nicht auf die Identität der Individuen geschlossen werden.

Distanzmetriken sind im allgemeinen weder in Richtung der Zeit noch der Meßgröße translationsinvariant. Bei der Meßgröße wird eine Normierung vorausgesetzt. Die Translationsvarianz in Richtung der Zeit ist dahingegen problematisch, vgl. hierzu Unterkapitel 8.2.1.

Die Wahl der Metrik kann nicht unabhängig von der Art des betrachteten Attributes geschehen. Die Semantik der Metrik ist von entscheidender Bedeutung. Unter Semantik verstehe ich in diesem Zusammenhang, welche phänomenologischen Aspekte von einer Metrik widergespiegelt werden.

Auch wenn ich im folgenden nur eine Auswahl gebräuchlicher Metriken vorstelle, kann im Rahmen von ACTS jedes Maß implementiert werden, das aus Sicht der Epidemiologie interessant sein könnte. Die implementierten Metriken sind entnommen aus KAUFMAN UND ROUSSEEUW (1990) und RUMMEL (1997).

Im folgenden seien X, Y, Z Zeitreihen mit Werten zu jeweils l diskreten Zeitpunkten. Diese Annahme gilt wegen der Interpolierbarkeit der Werte ohne Beschränkung der Allgemeinheit und reflektiert nur die Anzahl diskreter Zeitpunkte, zu denen die Metrik berechnet wird.

Ferner seien x, y einzelne Werte aus X, Y. Werte zu einem bestimmten Zeitpunkt $i, 1 \leq i \leq l$ aus X, Y, Z seien bezeichnet mit x_i, y_i, z_i.

6.2 Hamming

Die erste hier vorgestellte Distanzmetrik ist eine Modifikation der Hamming-Distanz. Sie ist ursprünglich nur sinnvoll anwendbar auf Attribute mit einer endlichen Menge als Wertebereich. Denkbar wäre z.B., daß in einer derartigen Zeitreihe die Frage „Rauchen Sie?“ wiedergegeben wird.

Bei der Hamming-Distanz wird die Anzahl der Übereinstimmungen aufsummiert. Ich habe die Metrik etwas modifiziert, so daß jetzt jeweils Werte, die nur um ein ϵ_h voneinander differieren, als gleich angesehen werden. Damit lassen sich z.B. bei der Frage „Wieviele Zigaretten rauchen Sie pro Tag?“ Antworten, die nur um eine gewisse Menge voneinander abweichen, als gleich bewerten.

Die allgemeine Formulierung wird mit $\epsilon_h = 0$ subsumiert.

Es gilt also:

$$Hamming(X, Y) = \sum_{i=1}^{l} f(x_i, y_i) \tag{6.1}$$

mit

$$f(x, y) = \begin{cases} 0 & \text{falls } |x - y| \leq \epsilon_h, \\ 1 & \text{sonst.} \end{cases} \tag{6.2}$$

Die modifizierte Hamming-Distanz wird in ACTS mit Hamming bezeichnet. Die Umgebungsvariable ACTS_HAMMING_EPSILON entspricht ϵ_h. Sie hat einen Default-Wert von 0.1.

6.3 L_p-Norm

Bei der L_p-Norm handelt es sich um eine Distanzmetrik, welche die Unterschiede der Werte der Zeitreihen zu unterschiedlichen Zeitpunkten betrachtet. Der Wert von $p, p \in \mathbb{N}$ gewichtet, wie stark der Einfluß von „Ausreißern“ ist. Je größer p ist, um so stärker ist der Einfluß von stark differierenden Werten.

Es gilt also:

$$L_p(X, Y) = \sqrt[p]{\sum_{i=1}^{l} (x_i - y_i)^p} \tag{6.3}$$

Derzeit ansprechbar sind nur die L_1 und die L_2-Norm mit den Bezeichnungen LP1 und LP2. Die Metrik ist allerdings verallgemeinert implementiert, so daß mit geringen Änderungen an ACTS die allgemeine Fassung ansprechbar wäre.

6.4 L_1-Norm

Hierbei handelt es sich um eine effizientere Implementierung eines Spezialfalles.

$$L_p(X,Y) = \sum_{i=1}^{l} |x_i - y_i| \tag{6.4}$$

Aufgerufen wird diese Metrik mit LP1.

6.5 Differenzmetriken

Bei Differenzmetriken werden die Zeitreihen zuerst auf einen skalaren Wert abgebildet. Der Absolutwert der Differenz dieser Skalare ist dann die gesuchte Differenz der beiden Zeitreihen. Eine einfache Abbildung in diesem Sinne wären z.B. die Mittelwert- oder Medianbildung über der einzelnen Zeitreihe.

Die allgemeine Fassung einer Differenzmetrik lautet:

$$d(X,Y) = |\, Trans(X) - Trans(Y)| \tag{6.5}$$

6.5.1 Mittelwert

Der schon angesprochene Fall: zuerst wird über beide Zeitreihen wie in Gleichung 6.6 beschrieben der Mittelwert gebildet.

$$\overline{Z} = \frac{1}{l} \sum_{i=1}^{l} z_i \tag{6.6}$$

Dann werden die beiden Mittelwerte gemäß 6.7 subtrahiert und der Absolutwert gebildet.

$$MeanDiff\,(X,Y) = |\overline{X} - \overline{Y}| \tag{6.7}$$

Diese Metrik wird über Mean angesprochen.

6.5.2 Trend

Die Trendanalyse soll zwei Zeitreihen dann als ähnlich charakterisieren, falls ihre Werte sich jeweils in der gleichen Richtung entwickeln. In Gleichung 6.8 wird die skalare Abbildung der Zeitreihe vorgenommen. Dazu wird ähnlich wie bei Bildung der Kovarianz (vgl. Gleichung 6.10) die Abhängigkeit der beiden zugrunde gelegten Zeitreihen in Beziehung gesetzt zur Abweichung der betrachteten Zeitreihe von ihrem Mittelwert.

$$Trend(X,Y) = \frac{1}{\sum_{i=1}^{l}(x_i - \overline{X})^2} * \sum_{i=1}^{l}(x_i - \overline{X})(y_i - \overline{Y}) \tag{6.8}$$

Jetzt bleibt als Aufgabe nur noch, die beiden so gewonnen Werte zu subtrahieren und den Betrag zu bilden.

$$TrendDiff\,(X,Y) = |\,Trend(X,Y) - Trend(Y,X)| \tag{6.9}$$

Diese Metrik ist als Trend in ACTS abrufbar.

6.6 Kreuz-Korrelation

Bei der Kreuz-Korrelation werden lineare Abhängigkeiten zweier Zeitreihen betrachtet. Dazu geht die Kovarianz in die Berechnung dieser Metrik ein.

Die Kovarianz (Gleichung 6.10) erfaßt nach LEISER (1983) das gemeinsame Abweichungsverhalten zweier Zeitreihen. Der Grundgedanke ist folgender: hängen die beiden Zeitreihen X und Y nicht voneinander ab, sind sie also statistisch unabhängig, so kann aus einer Abweichung der einen Zeitreihe von ihrem Mittelwert nichts für die Abweichung der anderen Zeitreihe von ihrem Mittelwert ausgesagt werden. Die Kovarianz von X und Y ist dann 0. Umgekehrt umgekehrt.

Hängen die Zeitreihen stark voneinander ab, so wird eine bestimmte Abweichung der einen Zeitreihe von einer bestimmten Abweichung der anderen begleitet. Dabei kann eine positive Abweichung von X mit einer positiven Abweichung von Y einhergehen. Man spricht dann von positiv kovarianten Zeitreihen mit $Cov(X,Y) > 0$. Oder: positive Abweichungen von X gehen mit negativen von Y einher (negative Kovarianz, $Cov(X,Y) < 0$).

$$Cov(X,Y) = \frac{1}{l}\sum_{i=1}^{l}(x_i - \overline{X})(y_i - \overline{Y}) \tag{6.10}$$

In die Berechnung der Kreuz-Korrelation geht zusätzlich noch die Standardabweichung σ ein, die wie folgt definiert ist:

$$\sigma_X = \sqrt{\frac{1}{l}\sum_{i=1}^{l}(x_i - \overline{X})^2} \tag{6.11}$$

Damit läßt sich die gesuchte Metrik schreiben als:

$$CrossCorr\,(X,Y) = \frac{Cov(X,Y)}{\sigma_X \sigma_Y} \tag{6.12}$$

Der resultierende Wert liegt innerhalb des Intervalls $[-1 \ldots 1]$. Eine Kreuz-Korrelation von 1 sagt aus, daß X eine lineare Funktion von Y ist, während für den Wert von -1 X ein

Spiegelbild von Y ist. Ein Wert von 0 bedeutet, daß die beiden Zeitreihen linear unabhängig voneinander sind.

Dieses Ähnlichkeitsmaß wird angesprochen mit X-Corr. In der Implementierung habe ich noch eine Absolutwertbildung durchgeführt. Angesichts der normalisiert vorliegenden Zeitreihen sollte dies kein Problem darstellen.

Kapitel 7

Beispiele

Grau, teurer Freund, ist alle Theorie, Und grün des Lebens goldner Baum. Mephistoles, in: GOETHE *(1808).*

7.1 Einleitung

Zur ingenieurwissenschaftlichen Bearbeitung des vorliegenden Themas gehört zusätzlich zur Realisierung in Form von ACTS auch eine zumindest kurze Demonstration der Anwendung. Diese soll in diesem Kapitel vorgenommen werden. Dabei kann von einer umfassenden systematischen Untersuchung nicht die Rede sein, dies würde den Rahmen der vorliegenden Arbeit sprengen.

Zu beachten ist ferner, daß das System bisher nur mit stark eingeschränkten artifiziellen Datensätzen gearbeitet hat. Sobald Realdatensätze vorliegen, muß ein Test mit ihnen folgen.

Ich habe für diese kurze Demonstration des Einsatzes des vorgeschlagenen zweistufigen Verfahrens einen Datensatz *P5* ausgewählt, in dem eine Kausalbeziehung kodiert ist, und einen Datensatz *N*, der nur aus Rauschen besteht.

Es sind die statistischen Daten aus den ACTS-Läufen angegeben. Dazu sind jeweils Prototypen und Cluster für das relevante Attribut linear_1 abgebildet. Die maximal erlaubte Anzahl von Clustern war jeweils auf sechs gesetzt. Es ist gut zu sehen, daß dieser Wert häufig unterschritten wird

In der zweiten Phase kommen die Verfahren ID3, MC4, CN2 RIPPER und HOODG zum Einsatz. Die veröffentlichten Resultate zu OPTIONDT sind recht vielversprechend. Allerdings führt sein Einsatz in meinen Untersuchungen schon bei den vorliegenden kleinen, artifiziellen Problembereichen zu stark „aufgeblähten“ Bäumen. Es wurde praktisch an jedem Knoten für jedes mögliche Attribut ein Optionsknoten erzeugt. Ich habe daher aus Platzgründen auf die Darstellung verzichtet.

Zu den Verfahren der zweiten Phase ist anzumerken, daß sowohl ID3 als auch MC4 ohne Pruning liefen. Ansonsten wurden die Standardeinstellungen der implementierten Systeme benutzt.

Die erzeugten Hypothesen habe ich, soweit dies sinnvoll erschien, dokumentiert. Für NEFCLASS habe ich auf den Abdruck verzichtet, da ein direkter Vergleich mit den anderen Verfahren aufgrund der verwendeten Umkodierung nur schwerlich möglich gewesen wäre.

7.2 Datensatz mit Kausalbeziehung

Die in diesem Unterkapitel folgenden Tests wurden auf einem Datensatz durchgeführt, in dem es einen kausalen Zusammenhang zwischen dem Aussehen der Zeitreihen zum linearen Attribut Nr. 1 und der Klassifizierung gibt. Ich habe einen Test mit Kreuz-Korrelation und einen mit L_2-Norm als Metrik dokumentiert.

7.2.1 Kreuz-Korrelation

Allgemein

Eine statistische Übersicht über die unter der Kreuz-Korrelation auf dem P5-Datensatz gefundenen Prototypen findet sich in Abbildung 7.1. Die Prototypen und übereinander geplotteten Zeitreihen für das erste lineare Attribut sind in den Bildern 7.2 bis 7.5 dokumentiert.

ID3/MC4

Sowohl ID3 als auch MC4 fanden jeweils einen Entscheidungsbaum mit fünf Knoten (vgl. Abbildung 7.6). Die im linearen Attribut linear_1 liegende Kausalität wurde entdeckt.

HOODG

Der HOODG-Algorithmus erzeugt durch die Identifikation der beiden mit „ill“ beschrifteten Knoten der Bäume einen Graphen mit vier Knoten (vgl. Abbildung 7.7).

CN2

Der Regelerzeuger CN2 erzeugt neben der Default-Regel, welche die mehrheitlich vertretene Klasse ausgibt, drei Regeln. Das Attribut „linear_1“ hat hier die Bezeichnung „attr_1 “. Sehr gut zu sehen ist, daß die beiden Prototypen 0 und 1 bereits eine sehr saubere Trennung der Klassen ermöglicht.

```
IF    attr_1 = X1
THEN  class = ill  [25 0]

IF    attr_4 < 0.45
THEN  class = ill  [19 0]
```

```
Statistical Summary for Experiment ExperimentProto5 with:
Classes: ill, healthy
Number of linear attributes: 2
Number of discrete attributes: 2
Number of static attributes: 2

The attributes in detail:

linear_0:
With metrics X-Corr  has 2 prototypes
with overall distortion 7.40234878291
 Proto 0 representing [92] instances
 Proto 1 representing [8] instances

linear_1:
With metrics X-Corr has 2 prototypes
with overall distortion 5.3502112275
 Proto 0 representing [75] instances
 Proto 1 representing [25] instances

discrete_0:
With metrics X-Corr  has 1 prototypes
with overall distortion 25.9261382
 Proto 0 representing [100] instances

discrete_1:
With metrics X-Corr  has 1 prototypes
with overall distortion 29.6479246
 Proto 0 representing [100] instances
```

Abbildung 7.1: *Statistische Übersicht über die von* ACTS *auf dem P5-Datensatz mit der Kreuz-Korralation als Metrik generierten Prototypen (gekürzt).*

```
IF    attr_1 = X0
AND attr_4 > 0.45
THEN  class = healthy  [0 56]

(DEFAULT) class = healthy  [44 56]
```

RIPPER

RIPPER findet eine fast identische Regelmenge für die kranken Fälle. Da er eine geordnete Regelmenge ausgibt, produziert RIPPER allerdings außer der Default-Regel keine weitere Regel

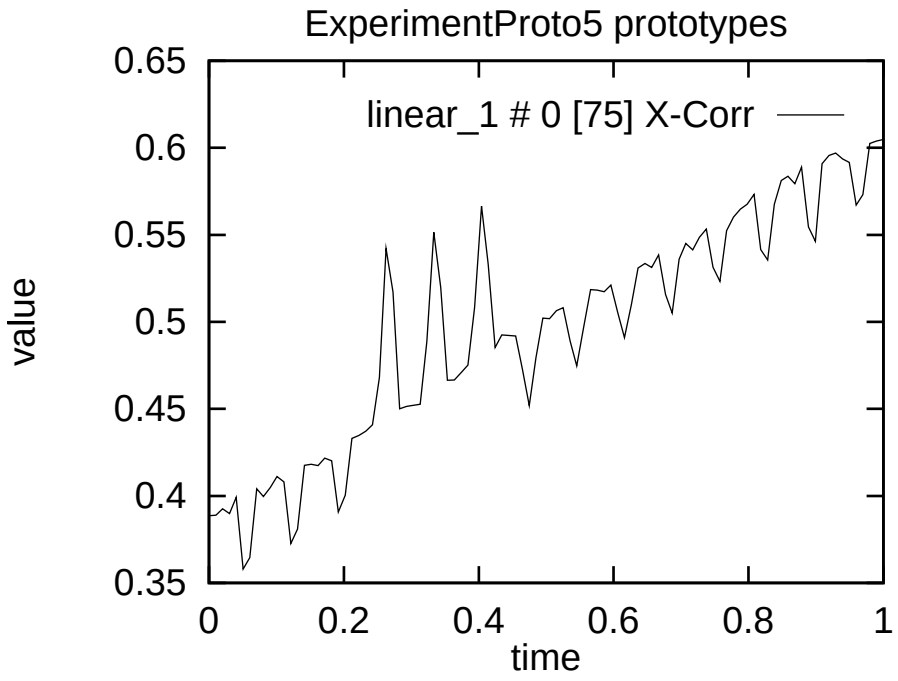

Abbildung 7.2: *X-Corr, Prototyp 0, Attribut linear_1 für P5.*

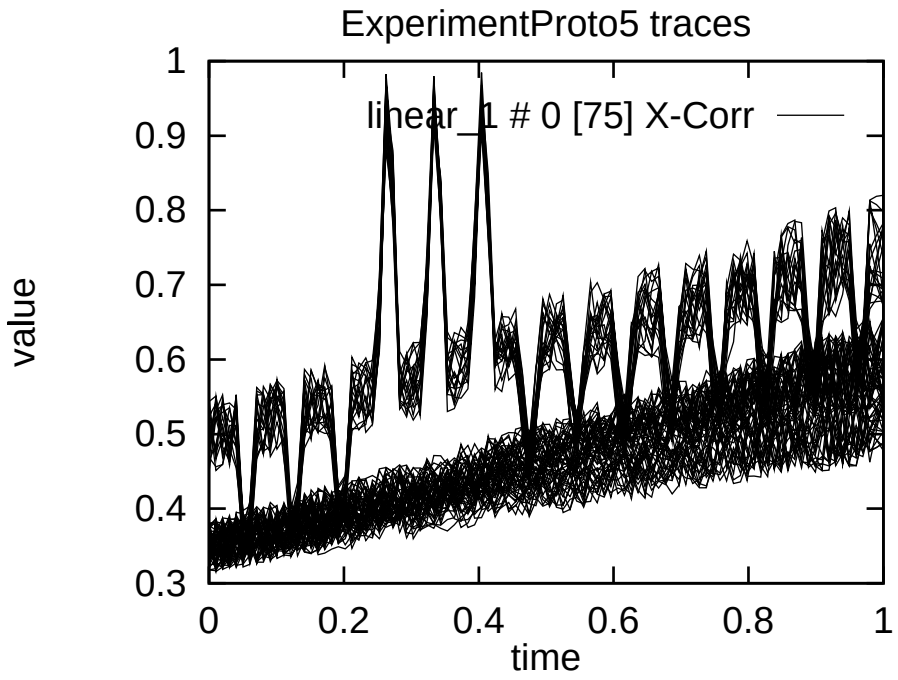

Abbildung 7.3: *X-Corr, Zeitreihenschar 0, Attribut linear_1 für P5.*

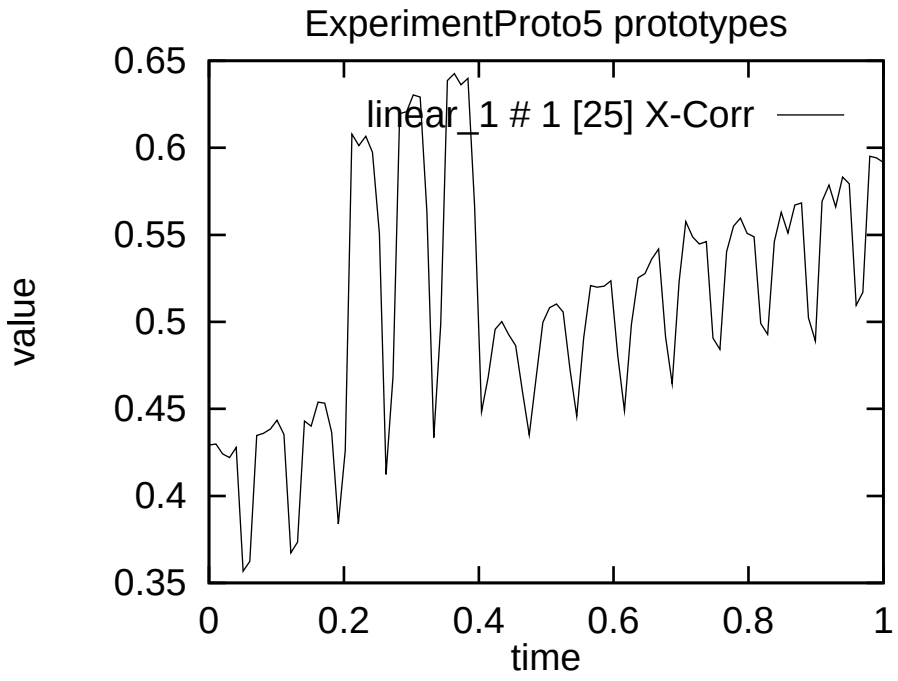

Abbildung 7.4: *X-Corr, Prototyp 1, Attribut linear_1 für P5.*

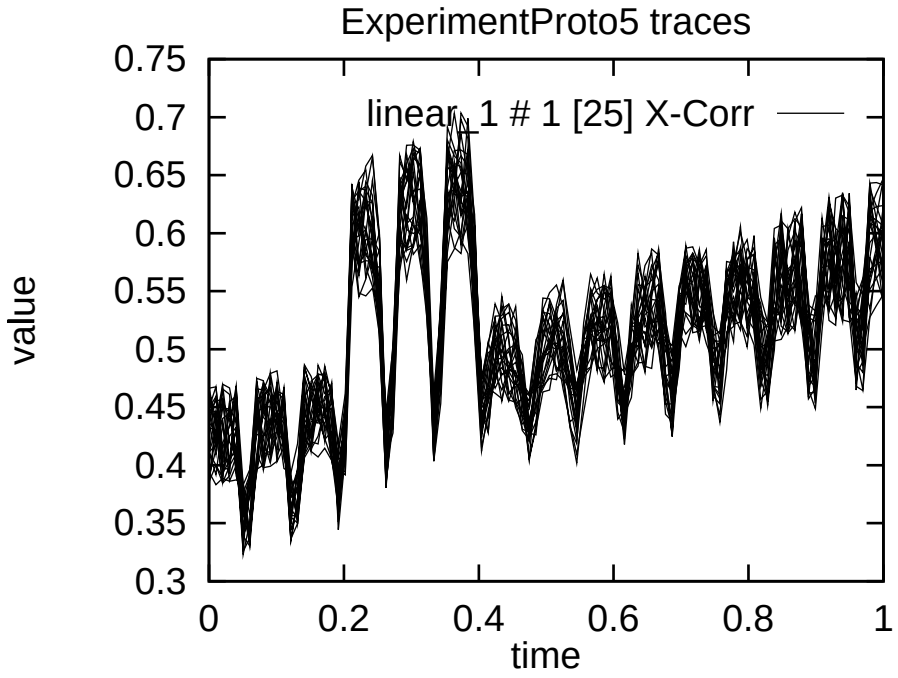

Abbildung 7.5: *X-Corr, Zeitreihenschar 1, Attribut linear_1 für P5.*

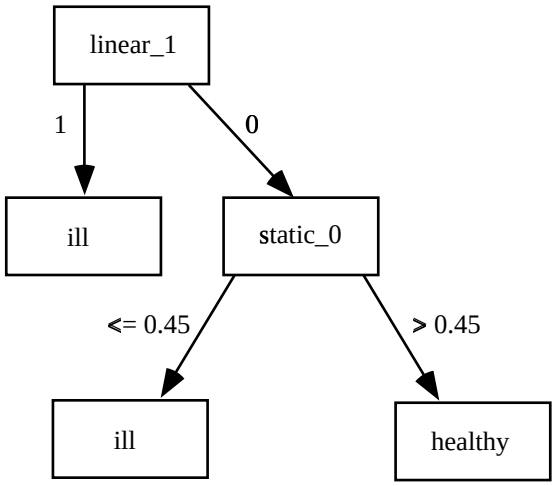

Abbildung 7.6: *Entscheidungsbaum von ID3 und MC4 zum Datensatz P5 mit Kreuz-Korrelation.*

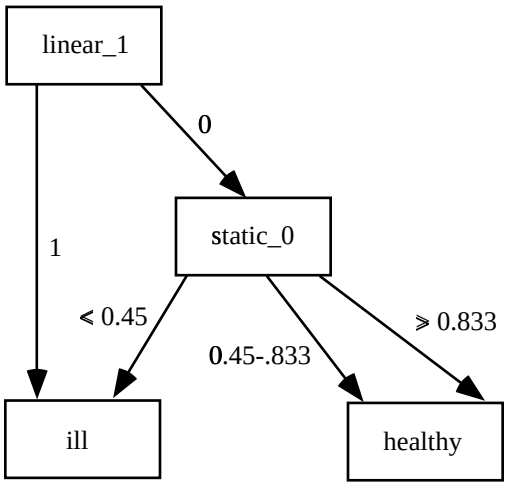

Abbildung 7.7: *Entscheidungsgraph von* HOODG *zum Datensatz P5 mit Kreuz-Korrelation.*

für die gesunden.

```
ill 25 0 IF linear_1 = _1 .
ill 19 0 IF static_0 <= 0.4 .
healthy 56 0 IF .
```

NEFCLASS

Das Neuro-Fuzzy System NEFCLASS findet 14 Regeln, die direkt aus der Eingabemenge ableitbar sind. Das Training des Netzes bricht mit der ersten Iteration ab, d.h., daß keine kleinere

Regelmenge erzeugt werden kann.

7.2.2 L_2-Norm

Allgemein

Die Prototypen und übereinander geplotteten Zeitreihen für das erste lineare Attribut sind den Abbildungen 7.9 bis 7.14 zu finden. Ein Abdruck der generierten statistischen Informationen findet sich in Abbildung 7.8.

ID3/MC4

Die beiden verwandten Algorithmen ID3 und MC4 erzeugen wieder identische Ausgaben (s. Abbildung 7.15), diesmal mit sogar nur drei Knoten.

HOODG

Hoodg stellt diesmal eine ganz andere Hypothese auf als die baumbasierten Verfahren. Die von ihm aufgezeichnete Vermutung (vgl. Abbildung 7.16) betrifft nur das diskrete Attribut Nr. 1. Eine Auswertung durch Epidemiologen müßte hier entscheiden, welche der generierten Hypothesen plausibler ist.

CN2

Die von CN2 gefundene Hypothese deckt sich mit den Hypothesen, die von den baumbasierten Verfahren ID3 und MC4 aufgestellt worden sind:

```
IF    attr_1 = X1
THEN  class = ill  [25 0]

IF    attr_1 = X2
THEN  class = ill  [19 0]

IF    attr_1 = X0
THEN  class = healthy  [0 56]

(DEFAULT) class = healthy  [44 56]
```

RIPPER

Auch hier generiert Ripper eine kleinere Hypothese als CN2. Außerdem werden wieder nur Hypothesen für eine der beiden Klassen aufgestellt.

```
Statistical Summary for Experiment ExperimentProto5 with:
Classes: ill, healthy
Number of linear attributes: 2
Number of discrete attributes: 2
Number of static attributes: 2

The attributes in detail:

linear_0:
With metrics LP2 has 2 prototypes
with overall distortion 0.654902920943
 Proto 0 representing [34] instances
 Proto 1 representing [66] instances

linear_1:
With metrics LP2 has 3 prototypes
with overall distortion 0.303808170562
 Proto 0 representing [56] instances
 Proto 1 representing [25] instances
 Proto 2 representing [19] instances

discrete_0:
With metrics LP2 has 6 prototypes
with overall distortion 0.215514053574
 Proto 0 representing [22] instances
 Proto 1 representing [15] instances
 Proto 2 representing [19] instances
 Proto 3 representing [6] instances
 Proto 4 representing [19] instances
 Proto 5 representing [19] instances

discrete_1:
With metrics LP2 has 6 prototypes
with overall distortion 0.202918765456
 Proto 0 representing [22] instances
 Proto 1 representing [15] instances
 Proto 2 representing [19] instances
 Proto 3 representing [17] instances
 Proto 4 representing [19] instances
 Proto 5 representing [8] instances
```

Abbildung 7.8: *Statistische Übersicht über die von* ACTS *auf dem P5-Datensatz mit der L_2-Norm als Metrik generierten Prototypen (gekürzt).*

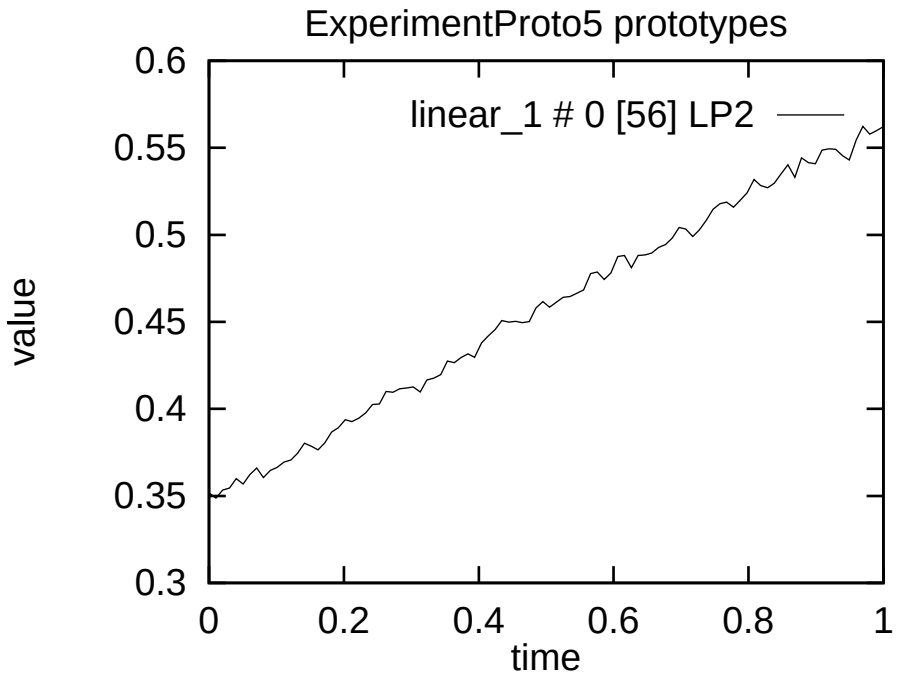

Abbildung 7.9: L_2, *Prototyp 0, Attribut linear_1 für P5.*

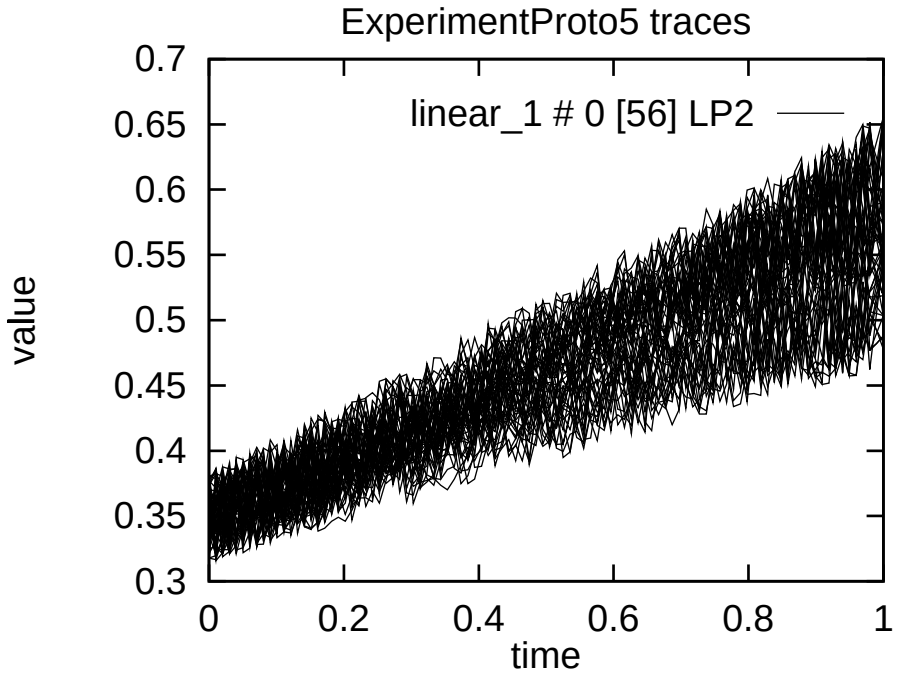

Abbildung 7.10: L_2, *Zeitreihenschar 0, Attribut linear_1 für P5.*

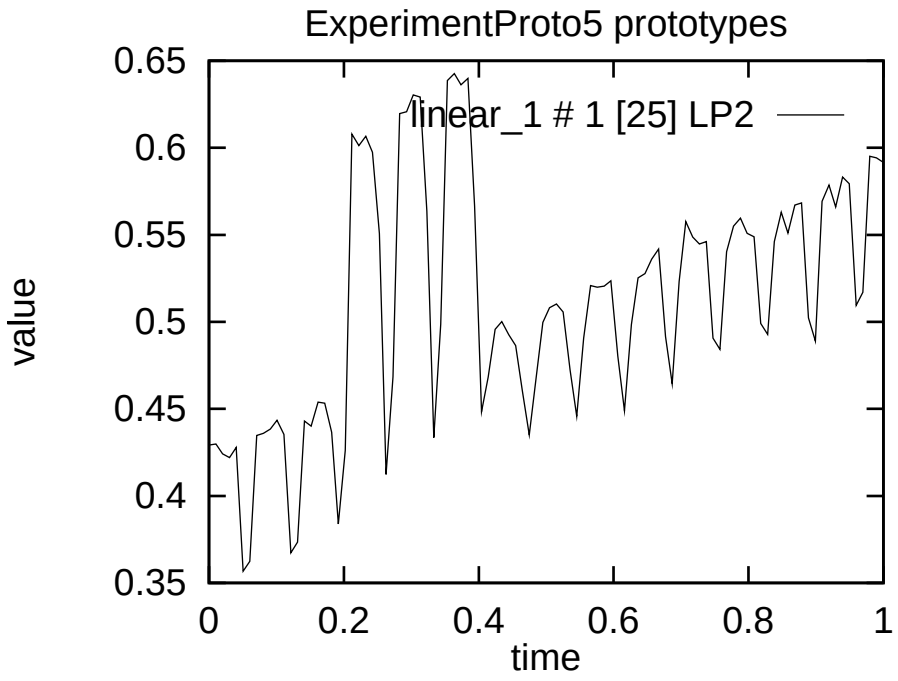

Abbildung 7.11: *L_2, Prototyp 1, Attribut linear_1 für P5.*

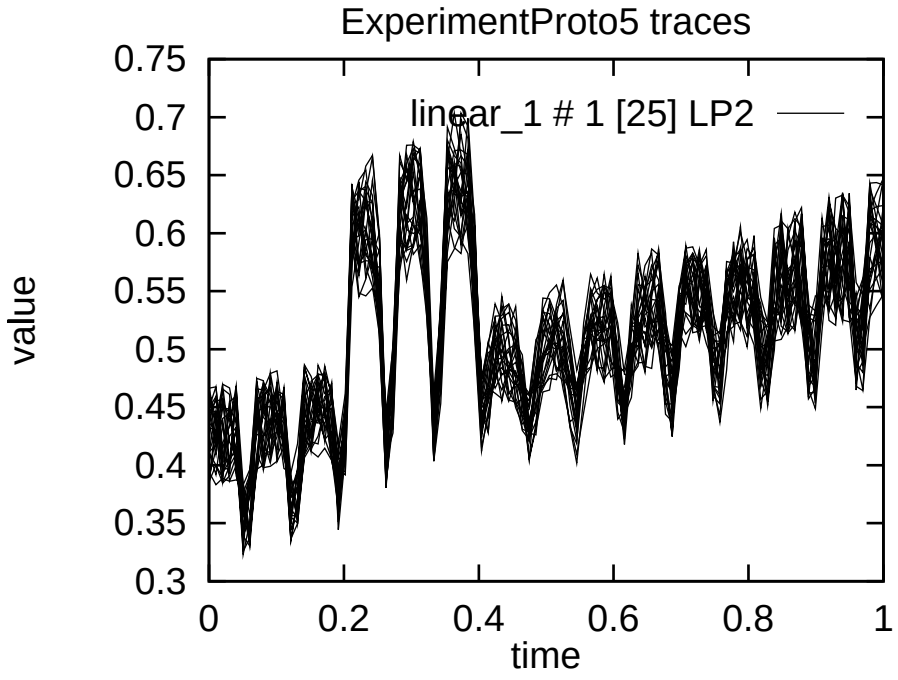

Abbildung 7.12: *L_2, Zeitreihenschar 1, Attribut linear_1 für P5.*

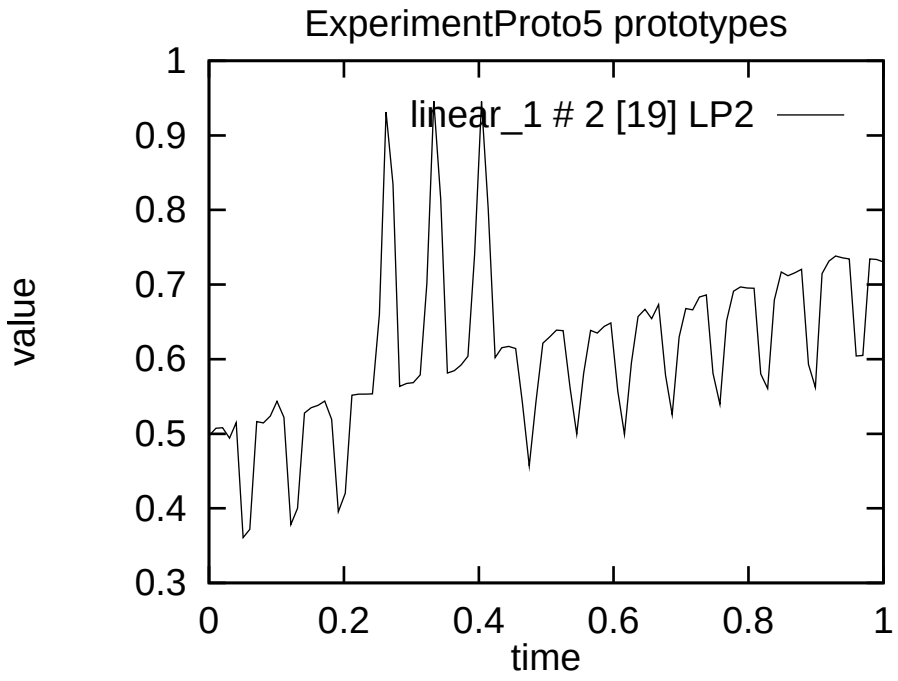

Abbildung 7.13: L_2, *Prototyp 2, Attribut linear_1 für P5.*

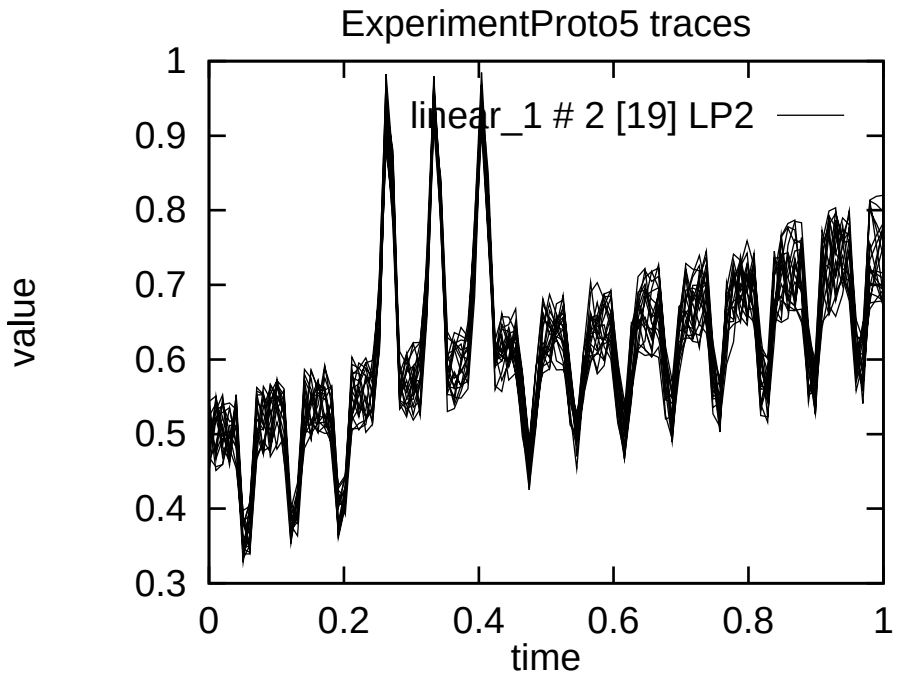

Abbildung 7.14: L_2, *Zeitreihenschar 2, Attribut linear_1 für P5.*

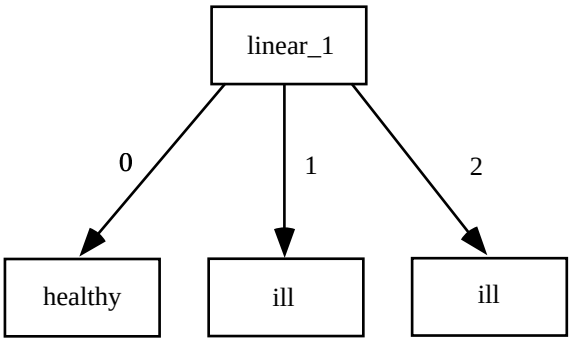

Abbildung 7.15: *Entscheidungsbaum von ID3 und MC4 zum Datensatz P5 mit L_2-Norm.*

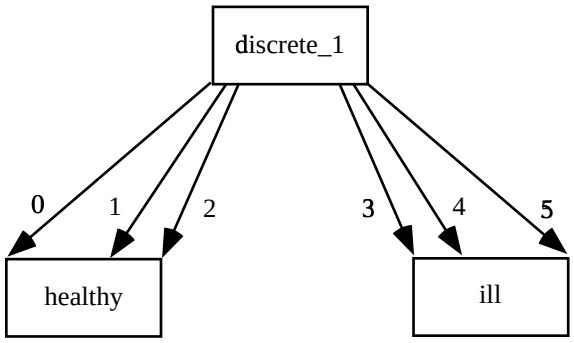

Abbildung 7.16: *Entscheidungsgraph von* HOODG *zum Datensatz P5 mit L_2-Norm.*

```
ill 25 0 IF linear_1 = _1 .
ill 19 0 IF linear_1 = _2 .
healthy 56 0 IF .
```

NEFCLASS

Diesmal findet NEFCLASS nur sieben Regeln. Aber auch hier konvergiert das Regelwerk nicht weiter, der Lernvorgang bricht mit der ersten Iteration ab.

7.3 Datensatz mit Rauschen

In diesem Unterkapitel habe ich ACTS mit einem Datensatz arbeiten lassen, der nur Rauschen enthält. Da das k-means Clustering garantiert terminiert, wurde auch hier eine Partitionierung gefunden. Dabei wurde unter der hier alleine dokumentierten L_2-Norm sogar die minimale Anzahl der Cluster gefunden.

Der Grund wird darin liegen, daß alle gefunden Partitionierungen relativ große Intervalle auf einen Prototypen abbilden. Damit dürfte die Distortion bei allen Partitionierungen so groß

```
Statistical Summary for Experiment ExperimentNoise with:
Classes: ill, healthy
Number of linear attributes: 2
Number of discrete attributes: 2
Number of static attributes: 2

The attributes in detail:

linear_0:
With metrics LP2 has 2 prototypes
with overall distortion 2.30788096219
 Proto 0 representing [40] instances
 Proto 1 representing [60] instances

linear_1:
With metrics LP2 has 2 prototypes
with overall distortion 2.31372786983
 Proto 0 representing [40] instances
 Proto 1 representing [60] instances

discrete_0:
With metrics LP2 has 2 prototypes
with overall distortion 2.79682635475
 Proto 0 representing [60] instances
 Proto 1 representing [40] instances

discrete_1:
With metrics LP2 has 2 prototypes
with overall distortion 2.80288710198
 Proto 0 representing [69] instances
 Proto 1 representing [31] instances
```

Abbildung 7.17: *Statistische Übersicht über die von* Acts *auf dem N-Datensatz mit der L_2-Norm als Metrik generierten Prototypen (gekürzt).*

sein, daß größere Werte von k wegen des durchgeführten Simulated Annealing kein gutes Ergebnis erreichen können. Die Graphiken zeigen in der Tat, daß die gefundenen Cluster eine große Spannweite von Zeitreihen enthalten.

Allgemein

Die Prototypen und übereinander geplotteten Zeitreihen für das erste lineare Attribut finden sich in den Bildern 7.18 bis 7.21. Die statistischen Informationen zu den gefundenen Partitionen sind in Abbildung 7.17 dokumentiert.

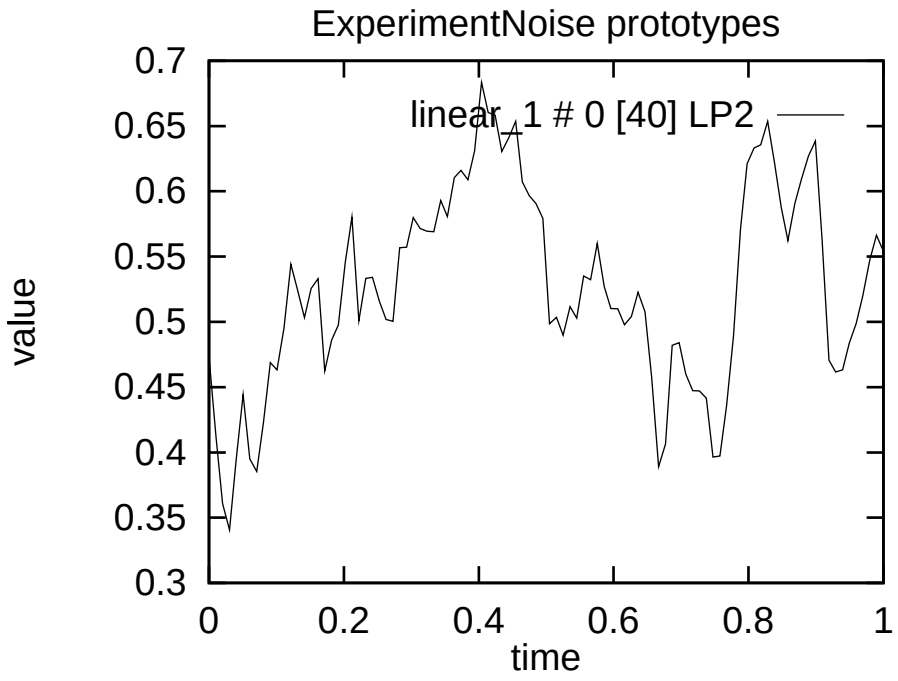

Abbildung 7.18: L_2, *Prototyp 0, Attribut linear_1 für N.*

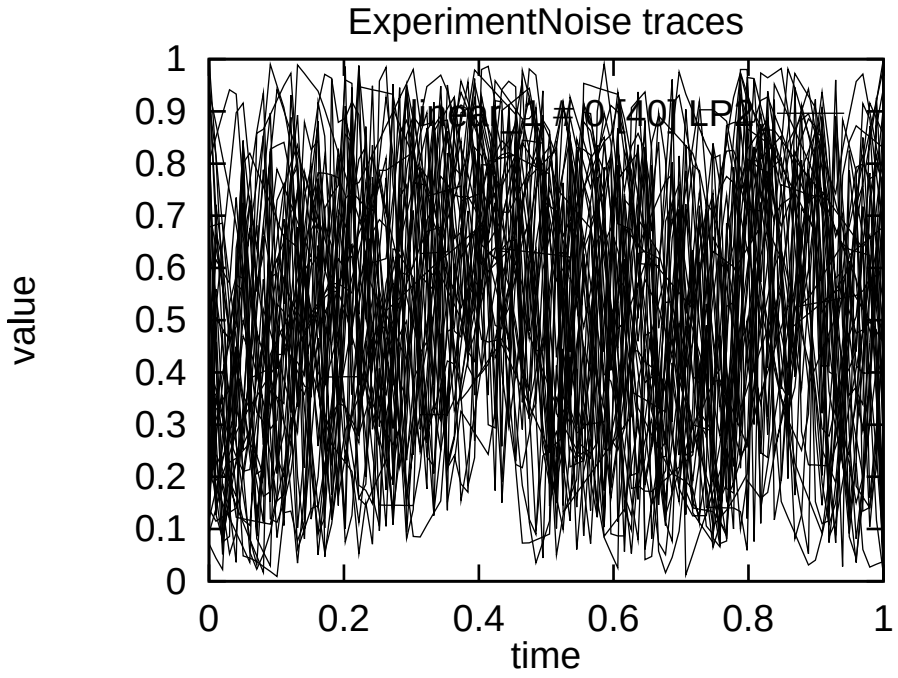

Abbildung 7.19: L_2, *Zeitreihenschar 0, Attribut linear_1 für N.*

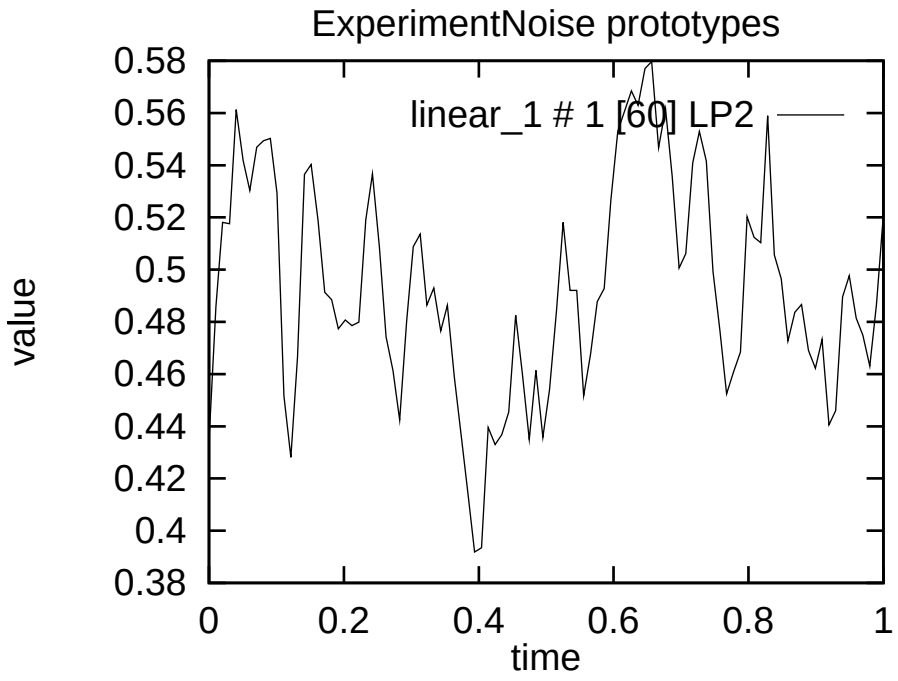

Abbildung 7.20: L_2, *Prototyp 1, Attribut linear_1 für N.*

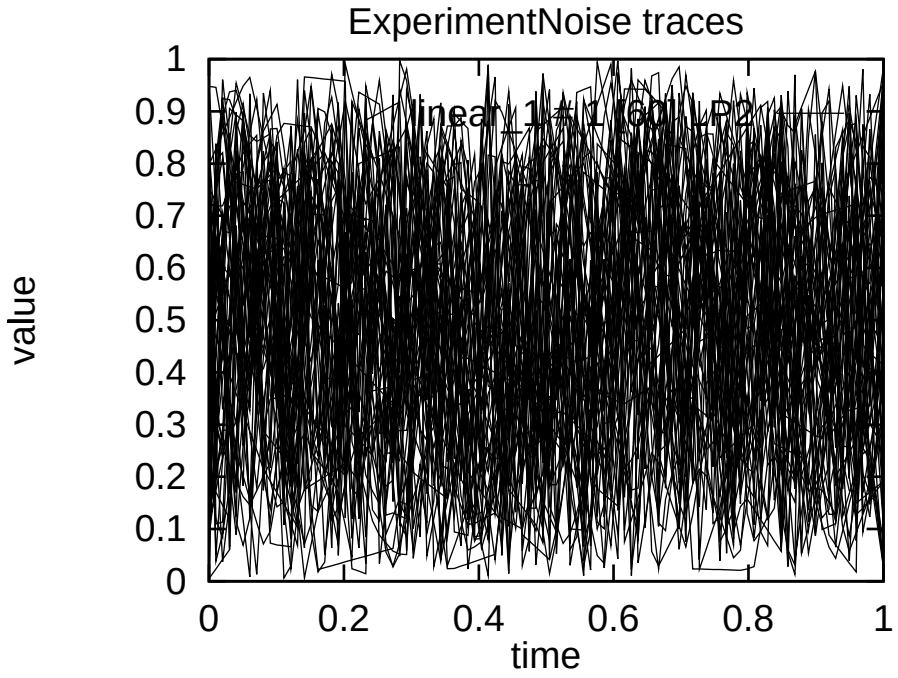

Abbildung 7.21: L_2, *Zeitreihenschar 1, Attribut linear_1 für N.*

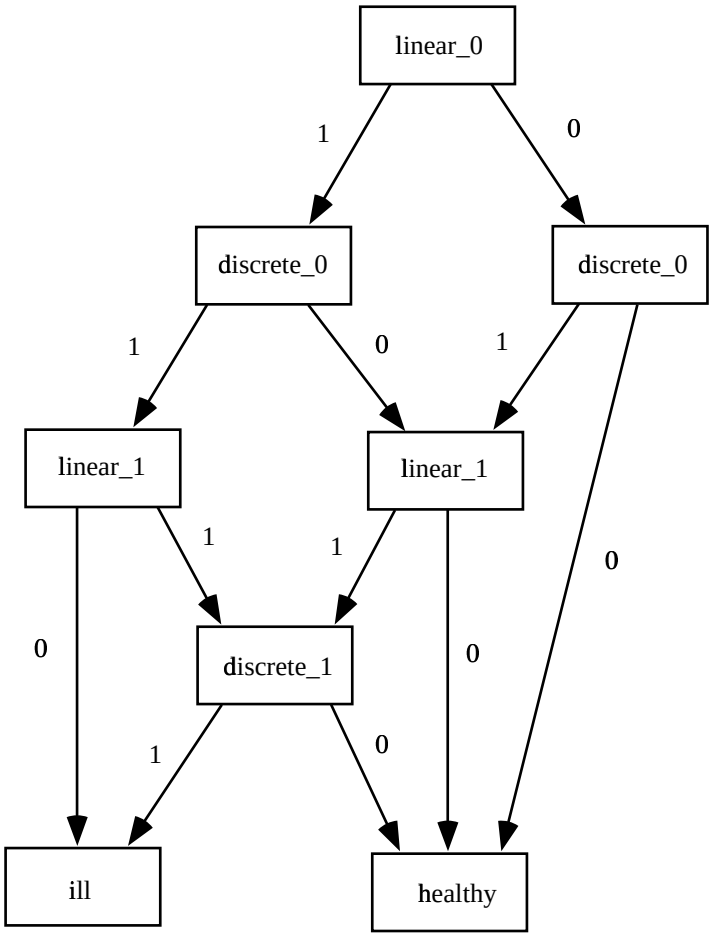

Abbildung 7.22: *Entscheidungsgraph von* HOODG *zum Datensatz N mit L_2-Norm.*

ID3/MC4

Der ID3-Algorithmus fand einen Baum mit 71 Knoten. Der Baum von MC4 hatte 69 Knoten. Beide Algorithmen klassifizierten letztlich fast jede Instanz der Eingabemenge mittels eines eigenen Knotens.

HOODG

Erstaunlicherweise fand HOODG dagegen eine Hypothese, die mit nur acht Knoten auskommt (vgl. 7.22). Die Kontrolle der Plausibilität dürfte angesichts der zugrunde liegenden Zeitreihen allerdings eher negativ ausfallen.

CN2

CN2 findet 28 Regeln und zusätzlich die Default-Regel. Jeder der Regeln deckt höchsten elf Beispiele ab, zumeist aber nicht mehr als vier oder fünf.

RIPPER

RIPPER fand wie HOODG eine kurze Hypothese, die aber wohl auch keine epidemiologische Plausibilität für sich beanspruchen dürfte.

```
ill 19 5 IF static_1 >= 0.462 discrete_0 = _1 .
ill 13 10 IF static_1 <= 0.267 .
healthy 37 16 IF .
```

NEFCLASS

NEFCLASS findet 99 Regeln. Das Verfahren konvergiert nicht auf eine kleinere Anzahl von Regeln.

7.4 Resumee

Die prinzipielle Anwendbarkeit des vorgeschlagenen Verfahrens wurde an einigen wenigen Beispielen skizziert. Auch wenn die hier dokumentierten Test keine Allgemeingültigkeit beanspruchen dürfen, so drängen sich doch einige Feststellungen auf.

Die beiden eng verwandten Verfahren ID3 und MC4 kamen auf den Testdatensätzen zu den gleichen Ergebnissen. Auch wenn dies nicht zwangsläufig so sein muß, so könnten die Unterscheide zwischen dem C4.5-Derivat MC4 und ID3 bei einem Verzicht auf Pruning geringer ausfallen als bei Standard-Anwendungen im Data Mining Bereich. Schließlich wurde C4.5 gegenüber ID3 vor allem im Hinblick auf seine Klassifikationleistung verbessert, während der Fokus der vorliegenden Arbeit auf den generierten Hypothesen liegt.

Bei den beiden regelerzeugenden Verfahren fällt auf, daß die geordnete Liste von RIPPER bereits bei den einfachen vorliegenden Fällen schwerer verständlich ist als die Regelmenge von CN2. Das völlige Fehlen von Hypothesen für eine der beiden Klassen fällt sofort ins Auge.

HOODG erzielt teilweise stark von den baumbasierten Algorithmen abweichende Ergebnisse. Vor allem sein Ergebnis mit der L_2-Norm sollte genau analysiert werden.

NEFCLASS scheint mit den gelieferten Datensätzen nicht sinnvoll arbeiten zu können. Inwieweit dies ein Problem schlecht gewählter Zugehörigkeitsfunktionen ist, sollte näher untersucht werden. Möglicherweise ist die gewählte lokale Kodierung der Daten nicht problemadäquat.

Bei der Arbeit mit dem Datensatz, der nur aus Rauschen besteht, ist ein Punkt hervorzuheben: zumindest die von den baumbasierten Verfahren generierten Hypothesen können sehr

schnell als trivial erkannt werden, denn mit ihnen werden praktisch nur Einzelfälle klassifiziert. Auch CN2 und NEFCLASS haben im Verhältnis zu HOODG und RIPPER große Regelmengen vorzuweisen.

Kapitel 8

Ausblick

*Ins Freie! Die letzten Worte können sich kaum noch als Text verstehen. Es ist vorbei! Hinaus! (*HIMMELREICH*, 1995, S. 120).*

8.1 Einleitung

Dieses letzte Kapitel dient der kurzen Reflexion des Erreichten. Diese Reflexion soll aber nicht rückwärtsgewandt sein, sondern sowohl in den positiven wie in den negativen Aspekten auf zukünftige Möglichkeiten verweisen. „Was kann noch getan werden?“ ist eine Frage, die genauso wichtig ist wie die Frage: „Habe ich mein Ziel erreicht?“

Ich benenne im ersten Unterkapitel einige Unzulänglichkeiten des vorgestellten Systems. Dabei versuche ich jeweils, mögliche Lösungswege vorzustellen und dabei auftretende Problembereiche zu skizzieren.

Zuletzt ziehe ich ein kurzes Fazit dieser Arbeit. Dabei nehme ich im besonderen Bezug auf das eingangs vorgestellte Ziel der Arbeit und ihre Einbettung in ein Gesamtsystem zur Unterstützung der epidemiologischen Forschung.

8.2 Weitere Arbeitsgebiete

Im folgenden weise ich auf Problembereiche des prototypisch implementierten Systems hin und diskutiere Möglichkeiten der Weiterentwicklung.

8.2.1 Translationsinvarianz

Einer der größten Nachteile des vorgestellten Systems ist sicherlich, daß die Clusterung der Zeitreihen nicht translationsinvariant bzgl. der Zeit ist. Dadurch werden beispielsweise zwei Zeitreihen, die beide einen charakteristischen Peak aufweisen, allerdings zu unterschiedlichen Zeitpunkten, nicht notwendig einem Cluster zugeordnet, auch wenn dies aus epidemiologischer Sicht evtl. wünschenswert wäre.

Verschiebung

Eine von BLUM (1982) im RADIX/RX-System benutzte und auch von RUMMEL (1997) für ADSEQ diskutierte Möglichkeit ist, die zu vergleichende Zeitreihe sukzessive um einen bestimmten Zeitabschnitt zu verschieben. Seien X, Y, Z Zeitreihen und $shift_t(Z)$ die um die Zeit t mit $t \in [0 \dots 1]$ verschobene Zeitreihe Z. Dann gilt für eine Ähnlichkeitsmetrik $s(X, Y)$ und eine Distanzmetrik $d(X, Y)$:

$$s_{shift}(X, Y) = \max_{t \in [0 \dots 1]} s(X, shift_t(Y))$$

respektive

$$d_{shift}(X, Y) = \min_{t \in [0 \dots 1]} d(X, shift_t(Y))$$

Die Ähnlichkeit einer Zeitreihe und eines Codebook-Vektors wäre dabei bestimmt als ihre Ähnlichkeit zum Zeitpunkt ihrer besten Übereinstimmung.

Bei einem solchen Vorgehen lassen sich allerdings folgende Probleme identifizieren:

1. **Grenzen:** Sowohl der Codebook-Vektor als auch jede andere Zeitreihe haben Werte für Zeiten t zwischen $t = 0$ und $t = 1$. Bei der Verschiebung einer Zeitreihe muß daher eine sinnvolle Handhabung der „überlappenden“ Bereiche gefunden werden.
2. **Brute Force:** Es handelt sich um ein nicht wissengesteuertes sog. „Brute Force“-Vorgehen. Dies setzt der Anwendbarkeit auf große Datenmengen Grenzen. Es muß dabei beachtet werden, daß die Berechnung der Ähnlichkeit nach jeder Modifikation des Codebook-Vektors erneut erfolgen muß. Dies geschieht zusätzlich noch für jeweils mehrere Werte von k.
3. **Abhängigkeiten:** Es kann nicht davon ausgegangen werden, daß die einzelnen Attribute eines Individuums voneinander unabhängig sind. Um epidemiologisch aussagekräftige Ergebnisse zu bekommen, müßte die Ermittlung der größten Ähnlichkeit gleichzeitig über alle Zeitreihen erfolgen. Dies wirft gleich zwei Probleme auf:
   (a) Ein globales Gütekriterium muß definiert werden, und
   (b) es kommt zu einer Komplexitätsexplosion.

Aus diesen Gründen halte ich dieses Vorgehen für problematisch, falls keine Heuristiken für die Verschiebung gefunden werden können.

Fourier-Transformation

Näher zu überprüfen wäre hingegen sicherlich, ob eine Clusterung von Fourier-transformierten Zeitreihen möglich wäre. Durch die Transformation in den Bildraum ist es unter Umständen möglich, zeitinvariante Ähnlichkeitsbeziehungen zu definieren.

Aufgrund der Kürze der Zeit war es mir nicht möglich, diese Variante zu untersuchen.

8.2.2 Ordnung auf den Clustern

In der ersten Phase verzichte ich auf die Nutzung der Informationen über die Klassenzugehörigkeit der Instanzen. Die Partitionierung der Menge der Zeitreihen wird mit einem nichtüberwachten Lernverfahren durchgeführt. Ich möchte dadurch ausschließen, Artefakte in die Clusterung einzubauen, die vom Lernverfahren der zweiten Phase trivialerweise „entdeckt“ werden.

Eine von mir nicht untersuchte Möglichkeit wäre, nach der Clusterung eine Ordnung resp. Halbordnung auf der so gefundenen Partitionierung zu definieren. Dies könnte z.B. über das Verhältnis der Klassenzugehörigkeiten in einem Cluster definiert sein. Cluster, in denen mit hoher Sicherheit Zeitreihen kranker Individuen zu finden sind, hätten einen höheren Ordnungswert als solche, die in der Hauptsache aus Zeitreihen gesunder Individuen bestehen, oder umgekehrt.

Für die Weiterverwendung mit einem symbolischen Verfahren dürfte dies nicht so interessant sein, wenngleich C4.5 (vgl. Unterkapitel 3.2.1) durchaus in der Lage ist, auf geordneten Wertebereichen Tests auf Untermengen zu realisieren.

Ein größeres Potential sehe ich allerdings beim Einsatz eines Neuro-Fuzzy Systems wie Nefclass (vgl. Unterkapitel 3.5.1). Der Einsatz einer lokalen Codierung lieferte hier nicht das gewünschte Ergebnis. Nefclass fand in den durchgeführten Tests nur die direkt aus der Eingabemenge ableitbaren Regeln und generierte keine neuen. Es wäre aber zu untersuchen, ob Nefclass mit Zugehörigkeitsfunktionen, die aus der Ordnung auf den Clustern begründet sind, ein besseres Verhalten zeigt.

8.2.3 Auswahl der Metriken

Derzeit müssen die Metriken, die bei der Prototyp-Bildung eingesetzt werden, dem System vorgegeben werden. Dies kann zwar der Integration von Hintergrundwissen dienen, es wäre aber wünschenswert, daß zusätzlich eine automatische Auswahl passender Metriken geschehen kann. Dabei taucht aber die Frage auf, ob sich über die Metriken ein Gütekriterium definieren läßt.

Ein Problem betrifft die schon erwähnten prinzipiellen Unterschiede von Ähnlichkeits- und Distanzmetriken (vgl. Unterkapitel 6.1.1). Eine Transformation von Ähnlichkeits- in Distanzmetriken ist für eine automatische Auswahl unabdingbar.

Die einzelnen Metriken haben darüber hinaus unterschiedliche Wertebereiche. Daher bedarf es zusätzlich einer Normierung.

Auch wenn damit mathematisch eine Vergleichbarkeit hergestellt ist, muß die Frage erlaubt sein, ob nicht trotzdem „Äpfel und Birnen“ verglichen werden. Daß eine Klassifizierung der Zeitreihen mit einer bestimmten Metrik bessere Ähnlichkeiten zu den Prototypen erzeugt, ist zunächst kein epidemiologisch begründetes Auswahlkriterium.

Eine automatische Auswahl von Metriken kann meines Erachtens nur über einen wissensgesteuerten Ansatz geschehen. Dazu müßten zwei Voraussetzungen geschaffen werden:

1. **Typisierung:** Die einzelnen Attribute werden verschiedenen Kategorien zugeordnet. Diese Kategorien sollen spezifische Eigenschaften der mit diesem Attribut beobachteten Entitäten reflektieren. So ist z.B. die Exposition gegenüber einem Giftstoff unterscheidbar von der Anzahl Kuchen, die das beobachtete Individuum täglich verspeist.

2. **Wissensbasis:** Das System erhält die Möglichkeit, in einer angeschlossenen Wissensbasis „nachzuschlagen", welche Arten von Metriken üblicherweise für die entsprechende Kategorie verwandt werden. Es könnte dann z.B. für jede dieser Möglichkeiten Hypothesen aufstellen, die dann von den menschlichen Expertinnen auf ihre Plausibilität hin untersucht werden könnten.

Für dieses Vorgehen wäre vorab allerdings eine eingehende Untersuchung der benutzten Metriken notwendig.

8.2.4 Minimalisierung

Wie bereits häufiger angesprochen, sind Verfahren des Maschinellen Lernens darauf ausgelegt, möglichst minimale Hypothesen zu generieren. Diese sind für die Klassifikation ungesehener Objekte durchaus geeignet. Von WEISS UND KULIKOWSKI (1991) wird die Klassifikationsleistung auf ungesehene Exemplare als Mittel der Wahl zur Messung der Leistung von Induktionslernverfahren vorgeschlagen.

Im Regelfall wird der symbolische Verarbeitungsschritt also abbrechen, sobald mit einem Test auf ein Attribut in einem Knoten respektive in einer Regel eine Aufteilung der Eingabemenge in Partitionen erfolgt, die jeweils (mehrheitlich) einer Klasse angehören. Dabei können in solchen Partitionen versteckte kausale Zusammenhänge übersehen werden.

Eine Möglichkeit, diese Schwäche zu überwinden, läge evtl. in folgendem Vorgehen:[1]

1. Erzeuge mit einem beliebigen Verfahren des Maschinellen Lernens eine Hypothese.

2. Entferne das Attribut, welches den größten Informationsgewinn bringt, aus der Eingabemenge.

3. Iteriere das Verfahren, bis nur noch triviale Hypothesen generiert werden, also solche, die jeden Einzelfall oder gar keinen Fall klassifizieren.

Auf den Lösungsraum bezogen bedeutet dies, daß die jeweils nächste Lösung orthogonal zur Achse des letzten entfernten Attributes gesucht wird.

Auch hier hatte ich leider nicht die Möglichkeit, die Anwendbarkeit des Verfahrens und die theoretische Fundierung zu untersuchen.

[1] Der Biologe Armin Rose machte mich auf diese Möglichkeit aufmerksam.

8.3 Fazit

Von der Ausgangsfrage her betrachtet, ist diese Arbeit dem Ziel einer Unterstützung der epidemiologischen Forschung in Form der automatischen Generierung von Hypothesen nur ein kleines Stück näher gekommen.

Es bleibt aber festzuhalten, daß trotz der im letzten Unterkapitel diskutierten Schwierigkeiten die prinzipielle Anwendbarkeit von Methoden des Maschinellen Lernens auf den vorliegenden Gegenstandsbereich[2] gezeigt wurde.

Gleichzeitig wurde gezeigt, daß mit der Kombination von Verfahren des symbolischen Lernens und der subsymbolischen Mustererkennung Leistungen erbracht werden können, zu denen die beteiligten Teile einzeln nicht in der Lage gewesen wären.

Obwohl das Laufzeitverhalten zum Beispiel bei der Kombination ACTS und C4.5 im Bereich $O(n^2)$ liegt, machen erste Versuche auf noch relativ kleinen Datenbeständen Mut. Die Anwendung des Systems sollte nicht schon am Laufzeitverhalten scheitern.

Das Framework, in welches sich das in dieser Arbeit vorgestellte System einbettet, ermöglicht es der empirischen Forschung, Hypothesen unter kontrollierten Laborbedingungen zu testen, wie in den exakten Naturwissenschaften vorausgesetzt. Dies sollte eine bessere Fundierung über statistische Evidenz hinaus ermöglichen.

Insofern denke ich, der eingangs zitierten Forderung von Reinhold Haux zumindest nicht widersprochen zu haben: das Gesamtsystem, in das sich diese Arbeit einbettet, hat das Potential, ein Werkzeug zu sein, das die epidemiologische Grundlagenforschung erleichtert.

ENDE des fachlichen Teils.

[2] Um diese Formulierung ein letztes Mal zu gebrauchen.

Kapitel 9

Danke

> *Da jeder neuen Erkenntnis das Potential des schon akkumulierten allgemeinen Wissens zugrunde liegt, ist ein davon abtrennbarer und allein dem besonderen Individuum zuzurechnender und dann gar in besondere Gratifikationen umzusetzender Verdienst nicht bestimmbar (*Ruschig*, 1995, S. 10).*

Schon da Wissenschaft ein Gattungsunternehmen ist, sind mehr Menschen am Gelingen dieser Arbeit beteiligt, als ich hier aufzählen könnte. Bei einigen, die unmittelbar am Entstehungsprozeß beteiligt waren, möchte ich mich hier aber explizit bedanken.

So bei meinem Betreuer Frank Köster, der auch zu Zeiten Vertrauen in die Arbeit hatte, in denen ich mir selber nicht so sicher war. Auch wenn wir nicht immer einer Meinung waren, so waren die Diskussionen fruchtbar.

Korrektur gelesen haben Harald Schütt, Bernd Kramer und Thomas Koddenberg. Die Fehler, die sie übersehen haben, gehen natürlich auf mein Konto.

Carsten Labinsky hat zu einer Zeit höchster Panik während der Implementierung bei einer Tasse Kaffee vor allem Ruhe ausgestrahlt.

Ohne meinen Mentor in der Philosophie, Ulli Ruschig, wäre diese Arbeit vielleicht nie geschrieben worden.

Allen Mitarbeiterinnen und Mitarbeitern der Abteilung Sonnenschein gilt mein Dank für die freundliche Aufnahme. Mir ist klar, daß dieses „Dankeschön“ in Form von Kuchen expliziert werden muß.

Ein ganz spezieller Dank geht an Marita Olsen dafür, daß sie es so lange mit mir ausgehalten und mir immer wieder Motivation gegeben hat. Dette var hele tida veldig viktig for meg.

Diese Arbeit ist meinen Eltern gewidmet.

Teil III

Anhang

Anhang A

Werkzeuge

Die Implementierung von ACTS erfolgte mit dem C++-Compiler GCC Version 2.7.2.3 unter Solaris 5.5.1 auf einer Sun Ultrasparc.

In der Implementierung stütze ich mich auf der $\mathcal{MLC}$++-Bibliothek[1] Version 2.01 ab. Diese Bibliothek enthält Implementierungen von u.a. ID3, der C4.5-Variante MC4 und HOODG und stellt Interfaces zu weiteren benutzten Algorithmen, wie CN2 und RIPPER, zur Verfügung.

Die Anwendung der Algorithmen erfolgt über die $\mathcal{MLC}$++-Utilities[2] in der Version 2.1. $\mathcal{MLC}$++ basiert selber wiederum auf der LEDA-Bibliothek[3] in der Version 3.5.1.

ACTS unterstützt das FuzzyNN-Tool NEFCLASS Version 1.0 mit TCL/TK basierter GUI. TCL/TK habe ich in der Version 8.0 installiert.

Baum- oder graphenorientierte Induktionsalgorithmen werden mit Hilfe von DOTTY[4] Version 1.3 visualisiert. Die Aufbereitung der Zeitreihen erfolgt mit GNUPLOTVersion 3.5.

Diese Arbeit wurde mit den Editoren ALPHA auf einem Apple Macintosh und GNU EMACS auf einer Sun Ultrasparc geschrieben und mit den LaTeX 2_ε-Implementationen OzTeX und TeTeXgesetzt. Das Literaturverzeichnis wurde mit BibTeX erstellt.

Die meisten Graphiken wurden mit SHAREDRAW auf einem Apple Macintosh erstellt.

[1] KOHAVI *et al.* (1997)
[2] KOHAVI UND SOMMERFIELD (1996)
[3] MEHLHORN *et al.* (1998)
[4] KOUTSOFIOS UND NORTH (1996), KOUTSOFIOS UND NORTH (ohne Datum)

Anhang B

Implementierung

B.1 Sourcen ACTS

Die kompletten Sourcen für das ACTS-System befinden sich auf dem Rechner `streisand.offis.uni-oldenburg.` im Verzeichnis `/export/home/cassens/ src/da`. Es handelt sich um folgenden Dateien:

`acts.cpp`	Hauptprogramm, Setzen von Parametern
`AttList.cpp`	Liste symbolischer Attributwerte
`AttList.hh`	Header dazu
`BinList.cpp`	Liste von Intervallen
`BinList.hh`	Header dazu
`Classifier.cpp`	Klassifikationsverfahren
`Classifier.hh`	Header dazu
`Instance.cpp`	Instanz mit symbolischen Attributen
`Instance.hh`	Header dazu
`Makefile`	Makefile des ACTS-Systems
`ProtoList.cpp`	Liste von prototypischen Zeitreihen
`ProtoList.hh`	Header dazu
`Prototype.cpp`	Prototypische Zeitreihe
`Prototype.hh`	Header dazu
`QuinlanDB.cpp`	Kern von ACTS, Clustering, Ausgabemethoden
`QuinlanDB.hh`	Header dazu
`Ugly.cpp`	Eine unschöne Konvertierung
`Ugly.hh`	Header dazu

Da ich das gleiche Eingabeformat benutze wie in RADTKE (1998) definiert, benutze ich Methoden aus dieser Arbeit zum Aufbau von Objekten mit Rohdaten. Der Sourcecode ist dabei von mir leicht modifiziert worden, um Namenskonflikte mit $\mathcal{MLG}$++auszuräumen. `TRUE` ist jetzt `TRUEVAL`, `FALSE` ist jetzt `FALSEVAL` und `ASSERT` ist jetzt `ASSERTATION`.

Die von mir benutzten Sourcecode-Dateien sind:

`Change.hh`	
`DataTrace.hh`	`DataTrace.cpp`
`DatabaseAccessor.hh`	
`DiscreteTrace.hh`	`DiscreteTrace.cpp`
`FakeDB.hh`	`FakeDB.cpp`
`Global.hh`	`Global.cpp`
`LinearTrace.hh`	`LinearTrace.cpp`
`Record.hh`	

B.2 Pfade

Im folgenden sind Pfade für die verwendeten Programme und Libraries angegeben. Alle Pfadangaben sind relativ zum Verzeichnis `/export/home/cassens/` auf dem Rechner `streisand.offis.uni-oldenbu`

ACTS	`src/da`
$\mathcal{MLC}$++-Libraries	`mlc/src/*`
$\mathcal{MLC}$++-Utilities	`mlc/util`
LEDA	`mlc/leda/src/*`
dotty	`opt/graphviz/bin`
GNUPLOT	`bin`
NEFCLASS	`opt/Nefclass-X`
TCL/TK	`bin`
CN2	`mlc/cn2`
RIPPER	`opt/ripper/bin`

Die Verfahren ID3, MC4 und EODG sind Teil von $\mathcal{MLC}$++.

Anhang C

Dateiformate

Auf den folgenden Seiten sind beispielhaft Dateien abgedruckt, die von ACTS generiert werden können.

C.1 Ausgabeformate

C.1.1 Definitionsdatei

In dieser Datei wird gemäß des von QUINLAN (1993) eingeführten Formats festgelegt, welche Attribute und Klassenzugehörigkeiten in der Trainings- bzw. Testmenge vorkommen können. Im Beispiel sind die statischen Attribute diskretisiert. Die Filename-Extension ist `*.names`.

```
| Machine readable .names file for MLC++ library.
| Generated automatically by ACTS

| classes:
ill, healthy

| preprocessed attributes:
linear_0: 0, 1.
linear_1: 0, 1.
discrete_0: 0, 1.
discrete_1: 0, 1.

| preprocessed static attributes:
static_0: 0, 1, 2, 3.
static_1: 0, 1, 2.
```

C.1.2 Datendatei

Diese Datei enthält entsprechend QUINLAN (1993) die Daten für jede Instanz. Die Filename-Extension ist `*.data`. Die ersten beiden Dateien können z.B. mit den $\mathcal{MLC}$++-Utilities weiter verarbeitet werden.

```
| Machine readable .data file for MLC++ library.
| Generated automatically by ACTS
| Number of Instances:  100
| Number of Attributes: 6
1, 0, 0, 0, 2, 1, ill.
0, 0, 1, 1, 1, 0, ill.
1, 1, 1, 1, 3, 2, healthy.
1, 1, 0, 0, 0, 1, healthy.
0, 0, 0, 0, 3, 2, ill.
1, 1, 1, 1, 3, 2, healthy.
0, 0, 1, 1, 1, 0, ill.
1, 1, 1, 1, 3, 2, healthy.
0, 0, 0, 0, 3, 2, ill.
1, 0, 0, 0, 2, 1, ill.
1, 1, 1, 1, 3, 2, healthy.
1, 0, 0, 0, 2, 1, ill.
1, 1, 1, 1, 3, 2, healthy.
0, 0, 1, 1, 1, 0, ill.
1, 1, 1, 1, 3, 2, healthy.
0, 0, 0, 0, 3, 2, ill.
1, 1, 1, 1, 3, 2, healthy.
0, 0, 0, 0, 3, 2, ill.
1, 1, 0, 0, 0, 1, healthy.
0, 0, 1, 1, 1, 0, ill.
1, 1, 0, 0, 0, 1, healthy.
0, 0, 1, 1, 1, 0, ill.
1, 1, 0, 0, 0, 1, healthy.
1, 0, 0, 0, 2, 1, ill.
1, 1, 1, 1, 3, 2, healthy.
1, 0, 0, 0, 2, 1, ill.
1, 0, 0, 0, 2, 1, ill.
1, 1, 0, 0, 0, 1, healthy.
0, 0, 0, 0, 3, 2, ill.

⋮

1, 1, 1, 1, 3, 2, healthy.
```

C.1.3 NEFCLASS

Diese Datei enthält die lokale Umkodierung der Daten für das Nefclass-System. Der Übersichtlichkeit halber und um überlange Zeilen zu vermeiden, habe ich die Umgebungsvariable ACTS_NEFCLASS_PRECISION (vgl. Anhang D.2) für dieses Beispiel auf den Wert 2 gesetzt. Eine genauere Beschreibung des Formats findet sich in Hoferichter (1997). Die Filename-Extension ist `*.dat`.

```
# Machine readable .dat file for NEFCLASS.
# Generated automatically by ACTS
# Number of Instances:  100
# Original number of Attributes: 6
# Local encoded: 10

# Dimensions attributes/classes (required)
DIMENSIONS
10   2

# Description of data set (optional)
NAME
ExperimentProto5

# Ranges of variables (optional)
# I leave it out since I haven't computed it yet
#INRANGES

# (optional) names of input features and classes
VARNAMES
linear_0 prototype 0
linear_0 prototype 1
linear_1 prototype 0
linear_1 prototype 1
discrete_0 prototype 0
discrete_0 prototype 1
discrete_1 prototype 0
discrete_1 prototype 1
static_0 (raw)
static_1 (raw)
Ill
Healthy
```

...

```
# And now: the data.
# Beginning with the nummber of patterns (required)
PATTERNS
100
0.13 0.02 0.00 0.09 0.15 0.18 0.06 0.32 0.67 0.60   0 1
0.03 0.18 0.03 0.05 0.38 0.05 0.44 0.06 0.50 0.48   0 1
0.13 0.01 0.07 0.02 0.29 0.04 0.35 0.04 1.00 1.00   1 0
0.18 0.03 0.17 0.09 0.06 0.27 0.11 0.27 0.40 0.60   1 0
0.03 0.12 0.03 0.05 0.23 0.56 0.18 0.56 1.00 1.00   0 1
0.13 0.01 0.07 0.02 0.30 0.03 0.35 0.03 1.00 1.00   1 0
0.03 0.18 0.04 0.05 0.38 0.05 0.44 0.06 0.50 0.48   0 1
0.13 0.01 0.07 0.02 0.30 0.03 0.35 0.03 1.00 1.00   1 0
0.02 0.12 0.03 0.06 0.23 0.56 0.18 0.56 1.00 1.00   0 1
0.14 0.01 0.07 0.01 0.30 0.03 0.35 0.03 1.00 1.00   1 0
0.13 0.02 0.01 0.09 0.15 0.18 0.06 0.32 0.67 0.60   0 1
0.13 0.02 0.07 0.02 0.29 0.04 0.35 0.03 1.00 1.00   1 0
0.03 0.18 0.03 0.05 0.39 0.06 0.44 0.05 0.50 0.48   0 1
0.13 0.01 0.07 0.01 0.30 0.03 0.35 0.03 1.00 1.00   1 0
0.14 0.00 0.07 0.02 0.29 0.04 0.35 0.03 1.00 1.00   1 0
0.03 0.12 0.04 0.05 0.23 0.56 0.18 0.56 1.00 1.00   0 1
0.12 0.02 0.07 0.01 0.29 0.04 0.35 0.03 1.00 1.00   1 0
0.02 0.12 0.03 0.05 0.23 0.56 0.18 0.56 1.00 1.00   0 1
0.19 0.04 0.17 0.08 0.06 0.27 0.12 0.27 0.40 0.60   1 0
0.03 0.18 0.03 0.05 0.38 0.05 0.44 0.06 0.50 0.48   0 1
0.19 0.05 0.17 0.09 0.06 0.27 0.11 0.27 0.40 0.60   1 0
0.04 0.19 0.03 0.06 0.38 0.05 0.44 0.06 0.50 0.48   0 1
0.18 0.04 0.17 0.09 0.06 0.27 0.11 0.27 0.40 0.60   1 0
0.14 0.01 0.06 0.02 0.30 0.03 0.35 0.04 1.00 1.00   1 0
0.14 0.01 0.01 0.09 0.15 0.18 0.06 0.32 0.67 0.60   0 1
0.18 0.04 0.18 0.09 0.06 0.27 0.11 0.27 0.40 0.60   1 0
0.03 0.12 0.03 0.05 0.23 0.56 0.18 0.56 1.00 1.00   0 1
0.13 0.02 0.07 0.02 0.30 0.03 0.35 0.03 1.00 1.00   1 0
0.13 0.01 0.07 0.01 0.30 0.03 0.35 0.04 1.00 1.00   1 0
0.03 0.12 0.03 0.06 0.23 0.56 0.18 0.56 1.00 1.00   0 1
0.19 0.04 0.17 0.09 0.06 0.27 0.12 0.27 0.40 0.60   1 0
0.13 0.01 0.07 0.02 0.29 0.04 0.34 0.04 1.00 1.00   1 0
0.04 0.19 0.03 0.06 0.39 0.06 0.43 0.05 0.50 0.48   0 1
0.13 0.02 0.00 0.09 0.15 0.18 0.06 0.32 0.67 0.60   0 1

⋮

0.03 0.11 0.03 0.05 0.23 0.56 0.18 0.56 1.00 1.00   0 1
END
```

C.1.4 Statistik

Diese Datei enthält Daten über die Anzahl der erzeugten Prototypen pro Attribut, die verwendete Metrik sowie die Abweichungen pro gebildetem Cluster und für die jeweiligen Attribute. Die Filename-Extension ist `*.stat`.

```
Statistical Summary for Experiment ExperimentProto5 with:
Classes: ill, healthy
Number of linear attributes: 2
Number of discrete attributes: 2
Number of static attributes: 2

The attributes in detail:

linear_0:
With metrics Mean has 2 prototypes
with overall distortion 0.0251451319355
 Proto 0 representing [34] instances
         with distortion 0.0109031516294
 Proto 1 representing [66] instances
         with distortion 0.0142419803061

linear_1:
With metrics Mean has 2 prototypes
with overall distortion 0.0335905725716
 Proto 0 representing [56] instances
         with distortion 0.0128799667629
 Proto 1 representing [44] instances
         with distortion 0.0207106058087

discrete_0:
With metrics Mean has 2 prototypes
with overall distortion 0.104862593333
 Proto 0 representing [60] instances
         with distortion 0.0884043433333
 Proto 1 representing [40] instances
         with distortion 0.01645825

discrete_1:
With metrics Mean has 2 prototypes
with overall distortion 0.0847940616667
 Proto 0 representing [60] instances
         with distortion 0.0681264366667
 Proto 1 representing [40] instances
         with distortion 0.016667625
```

...

```
static_0: 4 Bins with distortion 0.00000000000000699440505514
 Bin 0 Value "0.4" representing [19] entries
       in range [0.4..0.4]
 Bin 1 Value "0.5" representing [15] entries
       in range [0.5..0.5]
 Bin 2 Value "0.666667" representing [22] entries
       in range [0.666667..0.666667]
 Bin 3 Value "1" representing [44] entries
       in range [1..1]

static_1: 3 Bins with distortion 0.00000000000000215383266777
 Bin 0 Value "0.48" representing [15] entries
       in range [0.48..0.48]
 Bin 1 Value "0.6" representing [41] entries
       in range [0.6..0.6]
 Bin 2 Value "1" representing [44] entries
       in range [1..1]
```

C.1.5 Prototypen

Hierbei handelt es sich um eine für die Weiterverarbeitung mit z.B. GNUPLOT vorbereitete Ausgabe einer prototypschen Zeitreihe (hier eines linearen Attributes). Ähnliche Dateien werden auch für jedes Cluster erzeugt, dabei können alle Zeitreihen, die zu einem Cluster gehören, übereinander geplottet werden. Die Filename-Extension ist `*.disc.#.#` bzw. `*.lin.#.#`.

```
# Machine readable prototype file for GnuPlot.
# Generated automatically by ACTS
# Prototype for linear Attributes
# Attribute Nr. 0
# Metrics used: Mean
# stands for 56 instances
# distortion 0.0128799667629
0 0.347550516129
0.01 0.351480483871
0.02 0.351107225806
0.03 0.355043032258
0.04 0.357883741935
0.05 0.352284612903
0.06 0.360659419355
0.07 0.35966483871
0.08 0.357068548387
0.09 0.36050683871
0.1 0.361325193548

⋮

0.88 0.512198096774
0.89 0.512410870968
0.9 0.509894580645
0.91 0.521453354839
0.92 0.521787903226
0.93 0.523722129032
0.94 0.515370096774
0.95 0.520497129032
0.96 0.528784419355
0.97 0.533843064516
0.98 0.532831935484
0.99 0.53495066129
1 0.537069387097

# eof ExperimentProto5.lin.1.0
```

C.1.6 GnuPlot

Mit dieser Datei wird GNUPLOT angesteuert. Bei diesem Beispiel werden die Prototypen eines Experiments hintereinander in eine PostScript-Datei geschrieben. Ähnliche Dateien können für die Darstellung von Clustern erzeugt werden, hierbei werden alle Zeitreihen, die zu einem Cluster gehören, übereinander geplottet. Weiterhin kann eine GNUPLOT-Steuerdatei ausgegeben werden, mit deren Hilfe komfortabel EPSF-Dateien für jeden Prototypen und jeden Cluster generiert werden können. Die Filename-Extension ist `*.gnuplot`.

```
# Visualization of Prototypes for ExperimentProto5
# generated by ACTS.
set title "ExperimentProto5 prototypes"
set output "ExperimentProto5.pr.ps"
# Want output for papers? You may use EPS and
# one image per output file instead.
set terminal postscript monochrome
set xlabel "time"
set ylabel "value"
# and now: the data
#
plot "ExperimentProto5.lin.0.0" title "linear_0 # 0 [34] LP2"
      with lines
plot "ExperimentProto5.lin.0.1" title "linear_0 # 1 [66] LP2"
      with lines
#
plot "ExperimentProto5.lin.1.0" title "linear_1 # 0 [56] Trend"
      with lines
plot "ExperimentProto5.lin.1.1" title "linear_1 # 1 [44] Trend"
      with lines
#
plot "ExperimentProto5.disc.0.0" title "discrete_0 # 0 [60] Mean"
      with lines
plot "ExperimentProto5.disc.0.1" title "discrete_0 # 1 [40] Mean"
      with lines
#
plot "ExperimentProto5.disc.1.0" title "discrete_1 # 0 [60] Mean"
      with lines
plot "ExperimentProto5.disc.1.1" title "discrete_1 # 1 [40] Mean"
      with lines
# eof
```

C.2 Eingabeformate

C.2.1 Rohdaten

In einer solchen Datei werden für jedes Individuum („Record“) die einzelnen Zeitreihen und statischen Attribute aufgelistet. Für eine ausführliche Darstellung verweise ich auf RADTKE (1998). Die Filename-Extension ist `*.db`.

```
FakeDB template file
 NumRecords = 100
 NumLinearAttributes = 2
 NumDiscreteAttributes += 2
 NumStaticAttributes = 2

 Record 0 {

   Classifier : 0
   Static:  0:0.666667  1:0.6  ;

   0     0:0.658083  1:0.363092  2:0.279  3:0.095  ;
   0.01  0:0.688069  1:0.354298  2:0.312  3:0.091  ;


⋮


   0.59  0:0.815708  1:0.482993  2:0.285  3:0.099  ;
   0.6   0:0.77711  1:0.444389  2:0.297  3:0.099  ;
   0.61  0:0.685894  1:0.423508  2:0.9  3:0.99  ;
   0.62  0:0.720429  1:0.434115  2:0.91  3:0.96  ;
   0.63  0:0.749056  1:0.499252  2:0.291  3:0.096  ;
   0.64  0:0.79073  1:0.432751  2:0.327  3:0.109  ;
   0.65  0:0.727468  1:0.411471  2:0.315  3:0.091  ;
   0.66  0:0.882604  1:0.504738  2:0.315  3:0.096  ;
   0.67  0:0.859828  1:0.497357  2:0.321  3:0.109  ;
   0.68  0:0.629356  1:0.485287  2:0.91  3:0.97  ;
   0.69  0:0.738054  1:0.468479  2:0.98  3:0.91  ;
   0.7   0:0.758369  1:0.479468  2:0.324  3:0.109  ;
   0.71  0:0.82505  1:0.471837  2:0.291  3:0.104  ;
   0.72  0:0.785579  1:0.46414  2:0.291  3:0.105  ;
   0.73  0:0.745966  1:0.442261  2:0.315  3:0.102  ;
   0.74  0:0.755479  1:0.486318  2:0.306  3:0.101  ;
   0.75  0:0.699928  1:0.516459  2:0.97  3:0.96  ;
   0.76  0:0.74226  1:0.513516  2:0.92  3:0.95  ;
   0.77  0:0.775937  1:0.472369  2:0.288  3:0.102  ;
```

...

```
   0.78  0:0.802089  1:0.526683  2:0.276  3:0.101  ;
   0.79  0:0.902876  1:0.451638  2:0.315  3:0.09  ;
   0.8   0:0.779971  1:0.433915  2:0.33  3:0.11  ;
   0.81  0:0.864435  1:0.503142  2:0.291  3:0.097  ;
   0.82  0:0.648183  1:0.451471  2:0.97  3:0.9  ;
   0.83  0:0.746266  1:0.491987  2:0.97  3:0.9  ;
   0.84  0:0.919027  1:0.48389  2:0.285  3:0.103  ;
   0.85  0:0.887124  1:0.446301  2:0.324  3:0.106  ;
   0.86  0:0.78804  1:0.52655  2:0.273  3:0.091  ;
   0.87  0:0.831373  1:0.488828  2:0.324  3:0.11  ;
   0.88  0:0.79073  1:0.48552  2:0.297  3:0.106  ;
   0.89  0:0.643605  1:0.482178  2:0.93  3:0.92  ;
   0.9   0:0.743062  1:0.468828  2:0.9  3:0.94  ;
   0.91  0:0.837039  1:0.450374  2:0.324  3:0.102  ;
   0.92  0:0.796109  1:0.542244  2:0.306  3:0.097  ;
   0.93  0:0.772003  1:0.54404  2:0.273  3:0.109  ;
   0.94  0:0.824292  1:0.490241  2:0.324  3:0.091  ;
   0.95  0:0.808655  1:0.491854  2:0.276  3:0.101  ;
   0.96  0:0.759256  1:0.473117  2:0.92  3:0.94  ;
   0.97  0:0.682575  1:0.469559  2:0.9  3:0.91  ;
   0.98  0:0.787067  1:0.506933  2:0.327  3:0.099  ;
   1     0:0.942632  1:0.477756  2:0.303  3:0.11  ;
 }

⋮

 Record 99 {

   Classifier : 0
   Static:  0:0.5  1:0.48  ;

   0     0:0.399142  1:0.35611  2:0.93  3:0.92  ;
   0.01  0:0.404778  1:0.330524  2:0.97  3:0.94  ;

⋮

   1     0:0.616481  1:0.57217  2:0.93  3:0.97  ;
 }
End
```

C.2.2 Metriken

In der folgenden Form kann für ein Experiment vorgegeben werden, welche Metrik für das jeweilige Attribut zu benutzen ist. Dies ermöglicht die Integration von Anwendungswissen in die Testläufe. Die Filename-Extension ist `*.class`.

```
| Classifiers to use with ExperimentProto5
X-Corr
Trend
LP2
Hamming
| EOF ExperimentProto5.class
```

Anhang D

Umgebung

Damit auf `streisand.offis.uni-oldenburg.de` mit ACTS, den $\mathcal{MLC}$++-Utilities, NEFCLASS und weiteren Programmen zum Maschinellen Lernen gearbeitet werden kann, sollte die Arbeitsumgebung wie folgt modifiziert werden:

D.1 Allgemein

Es folgt ein Auszug aus den Dateien `.cshrc` und `.mlcrc` des Logins `cassens`. Unter anderem wird der Suchpfad so modifiziert, so daß die benötigten ausführbaren Dateien gefunden werden. Außerdem werden Umgebungsvariablen gesetzt, auf die $\mathcal{MLC}$++ zugreift.

```
#################################################################
# Some environment variables must be set
# Misc stuff

# some basic executables like GnuPlot and Tcl/Tk-stuff...
set path=(. /export/home/cassens/bin $path )
setenv MANPATH "/export/home/cassens/man:"$MANPATH

# ...and adjacent libraries
setenv CPLUS_INCLUDE_PATH
       $CPLUS_INCLUDE_PATH":/export/home/cassens/include"
setenv LD_LIBRARY_PATH $LD_LIBRARY_PATH":/export/home/cassens/lib"

# Graph Visualisation (used e.g. by mlc++-utilities)
set path=( $path /export/home/cassens/opt/graphviz/bin )
# manpages reside in /export/home/cassens/man
```

...

```
# tcl/tk (used e.g. by NEFCLASS-frontend)
setenv TCL_LIBRARY /export/home/cassens/lib
setenv TK_LIBRARY /export/home/cassens/lib
# manpages reside in /export/home/cassens/man

######################################################################
# MLC++ related stuff
# Basics

# where is mlc++?
setenv MLCDIR "/export/home/cassens/mlc"

# where is LEDA?
setenv LEDAROOT "/export/home/cassens/mlc/leda"

# Where do the libraries reside?
# leda
setenv LD_LIBRARY_PATH $LD_LIBRARY_PATH":${MLCDIR}/leda"
# mlc++
setenv LD_LIBRARY_PATH $LD_LIBRARY_PATH":${MLCDIR}/src/lib/gnu-SUNOS"

# for sources including mlc++-libraries, a wrapper for gcc should
# be used to make clear that compiler-flags are properly set
alias   mlcc     '$MLCDIR/bin/mlc_cc'

# mlc++ executables
set path=( $path /export/home/cassens/opt/mlc )

# Setup platform variables for SUN GNU
setenv MLC_COMPILER GNU
setenv MLC_OS SUNOS
setenv MLC_OS_VERSION 5.5.1
setenv MLC_EXTERNAL_TYPE GNU
setenv MLC_MODE standard

######################################################################
# Path settings

# Set MLCPATH variable; this specifies the search path for
# datafiles when running MLC++ executables
setenv MLCPATH ".:$MLCDIR/db"

# which cn2-Version to use
setenv CN2_PGM_NAME /export/home/cassens/mlc/external/GNU/cn

######################################################################
# Misc settings

# Display Options - use a visualization tool whenever possible
setenv DISPLAY_STRUCT dotty
setenv DISPGRAPH yes
```

...

```
# Ask for every options while testing
setenv PROMPTLEVEL all

# how much information to be displayed
setenv LOGLEVEL 2
setenv SHOW_LOG_ORIGIN yes

#####################################################################
# ML-algorithm interfaced by ACTS/MLC++
# manpages reside in /export/home/cassens/man

# ripper-Algorithm (rule induction)
set path=( $path /export/home/cassens/opt/ripper/bin )

# Aspirin/MIGRAINES V 6.0 (neural networks)
set path=( $path /export/home/cassens/opt/am6/bin/sun_sparc )
setenv NNTOOLS /export/home/cassens/opt/am6

# C5.0R1 demo (tree induction)
set path=( $path /export/home/cassens/opt/C50Release1/bin )

# NEFCLASS (neuro fuzzy)
set path=( $path /export/home/cassens/opt/Nefclass-X )
setenv NFCPATH /export/home/cassens/opt/Nefclass-X

# other ml-Algorithms (external inducers) are provided by mlc++
# T2, CN2, IBL, OC1, PEBLS
set path=( $path ${MLCDIR}/external/GNU )
```

Für eine effiziente Arbeit mit den $\mathcal{MLC}$++-Utilities sind vor allem folgende Umgebungsvariablen wichtig:

Variablenname	Wertebereich	Default	Bedeutung
INDUCER	Verfahren	–	Name des benutzten Induktionsverfahrens
DATAFILE	Dateiname	–	Stammname der zu nutzenden Dateien (ohne `*.data`, `*.names`).

D.2 ACTS-Steuervariablen

Die folgenden Variablen dienen der Steuerung des Verhaltens von ACTS. Dem hier angegebenen Name ist jeweils der Prefix ACTS_ voranzustellen, es ist also beispielsweise ACTS_CLASSIFIER statt CLASSIFIER zu schreiben.

Variablenname	Wertebereich	Default	Bedeutung
HAMMING_EPSILON	Real $\in [0 \ldots 1]$	0.1	Falls benutzt: Epsilonumgebung für die Hamming-Metrik.
EXPERIMENT	Dateiname	–	Stammname der zu nutzenden Dateien (ohne `*.db`, `*.class`).
CLASSIFIER	Metrik, „file“	–	Zu benutzende Metrik beim Clustering oder „file“ für `*.class`-Datei.
CONVERGE_EPSILON	Real $\in [0 \ldots 1]$	0.01	Epsilonumgebung für die Konvergenz beim Clustering der Zeitreihen.
MAX_PROTOS	Int. ≥ 2	6	Max. Anzahl von generierten Prototypen.
VAL_PROTOS	Int. $\in [10 \ldots 500]$	100	Max. Anzahl diskreter Zeitpunkte pro Prototyp.
SPLIT_CLUSTER	Boolean	false	Sollen volle Cluster beim Auftreten leerer Cluster geteilt werden?
STATIC_DISC	Boolean	false	Sollen Statische Attribute diskretisiert werden?
STATIC_BINS	Int. $\in [1 \ldots 100]$	5	Wenn ja: max. Anzahl der Intervalle.
STATIC_EPSILON	Real $\in [0 \ldots 1]$	0.01	Wenn ja: Epsilonumgebung für die Konvergenz beim Clustering.
DUMP_MLC	Boolean	true	Sollen Dateien im „Quinlan“-Format (z.B. zur Verwendung mit $\mathcal{MLC}$++) erzeugt werden?
DUMP_FAKE_TEST	Boolean	false	Soll eine Kopie der Lerndatei mit Endung `*.test` erzeugt werden?
PRINT_STAT	Boolean	true	Soll eine Statistik zum Clustering auf dem Bildschirm ausgegeben werden?
DUMP_STAT	Boolean	true	Soll eine Statistik zum Clustering in eine Datei geschrieben werden?
DUMP_PROTOS	Boolean	true	Sollen die Prototypen in eine Datei geschrieben werden?
DUMP_TRACES	Boolean	true	Sollen die Zeitreihen in eine Datei geschrieben werden?
DUMP_GNUPLOT	Boolean	true	Soll eine Steuerdatei für GNUPLOT (POSTSCRIPT, A4-Ausgabe) erzeugt werden?

...

Variablenname	Wertebereich	Default	Bedeutung
DUMP_GNU_EPS	Boolean	false	Soll eine Steuerdatei für GnuPlot (erstellt Encapsulated PostScript-Dateien) erzeugt werden?
DUMP_NEFCLASS	Boolean	false	Soll eine Datei zur Ansteuerung von Nefclass erzeugt werden?
NEFCLASS_PRECISION	Int. $\in [2 \ldots 10]$	5	Wenn ja: Genauigkeit der lokalen Umkodierung.

Des weiteren werden u.a. folgende Umgebungsvariablen ausgewertet, die auch das Verhalten von $\mathcal{MLC}$++ in entsprechender Weise steuern (hierfür ist kein Prefix erforderlich).

Variablenname	Wertebereich	Default	Bedeutung
PROMPTLEVEL	{required, basic, all}	required	Soll ein Default-Wert benutzt oder immer nachgefragt werden?
MLCPATH	Pfadnamen	pwd	Pfad für Eingabedateien (`*.db`, `*.class`).
LOGLEVEL	Int. ≥ 0	0	Wieviel an Information soll während des Programmlaufes angezeigt werden?
SHOW_LOG_ORIGIN	Boolean	false	Soll angezeigt werden, welche Klasse und Methode die Ausgabe erzeugt hat?

Abbildungen

Tabellen

Literatur

Aha, David W., 1991, Incremental Constructive Induction – An Instance-Based Approach. In: *Proceedings – IMLW-91*, Seiten 117–121, Morgan Kaufman, Evanston, URL `http://www.aic.nrl.navy.mil/~aha`.

Aha, David W., 1998, Sammlung von HomePages im Bereich Maschinelles Lernen. WWW, URL `http://www.aic.nrl.navy.mil/~aha/`.

Aha, David W., Dennis Kibler und Marc K. Albert, 1991, Instance-Based Learning Algorithms. *Machine Learning*, Band 6: Seiten 37–66.

Aristoteles, Metaphysik, *Übersetzt von Hermann Bonitz.* Nummer 544 in enzyklopädie, rororo, Reinbek bei Hamburg.

Auer, Peter, Robert C. Holte und Wolfgang Maass, 1995, Theory and Application of Agnostic PAC-Learning with Small Decision Trees. In: Prieditis und Russell (1995), Seiten 21–29, URL `http://www.site.uottawa.ca/~holte`.

Bartels, Stefan, 1998, *Untersuchungen epidemiologischer Fragestellungen mit Hilfe individuen-orientierter Simulationsmodelle – ein Basis-Modell.* Diplomarbeit, Fachbereich Informatik der Carl von Ossietzky Universität Oldenburg.

Bauer, Eric und Ron Kohavi, Unveröffentlicht, An Empirical Comparison of Voting Classification Algorithms – Bagging, Boosting, and Variants. *Machine Learning*, URL `http://robotics.stanford.edu/users/ronnyk`. Akzeptierter Artikel, erscheint voraussichtlich 1998.

Bensch, Hans-Georg, 1994, *Vom Reichtum der Gesellschaften.* Dissertation, Universität Hannover.

Bentrup, John A. und Sylvian R. Ray, 1993, *An Examination of Inductive Learning Algorithms for the Classification of Sleep Signals.* Report, Department of Computer Science, University of Illinois at Urbana-Champaign, URL `ftp://a.cs.uiuc.edu/pub/dept/tech_reports/1997/UIUCDCS-R-93-1%792.ps.gz`. Geänderte Fassung in: Proceedings des 30th Annual Rocky Mountain Bioengineering Symposium.

Berndt, Donald J. und James Clifford, 1996, Finding Patterns in Time Series – A Dynamic Programming Approach. In: Fayyad *et al.* (1996b), Kapitel 9, Seiten 229–248.

Biehler, Rolf, 1982, *Explorative Datenanalyse – Eine Untersuchung aus der Perspektive einer deskriptiv-empirischen Wissenschaftstheorie.* Nummer 24 in Materialien und Studien, Institut für Didaktik der Mathematik der Universität Bielefeld.

Blum, Robert L., 1982, *Discovery and Representation of Causal Relationships from a Large Time-Oriented Clinical Database: The RX Project.* Lecture Notes in Medical Informatics, Springer-Verlag, Berlin.

Blumer, Anselm, Andrzej Ehrenfeucht, David Haussler und Manfred K. Warmuth, 1987, Occam´s Razor. *Information Processing Letters*, Band 24: Seiten 377–380.

Boswell, Robin, 1990, *Manual for CN2 version 6.1.* Manual TI/P2154/RAB/4/1.5, The Turing Institute, URL `http://www.cs.utexas.edu/users/pclark/software.html`.

Bratko, Ivan, Bojan Cestnik und Igor Kononko, 1996, Attribute-Based Learning. *AI Communications*, Band 9(1): Seiten 27–32.

Bratko, Ivan und Stephen Muggleton, 1995, Applications of Inductive Logic Programming. *Communications of the ACM*, Band 38(11): Seiten 65–70.

Bratko, Ivan, Stephen Muggleton und Aram Karalič, 1998, Applications of Inductive Logic Programming. In: Michalski *et al.* (1998), Kapitel 4, Seiten 131–143.

Brazdil, Pavel B., Herausgeber, 1993, *Machine Learning – Proceedings ECML-93 in Wien*, Nummer 667 in Lecture Notes in AI, Berlin, Springer-Verlag.

Breiman, L., J. H. Friedman, R. A. Ohlsen und C. J. Stone, 1984, *Classification and Regression Trees (CART).* The Wadsworth statistics & probability series, Chapman & Hall, New York.

Breiman, Leo, 1996, Bagging Predictors. *Machine Learning*, Band 24: Seiten 123–140, URL `ftp://ftp.stat.berkeley.edu/pub/users/breiman/bagging.ps.Z`. Im Internet ist nur eine ältere Fassung zugänglich.

Carbonell, Jaime G., Ryszard S. Michalski und Tom M. Mitchell, 1983, An Overview of Machine Learning. In: Michalski *et al.* (1983), Seiten 3–23.

Cestnik, Bojan und Ivan Bratko, 1991, On Estimating Probabilities in Tree Pruning. In: Kodratoff (1991), Seiten 138–150.

Charniak, Eugene und Drew McDermott, 1985, *Artificial Intelligence.* Addison-Wesley, Reading.

Cheeseman, Peter und John Stutz, 1996, Bayesian Classification (AutoClass): Theory and Results. In: Fayyad *et al.* (1996b), Kapitel 6, Seiten 153–180.

Clark, Peter und Robin Boswell, 1991, Rule Induction with CN2: Some Recent Improvements. In: Kodratoff (1991), Seiten 151–163, URL `http://www.cs.utexas.edu/users/pclark/`.

Clark, Peter und Tim Niblett, 1989, The CN2 Induction Algorithm. *Machine Learning*, Band 3: Seiten 261–283, URL `http://www.cs.utexas.edu/users/pclark/`.

Cohen, William W., 1995, Fast Effective Rule Induction. In: Prieditis und Russell (1995), URL `http://www.research.att.com/~wcohen`.

Craven, Mark W. und Jude W. Shavlik, 1996, Extracting Tree-Structured Representations of Trained Networks. In: David S. Touretzky, Herausgeber, *Proceedings – NIPS-95 in Denver*, Band 8 von *Advances in Neural Information Processing Systems*, MIT Press, Cambridge, MA, URL `http://www.cs.cmu.edu/~craven`.

Dietterich, Thomas G. und Ryszard S. Michalski, 1983, A Comparative Review of Selected Methods for Learning from Examples. In: Michalski *et al.* (1983), Seiten 41–81.

Dorffner, G., Herausgeber, 1990, *Konnektionismus in Artificial Intelligence und Kognitionsforschung. Proceedings KONNAI-90 in Salzburg*, Nummer 252 in Informatik-Fachberichte, Berlin, Springer-Verlag.

Dorffner, Georg, 1991, *Konnektionismus – Von neuronalen Netzwerken zu einer „natürlichen" KI*. Leitfäden der angewandten Informatik, Teubner, Stuttgart.

Dougherty, James, Ron Kohavi und Mehran Sahami, 1995, Supervised and Unsupervised Discretization of Continuous Features. In: Prieditis und Russell (1995), Seiten 194–202, URL `http://robotics.stanford.edu/users/ronnyk`.

Dürer, Holger, Yildiray Ogurol, Alex Rummel, Thomas Waschulzik und Manfred B. Wischnewsky, 1997, Einsatz von Data Mining-Verfahren in der Kardiologie. *KI – Künstliche Intelligenz*, Band 1997(3): Seiten 37–41.

Emde, Werner und Dietrich Wettschereck, 1996, Relational Instance-Based Learning. In: Lorenzo Saitta, Herausgeber, *Proceedings – ICML-96 in Bari*, Morgan Kaufmann, San Francisco, URL `http://www.gmd.de/ml-archive/frames/papers/papers-frames.html%`.

Enzinger, Andrea, Frank Puppe und Gerhard Strube, 1994, Problemlösen ohne Suchen? *KI – Künstliche Intelligenz*, Band 1994(1): Seiten 73–81.

Fayyad, Usama, Gregory Piatetsky-Shapiro und Padhraic Smyth, 1996a, From Data Mining to Knowledge Discovery in Databases. *AI-Magazine*, Band 17: Seiten 37–54.

Fayyad, Usama M., Gregory Piatetsky-Shapiro, Padhraic Smyth und Ramasamy Uthurusamy, Herausgeber, 1996b, *Advances in Knowledge Discovery and Data Mining*. AAAI/MIT Press, Cambridge, USA.

Fensel, Dieter, 1993a, Ein integriertes System zum maschinellen Lernen aus Beispielen. *KI – Künstliche Intelligenz*, Band 1993(3): Seiten 17–23.

Fensel, Dieter, 1993b, JoJo – An Integration of Generalization and Specialization. In: *Proceedings – Workshop Knowledge and Data Engineering, Strasbourg*, URL `http://www.aifb.uni-karlsruhe.de/WBS/dfe/`.

Fensel, Dieter und Jörg Klein, 1991, A New Approach to Rule Induction and Pruning. In: *Proceedings of the 1991 IEEE Int. Conf. on Tools for AI, San Jose*, Seiten 538–539.

Fensel, Dieter, Jörg Klein und Ulrike Neubronner, 1993, RJ – An Environment for Learning from Example. In: *Proceedings – Avignon-93*, URL `http://www.aifb.uni-karlsruhe.de/WBS/dfe/`.

FENSEL, DIETER UND MARKUS WIESE, 1993, Refinement of Rule Sets with JoJo. In: BRAZDIL (1993), Seiten 378–383, URL `http://www.aifb.uni-karlsruhe.de/WBS/dfe/`.

FENSEL, DIETER UND MARKUS WIESE, 1994, From JoJo to Frog – Extending a bi-directional Search Strategy to a more flexible three-directional Search. In: C. Globig und D. Althoff, Herausgeber, *Proceedings – FGML-94*, Nummer LSA-095-01 in Forschungsberichte, Seiten 37–44, Zentrum für Lernende Systeme und Anwendungen, Universität Kaiserslautern, URL `http://www.aifb.uni-karlsruhe.de/WBS/dfe/`.

FRAWLEY, WILLIAM J., GREGORY PIATETSKY-SHAPIRO UND CHRISTOPHER J. MATHEUS, 1992, Knowledge Discovery in Databases: An Overview. *AI-Magazine*, Band 13(3): Seiten 213–218.

FRIEDMAN, JEROME, RON KOHAVI UND YEOGIRL YUN, 1996, Lazy Decision Trees. In: *Proceedings AAAI-96, Portland*, Seiten 717–724, AAAI/MIT Press, Menlo Park, URL `http://robotics.stanford.edu/users/ronnyk`.

GOETHE, JOHANN WOLFGANG, 1808, *Faust. Der Tragödie Erster Teil.* Reclam, Stuttgart, URL `http://gutenberg.aol.de/`.

GRASSHOFF, GERD, 1995, Computermodellierung des wissenschaftlichen Entdeckungsprozesses. *KI – Künstliche Intelligenz*, Band 1995(6): Seiten 32–35.

GUO, HENG UND SAUL B. GELFAND, 1992, Classification Trees with Neural Network Feature Extraction. *IEEE Transactions on Neural Networks*, Band 3(6): Seiten 923–933.

HAUX, REINHOLD, BIRGIT BRIGL, PETRA KAUP UND THOMAS WETTER, 1997, Wissensbasierte Entscheidungsunterstützung in der Medizin: Geleistetes und zu Leistendes. *KI – Künstliche Intelligenz*, Band 1997(3): Seiten 23f.

HIMMELREICH, JENS, 1995, *Eine subjektwissenschaftliche Betrachtung der Software-Entwicklung.* Diplomarbeit, Fachbereich Mathematik und Informatik der Universität Bremen.

HOFERICHTER, THOMAS, 1996, *Entwurf und Implementierung eines UNIX-basierten Softwaretools zu Realisierung des Neuro-Fuzzy Klassifikationsansatzes NEFCLASS.* Diplomarbeit, Fachbereich Informatik der Universität Braunschweig, URL `http://fuzzy.cs.uni-magdeburg.de/`.

HOFERICHTER, THOMAS, 1997, *NEFCLASS-X – User Manual.* Universität Magdeburg, URL `http://fuzzy.cs.uni-magdeburg.de/`.

HOLSHEIMER, MARCEL UND ARNO SIEBES, 1994, *Data Mining – The Search for Knowledge in Databases.* Report CS-R9406, Centrum voor Wiskunde en Informatica (CWI), Amsterdam, URL `ftp://ftp.cwi.nl/pub/CWIreports/AA/CS-R9406.ps.Z`.

JANTKE, KLAUS P., 1995, Aktualität der Wissensrepräsentation aus Sicht des Algorithmischen Lernens. *KI – Künstliche Intelligenz*, Band 1995(5): Seiten 6–12.

JOHN, GEORGE H., RON KOHAVI UND KARL PFLEGER, 1994, Irrelevant Features and the Subset Selection Problem. In: William W. Cohen und Haym Hirsh, Herausgeber, *Machine*

Learning – Proceedings of the Eleventh International Conference in New Brunswick, Seiten 121–129, Morgan Kaufmann, San Francisco, URL `http://robotics.stanford.edu/users/ronnyk`.

KAINDL, HERMANN, 1994, Problemlösen durch Suche. *KI – Künstliche Intelligenz*, Band 1994(1): Seiten 81–84.

KAUFMAN, LEONARD UND PETER J. ROUSSEEUW, 1990, *Finding Groups in Data – An Introduction to Cluster Analysis*. Probability and Mathematical Statistics, Wiley, New York.

KEARNS, MICHAEL J. UND ROBERT SHAPIRE, 1992, Toward Efficient Agnostic Learning. *Machine Learning*, Band 9: Seiten 275–302, URL `http://www.research.att.com/~mkearns`.

KODRATOFF, YVES, Herausgeber, 1991, *Machine Learning – Proceedings EWSL-91 in Porto*, Nummer 482 in Lecture Notes in AI, Berlin, Springer-Verlag.

KOHAVI, RON, 1993, Bottom-Up Induction of Oblivious Read-Once Decision Graphs. In: BRAZDIL (1993), URL `http://robotics.stanford.edu/users/ronnyk`.

KOHAVI, RON, 1994, Bottom-Up Induction of Oblivious Read-Once Decision Graphs: Strenghts and Limitations. In: *Proceedings AAAI-94, Seattle*, AAAI/MIT Press, Menlo Park, URL `http://robotics.stanford.edu/users/ronnyk`.

KOHAVI, RON UND CLAYTON KUNZ, 1997, Option Trees with Majority Votes. In: D. Fisher, Herausgeber, *Proceedings – ICML-97*, Seiten 161–169, Morgan Kaufman, URL `http://robotics.stanford.edu/users/ronnyk`.

KOHAVI, RON UND CHIA-HSIN LI, 1995, Oblivious Decision Trees, Graphs and Top-Down Pruning. In: Chris S. Mellish, Herausgeber, *International Joint Conference on Artificial Intelligence – Proceedings IJCAI-95 in Montreal*, Morgan Kaufmann, San Mateo, URL `http://robotics.stanford.edu/users/ronnyk`.

KOHAVI, RON UND DAN SOMMERFIELD, 1996, $\mathcal{MLC}$++ – *Machine Learning Library in C++*. Silicon Graphics, URL `http://www.sgi.com/Technology/mlc`.

KOHAVI, RON, DAN SOMMERFIELD UND JAMES DOUGHERTY, 1996, Data Mining using $\mathcal{MLC}$++ – A Machine Learning Library in C++. In: *Tools with Artificial Intelligence*, Seiten 234–245, IEEE Computer Society Press, URL `http://robotics.stanford.edu/users/ronnyk`.

KOHAVI, RON, DAN SOMMERFIELD UND JAMES DOUGHERTY, 1997, Data Mining using $\mathcal{MLC}$++ – A Machine Learning Library in C++. *International Journal on Artificial Intelligence Tools*, Band 6(4): Seiten 537–566, URL `http://robotics.stanford.edu/users/ronnyk`. Längere Version eines Artikels aus der Zeitschrift *Tools with Artificial Intelligence*.

KONONENKO, IGOR, 1991, Semi-Naive Bayesian Classifier. In: KODRATOFF (1991), Seiten 206–219.

KONONENKO, IGOR, 1993, Inductive and Bayesian Learning in Medical Diagnosis. *Applied Artificial Intelligence*, Band 7: Seiten 317–337.

Köster, Frank und Michael Sonnenschein, 1998, Individual-Oriented Modeling and Simulation – an Approach to support Epidemiological Research. In: *Proceedings – MSS-98 in San Diego.*

Köster, Frank und Michael Sonnenschein, 1999, An Approach to the Creation and Validation of Hypotheses about Cause-and-Effect Relationships in Individual-Oriented Models by Knowledge Discovery in Databases. In: *Proceedings – MSS-99 in San Francisco.* Akzeptierter Artikel.

Koutsofios, Eleftherios und Stephen C. North, 1996, *Editing Graphs with Dotty.* Manual, AT&T Bell Laboratories, Murray Hill, NJ, URL `http://www.research.att.com/sw/tools/graphviz/refs.html`.

Koutsofios, Eleftherios und Stephen C. North, ohne Datum, *Drawing Graphs with Dot.* Manual, AT&T Bell Laboratories, Murray Hill, NJ, URL `http://www.research.att.com/sw/tools/graphviz/refs.html`.

Kruse, Rudolf, 1996, Fuzzy-Systeme – Positive Aspekte der Unvollkommenheit. *Informatik Spektrum*, Band 19: Seiten 4–11.

Kruse, Rudolf und Detlef Nauck, 1996, Neuronale Fuzzy-Systeme. In: Georg Dorffner, Herausgeber, *Proceedings HeKoNN-96, Münster*, Seiten 1–10, GMD, St. Augustin, URL `http://fuzzy.cs.uni-magdeburg.de/`.

Kubat, Miroslav, Ivan Bratko und Ryszard S. Michalski, 1998, A Review of Machine Learning Methods. In: Michalski *et al.* (1998), Kapitel 1, Seiten 3–96.

Kubat, Miroslav, Doris Flotzinger und Gert Pfurtscheller, 1993, Discovering Patterns in EEG-Signals: Comparative Study of a few Methods. In: Brazdil (1993), Seiten 366–377.

Langley, Pat, Gary L. Bradshaw und Herbert A. Simon, 1983, Rediscovering Chemistry with the Bacon System. In: Michalski *et al.* (1983), Seiten 307–329.

Langley, Pat, Wayne Iba und Kevin Thompson, 1992, An Analysis of Bayesian Classifiers. In: *Proceeding AAAI-92*, Seiten 223–228, Morgan Kaufmann, San Mateo, URL `http://www.isle.org/~iba/`.

Langley, Pat und Herbert A. Simon, 1995, Applications of Machine Learning and Rule Induction. *Communications of the ACM*, Band 38(11): Seiten 55–64.

Leiser, Eckart, 1983, *Grundkurs Statistik.* Studien zur Kritischen Psychologie, Pahl Rugenstein, Köln.

Long, Richard, 1994, *Overview of MLC++ Library Classes.* Silicon Graphics, URL `http://www.sgi.com/Technology/mlc`.

Mehlhorn, Kurt, Stefan Näher, Michael Seel und Christian Uhrig, 1998, *The LEDA User Manual Version 3.6.* Manual, Max-Planck-Institut für Informatik, Saarbrücken, URL `http://www.mpi-sb.mpg.de/LEDA/leda.html`.

MICHALSKI, RYSZARD S., 1983, A Theory and Methodology of Inductive Learning. In: MICHALSKI *et al.* (1983), Seiten 83–129.

MICHALSKI, RYSZARD S., IVAN BRATKO UND MIROSLAV KUBAT, Herausgeber, 1998, *Machine Learning and Data Mining*. Wiley, Chichester.

MICHALSKI, RYSZARD S., JAIME G. CARBONELL UND TOM M. MITCHELL, Herausgeber, 1983, *Machine Learning – An Artificial Intelligence Approach*. Tioga Publishing, Palo Alto.

MICHALSKI, RYSZARD S., IGOR MOZETIC, JIARONG HONG UND NADA LAVRAC, 1986, The Multi-Purpose Incremental Learning System AQ15 and its Testing Application to three Medical Domains. In: *Proceeding AAAI-86, Philadelphia*, Seiten 1041–1045, Morgan Kaufman, Los Altos, CA.

MINSKY, MARVIN LEE UND SEYMOUR PAPERT, 1969, *Perceptrons – an Introduction to Computational Geometry*. MIT Press, Cambridge, MA.

MORIK, KATHARINA, 1995, Maschinelles Lernen. In: Günther Görz, Herausgeber, *Einführung in die Künstliche Intelligenz*, Seiten 243–297, Addison-Wesley, Bonn.

MUGGLETON, STEPHEN, 1993, Inductive Logic Programming: Derivations, Successes and Shortcommings. In: BRAZDIL (1993), Seiten 21–37.

MURPHY, PATRICK M. UND MICHAEL J. PAZZANI, 1994, Exploring the Decision Forest: An Empirical Investigation of Occam's Razor in Decision Tree Induction. *JAIR – Journal of Artificial Intelligence Research*, Band 1: Seiten 257–275, URL `http://www.cs.washington.edu/research/jair/home.html`.

NAUCK, DETLEF, 1997, Ein Neuro-Fuzzy System zur Funktionsapproximation. In: *2. Int. Workshop Neuronale Netze in Industrieanwendungen, Stuttgart*, Institut für Statik und Dynamik von Luftfahrzeugen der Universität Stuttgart, URL `http://fuzzy.cs.uni-magdeburg.de/`.

NAUCK, DETLEF UND FRANK KLAWONN, 1995, NEFCLASS – A Neuro-Fuzzy Approach for the Classification of Data. In: K. M. George, Janice H. Carroll, Ed Deaton, Dave Oppenheim und Jim Hightower, Herausgeber, *Proceedings SAC-95, Nashville*, Seiten 461–465, ACM Press, New York, URL `http://fuzzy.cs.uni-magdeburg.de/`.

NAUCK, DETLEF UND FRANK KLAWONN, 1996, Neuro-Fuzzy Classification by Fuzzy Clustering. In: Hans-Jürgen Zimmermann, Herausgeber, *Proceedings EUFIT-96, Aachen*, URL `http://fuzzy.cs.uni-magdeburg.de/`.

NAUCK, DETLEF, ULRIKE NAUCK UND RUDOLF KRUSE, 1996, Generating Classification Rules with the Neuro-Fuzzy System NEFCLASS. In: Michael H. Smith, Herausgeber, *Proceedings NAFIPS-96, Berkeley*, IEEE, New York, URL `http://fuzzy.cs.uni-magdeburg.de/`.

NEUNEIER, RALPH UND HANS-GEORG ZIMMERMANN, 1996, Neuro-Fuzzy Methoden in der Finanzanalyse. *KI – Künstliche Intelligenz*, Band 1996(4): Seiten 55–58.

NÚÑEZ, MARLON, 1991, The Use of Background Knowledge in Decision Tree Induction. *Machine Learning*, Band 6: Seiten 231–250.

PIATETSKY-SHAPIRO, GREGORY UND MICHAEL BEDDOWS, 1998, KDNuggets? Directory – Data Mining and Knowledge Discovery Resources. WWW, URL `http://www.kdnuggets.com/`.

PRIEDITIS, ARMAND UND STUART RUSSELL, Herausgeber, 1995, *Proceedings – ML-95 in Tahoe City*, San Francisco, Morgan Kaufmann.

PRYKE, ANDY, 1998, Bibliographie zu Data Mining und Knowledge Discovery in Databases. WWW, URL `http://www.cs.bham.ac.uk/~anp/`.

QUINLAN, J. ROSS, 1983, Learning efficient Classification Procedures and their Application to Chess End Games. In: MICHALSKI *et al.* (1983), Seiten 463–482.

QUINLAN, J. ROSS, 1986, Induction of Decision Trees. *Machine Learning*, Band 1: Seiten 81–106.

QUINLAN, J. ROSS, 1990, Learning Logical Definitions from Relations. *Machine Learning*, Band 5: Seiten 239–266.

QUINLAN, J. ROSS, 1993, *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA.

QUINLAN, J. ROSS, 1998, Ohne Titel – Vergleich C4.5 und C5. WWW, URL `http://www.rulequest.com/`.

QUINLAN, J. ROSS UND R. M. CAMERON-JONES, 1993, FOIL: A Midterm Report. In: BRAZDIL (1993), Seiten 4–20.

RADTKE, ROLAND, 1998, *Analysemethoden zur Untersuchung individuenorientierter Modelle in der Epidemiologie*. Diplomarbeit, Fachbereich Informatik der Carl von Ossietzky Universität Oldenburg.

RITTER, HELGE, KLAUS SCHULTEN UND THOMAS MARTINETZ, 1991, *Neuronale Netze – Eine Einführung in die Neuroinformatik selbstorganisierender Netzwerke*. Addison-Wesley, Bonn.

RUMMEL, ALEX, 1997, *Knowledge Extraction from Databases – Using Available Data More Efficiently Through Multi-Events*. DISKI, infix, Sankt Augustin.

RUSCHIG, ULRICH, 1995, Korruption der Wissenschaft. In: Jörg Cassens, Herausgeber, *Tagungsbändchen JOINT-95*, Nummer 3 in Schriftenreihe Kritische Wissenschaft, Seiten 3–17, akki, Oldenburg.

SCHEFE, PETER, 1991, *Künstliche Intelligenz – Überblick und Grundlagen*. BI Wissenschaftsverlag, Mannheim.

SEITZ, ALEXANDER UND ADELINDE M. UHRMACHER, 1998, The treatment of Time in a Case-Based Analysis of Experimental Medical Studies. In: Otthein Herzog und Andreas Günter, Herausgeber, *Proceedings KI-98 – Advances in Artificial Intelligence*, Nummer 1504 in Lecture Notes in AI, Seiten 213–223, Springer, Berlin.

SIMON, HERBERT A., 1983, Why should Machines Learn? In: MICHALSKI *et al.* (1983), Seiten 25–37.

Smyth, Padhraic und Rodney M. Goodman, 1992, An Information Theoretic Approach to Rule Induction from Databases. *IEEE Transactions on Knowledge and Data Engineering*, Band 4: Seiten 301–316.

Sonnenschein, Michael und Ute Vogel, 1996, *Diskrete Modellierung und Simulation ökologischer Systeme*. Interner Bericht, Fachbereich Informatik der Carl von Ossietzky Universität Oldenburg.

Utgoff, Paul E., 1989, Incremental Induction of Decision Trees. *Machine Learning*, Band 4: Seiten 161–186.

Valiant, L. G., 1984, A Theory of the Learnable. *Communications of the ACM*, Band 27(11): Seiten 1134–1142.

Vaughn, Marilyn Lougher, 1996, Interpretation and Knowledge Discovery from the Multilayer Perceptron Network: Opening the Black-Box. *Neural Computing & Applications*, Band 4: Seiten 72–82.

Waschulzik, T. und H. Geiger, 1990a, Eine Entwicklungsmethodik für strukturierte konnektionistische Systeme. In: Dorffner (1990), Seiten 202–206.

Waschulzik, T. und H. Geiger, 1990b, Theorie und Anwendung strukturierter konnektionistischer Systeme. In: Dorffner (1990), Seiten 143–152.

Waschulzik, T., K. Quandt, M. Lewis, A. Hörmann, R. Engelbrecht und W. Brauer, 1993, Evaluation of an Epidemiological Data Set as an Example of the Application of Neural Networks to the Analysis of Large Medical Data Sets. In: S. Andreassen, R. Engelbrecht und J. Wyatt, Herausgeber, *Artificial Intelligence in Medicine*, Nummer 10 in Technology and Informatics, Seiten 466–476, IOS Press, Amsterdam.

Wasserman, Philip D., 1989, *Neural Computing: Theory and Practice*. Van Nostrand Reinhold, New York, NY.

Weiss, Sholom M. und Casimir A. Kulikowski, 1991, *Computer Systems that Learn*. Morgan Kaufmann, San Francisco, CA.

Wiese, Markus, 1996, A Bidirectional ILP Algorithm. In: *Proceedings – ILP for KDD, ICML-96*, URL `http://www.aifb.uni-karlsruhe.de/Staff/mwi.html`.

Wrobel, Stefan, 1998, Data Mining und Wissensentdeckung in Datenbanken. *KI – Künstliche Intelligenz*, Band 1998(1): Seiten 6–10.

Wrobel, Stefan und Saso Džeroski, 1995, The ILP Description Learning Problem – Towards a general Model-Level Definition of Data Mining in ILP. In: K. Morik und J. Herrmann, Herausgeber, *Proceedings Fachgruppentreffen Maschinelles Lernen (FGML-95)*, Universität Dortmund, URL `ftp://ftp.gmd.de/ml-archive/GMD/papers/ML68.ps.gz`.

Zahlmann, Gudrun, Mathias Scherf und Aharon Wegner, 1997, Einsatz eines Neuro-Fuzzy-Klassifikators beim Glaukom-Monitoring. *KI – Künstliche Intelligenz*, Band 1997(4): Seiten 65–66.