Wissensgewinnung aus großen Datenbasen

Seminar im Wintersemester 95/96

Georg Bol[*], *Christoph Breitner*[†], *Jutta Mülle*[†] *Jörg Schlösser*[†] *(Hrsg.)*

[*]Institut für
Statistik & Mathematische Wirtschaftstheorie
Fakultät für Wirtschaftswissenschaften
Universität Karlsruhe
[*gbo*]@vwl3sun1.wiwi.uni-karlsruhe.de

[†]Institut für
Programmstrukturen & Datenorganisation
Fakultät für Informatik
Universität Karlsruhe
[*breitner*|*joerg*|*muelle*]@ira.uka.de

Februar 1996

Vorwort

In den vergangenen zehn Jahren hat die Flut von Daten, die in kommerziellen Datenbanken verwaltet wird, immer stärker zugenommen. Gleichzeitig werden Datenbanken zur Unterstützung immer komplexerer Anwendungen eingesetzt, bei denen oft nur ein geringes Wissen über die Zusammenhänge der Daten vorhanden ist.

Trotz der großen Informationsmenge ist der daraus resultierende Nutzen relativ gering und es stellt sich daher die Frage, wie weiteres nützliches, aber verborgenes Wissen effizient aus den existierenden Daten gewonnen werden kann. Mit diesen Fragestellungen befaßt sich die Forschungsrichtung KDD (Knowledge Discovery in Databases).

In diesem Seminar wurden von den teilnehmenden Studenten verschiedene Ansätze für und Einsatzmöglichkeiten von KDD vorgestellt. Die jeweiligen Ausarbeitungen sind in diesem Bericht kapitelweise zusammengestellt. Beginnend mit einer Übersicht über und Einführung in KDD (Kapitel 1) werden spezifische Verfahren und Ansätze zur Wissensgewinnung in den Kapiteln 2, 3, 4, 5 und 6 vorgestellt. Abgerundet werden diese eher den Grundlagen zuzuordnenden Kapitel durch die Kapitel 7 und 8, in denen zwei Anwendungen des KDD in der Praxis vorgestellt werden.

An dieser Stelle möchten wir Betreuer uns bei den teilnehmenden Studenten Jens Nimis, Ulrike Zintz, Dierk König, Ralph Groß, Wolfgang Decker, Sven Burk, Ingo Redeke und Renate Ziegler recht herzlich für ihr Engagement bedanken, welches ausschlaggebend für das gute Gelingen des Seminars war. Des weiteren sei auch Prof. Nakheizadeh gedankt. Ohne seine Hilfe hätte uns wohl nicht so schnell das erst kurz vor dem Seminar veröffentlichte Buch [13] zur Verfügung gestanden, das Ausgangspunkt für viele der Seminarthemen war.

Jörg Schlösser
Jutta Mülle
Christoph Breitner
Georg Bol

Inhaltsverzeichnis

Kapitel 1

Einführung in die Methoden der Wissensgewinnung

Jens Nimis

Kurzfassung *Das rasche Wachstum von Datenbasen in den letzten Jahren hat mittlerweile die menschliche Fähigkeit, die Daten zu interpretieren und zu verarbeiten, überschritten. Dies macht eine neue Generation von Werkzeugen zur automatischen intelligenten Datenbankanalyse unumgänglich. Die Werkzeuge und Techniken sind Gegenstand des in der jüngsten Zeit entstandenen Feldes des KDD (**K**nowledge **D**iscovery **D**atabases). Dieses Kapitel soll neben einigen grundlegenden Defintionen Einblicke in den Ablauf eines KDD-Prozesses geben. Es wird sich zeigen, daß dieser aus mehr besteht, als nur den Data-Mining Methodenn. Insbesondere wird dabei auf die wichtige Rolle des Menschen in diesem Prozeß eingegangen.*

1.1 Warum braucht man KDD?

In den letzten 10 Jahren ließ sich eine Entwicklung beobachten, die KDD (**K**nowledge **D**iscovery in **D**atabases) – Wissensgewinnung in Datenbasen – unumgänglich macht, nämlich die rasch wachsende Fähigkeit, Daten zu erzeugen und anzusammeln. Als Beispiele seien hier die weit verbreitete Einführung der Strichcodes (sog. *barcodes*) für nahezu alle kommerziellen Produkte, sowie im Umfeld der Wissenschaft die Weiterentwicklung im Bereich der Sensortechnik für Satelliten genannt. Viele Verwaltungsaufgaben in kleinen und großen Unternehmen (z.B. Banken, Versicherungen und im Handel) aber auch auf staatlicher Seite werden zunehmend durch den Einsatz von EDV-Mitteln gelöst. Welche Ausmaße diese Entwicklungen angenommen haben, zeigen die folgenden Beispiele für heutige Datenbasen. Um ein Gefühl für ihre Größe zu bekommen, überlege man sich bevor man weiterliest, was man sich bisher auf diesem Gebiet unter „groß“ vorgestellt hat.
In der Wirtschaft verwendete Datenbasen:

- Eine der größten Datenbasen der Welt wird von der nordamerikanischen Einzelhandelskette Wal-Mart betrieben. Sie muß über **20 Millionen Transaktionen pro Tag** verarbeiten.

- Mobil Oil entwickelt gerade eine Datenbank mit der Fähigkeit, mehr als **100 Terabyte** Daten (das entspricht der Speicherkapazität von 100.000 Festplatten der Größe 1 Gigabyte) über Ölausbeutung aufzunehmen. Die darin enthaltenen Informationen sollen dabei zur Vermarktung freigegeben werden.

Beispiele aus der Wissenschaft:

- Das „Human Genome Database Project" hat schon **mehrere Gigabyte** Daten über den menschlichen genetischen Code gesammelt und ist bei dieser Aufgabe noch nicht am Ende angelangt.
- Das EOS (**E**arth **O**bserving **S**ystem) der NASA besteht aus Satelliten und anderen Raumfahrtinstrumenten und soll Ende dieses Jahrzehnts und zu Beginn des nächsten mittels Funkübertragung Daten im Umfang von **50 Gigabytes pro Stunde** zur Erde liefern.

Daß man solch umfangreichen Datenmengen nicht mehr mit gewöhnlichen Methoden – quasi „von Hand" – sinnvoll zu Leibe rücken kann, versteht sich beinahe von selbst. So entstand ein Bedarf an intelligenter, automatischer Unterstützung bei der Datenanalyse, um aus Bergen von Daten die Goldnuggets wertvollen Wissens herauszuholen.

1.2 Geschichte der KDD

1.2.1 Entwicklung der KDD in den letzten Jahren

Das gestiegene Interesse an KDD zeigt sich auch an der steigenden Zahl der Workshops, die in den letzten Jahren zu diesem Thema stattfanden [44, 45, 57, 14]. 1995 mündete diese Entwicklung in die *„First International Conference on Knowledge Discovery and Data Mining"* [15].
Daneben sind auf dem Gebiet zahlreiche Publikationen erschienen. Diese dokumentieren einen Teil der bereits bestehenden Applikationen, ebenso wie die zahlreichen theoretischen Ansätze. Dabei ist die erste Sammlung über Knowledge Discovery in Databases ([46]) besonders hervorzuheben.

1.2.2 Einordnung in andere Wissenschaftsfelder

Ursprünglich gab es für das Auffinden von Mustern (den *„nuggets of knowledge"*) viele verschiedene Bezeichnungen, wie z.B. *knowledge discovery in databases, data mining, knowledge extraction, information discovery, information harvesting, data archaeology* oder *data pattern processing*. Der Begriff Data Mining wurde dabei von Statistikern, Datenanalytikern und der MIS-Gemeinde (**M**anagement **I**nformation **S**ystem) benutzt, während der Ausdruck KDD häufig von den Vertretern der künstlichen Intelligenz und des Maschinenlernens gebraucht wurde. Neuerdings hat eine gewisse Bedeutungsverschiebung eingesetzt, die die Unterschiede zwischen Data Mining und KDD betont. Der Begriff KDD wird

als Oberbegriff des Data Mining behandelt: Data Mining beschreibt nur die eigentliche Mustererkennung in den Daten, KDD hingegen den gesamten Prozeß, wie z.B. die Erarbeitung von Vorwissen, die vorbereitenden Arbeiten und die Auswertung der gewonnenen Muster.

Hier lassen sich schon die ersten Forschungsgebiete erkennen, für die KDD von Interesse ist. Das Ziel, das sie vereint, ist die Extraktion von Wissen aus großen Datenbasen. Maschinelles Lernen und Mustererkennung ist mit KDD durch das Studium von Algorithmen zum Finden von Mustern in Datenbasen verbunden. Aus der Statistik entleiht sich das KDD z.B. Methoden, um mit „Rauschen“ in den Daten umzugehen, oder statistische Prozeduren zur Datenmodellierung. Machine Discovery beschäftigt sich mit der Entdeckung empirischer Gesetze aus Beobachtungs- und Experimentenergebnissen, die Modellierung kausaler Abhängigkeiten mit der Ableitung kausaler Modelle aus Daten. Auf kommerzieller Seite gibt es die Entwicklung des Data Warehousing, wo die zu vermarktenden Daten erst einmal aus anderen gewonnen werden müssen. Eng mit dem Begriff des Data Warehousing ist das bzgl. der Mächtigkeit SQL überlegene **O**n-**L**ine **A**nalytical **P**rocessing (OLAP) als bekannter Ansatz der Analyse der Data-Warehouse-Daten verbunden.

1.3 Was ist KDD?

Nachdem nun Notwendigkeit und Herkunft der KDD bestimmt sind und die vergleichsweise kurze Geschichte geschildert wurde, werden nun die grundlegenden Begriffe anhand ihrer Definitionen eingeführt. Zuerst werden jedoch zwei Beispiele gegeben, an denen sich später einige der Definitionen anschaulich darstellen lassen.

1.3.1 Zwei einführende Beispiele für KDD

Beide Beispiele stammen aus dem Bereich der Entscheidungsfindung in Unternehmen:

Kreditvergabe: Die folgende Abbildung 1.1 zeigt einen einfachen zweidimensionalen Datensatz, bestehend aus 23 Fällen. Jeder Punkt der Grafik stellt eine Person dar, die von einer bestimmten Bank einen Kredit erhalten hat. Dabei sind auf der horizontalen Achse das Einkommen der Person und auf der vertikalen Achse die Höhe des Kredites aufgetragen. Ein x steht dabei für die Personen, die ihre Rückzahlungen nicht ordnungsgemäß geleistet haben, ein o für eine Person, die immer vereinbahrungsgemäß bezahlt hat.

Nun könnte man zu Recht annehmen, daß sich aus den Daten Aufschlüsse über die sinnvolle zukünftige Vergabe von Krediten gewinnen ließe. KDD heißt hier die Lösung.

Dabei ist zu beachten, daß reale KDD-Anwendungen mit Datenbasen in anderen Größenordnungen zurechtkommen müssen (mehrere hundert Dimensionen und Millionen von Datensätzen sind hier keine Seltenheit).

Vermarktungsstrategie: Ein anderes Beispiel stammt aus dem Bereich des Einzelhandels. Hier könnte die Fragestellung in der Werbeabteilung lauten: „Wie macht man

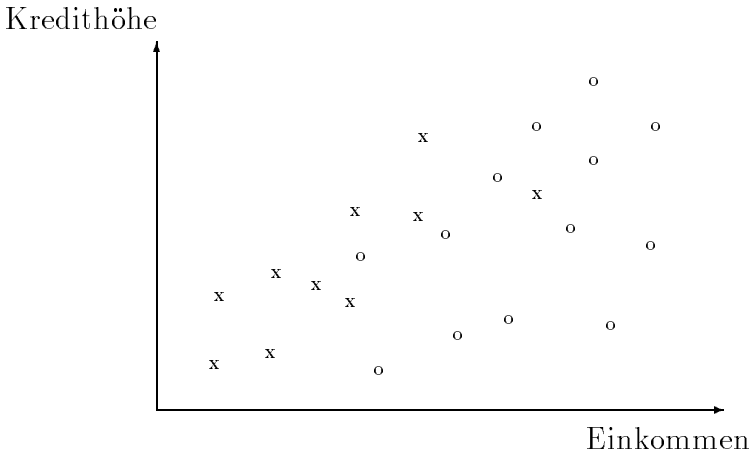


Abbildung 1.1: Kreditszenarium

eine erfolgreiche Weihnachts-Werbekampagne?". Dazu betrachtet der Datenanalytiker die Verkaufszahlen seiner Produkte in der Weihnachtszeit in einer bestimmten Woche im Vorfeld der Plazierung der Werbung und in der Woche danach. Außerdem sucht man nach Gemeinsamkeiten in der Charakteristik (z.B. Preiskategorie, Marke, Art) der allgemein zu dieser Zeit verkauften Produkte. Ein anderer interessanter Parameter ist die Art der Werbung, wie z.B. Preisnachlässe, Aushänge in den Geschäften selbst, Anzeigenkampagnen usw.

Wie diese Aufgaben mit dem Bereich des KDD zu zusammenhängen, zeigen die nun folgenden theoretischen Definitionen.

1.3.2 Definitionen

Die nun folgende Definition ist aus [21]. Sie lautet im Original:

> **Knowledge Discovery in Databases** is the non-trivial process of identifying valid, novel, potentially useful and ultimately understandable patterns in data.

Die einzelnen Begriffe werden jetzt genauer beleuchtet:

Data: (Daten) ist eine Menge von Tatsachen F (facts). Im ersten Beispiel eine Sammlung von 23 Kreditfällen mit den drei Feldern: Höhe des Kredits, Einkommen und Kreditstatus.

Pattern: (Muster) ist ein Ausdruck E (Expression) in einer Sprache L (Language), der eine Untermenge F_E von F beschreibt und dabei (in gewissem Sinne) einfacher ist als deren direkte Aufzählung. Abbildung 1.2 zeigt das Muster „Wenn das Einkommen kleiner als n ist, erfüllt die Person nicht die Rückzahlungen des Kredites", bei geeigneter Wahl von n.

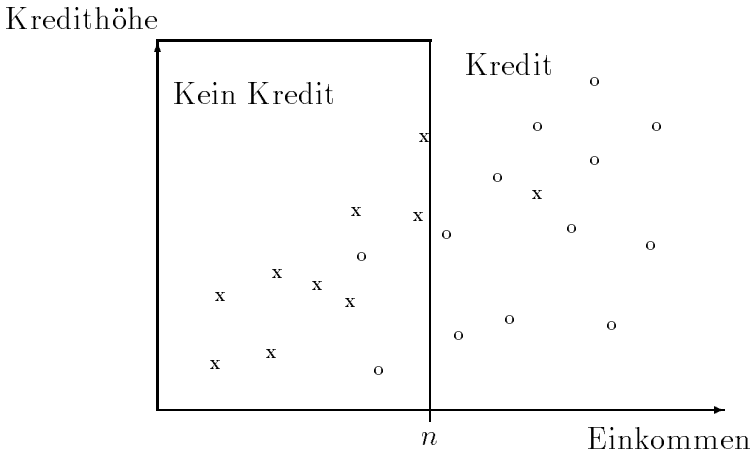


Abbildung 1.2: Verwendung der Einkommensvariablen zur Einteilung der Datensätze

Process: (Prozeß) Für gewöhnlich ist KDD ein mehrstufiger Prozeß, der Datenvorbereitung, Mustersuche, Wissensauswertung und Iterationsschleifen beinhaltet. Der Prozeß wird als nicht-trivial angesehen, womit z.B. die Durchschnittsberechnung der Einkommen nicht als typische KDD-Aufgabe angesehen wird.

Validity: (Gültigkeit) Das entdeckte Muster sollte mit bestimmter Gewißheit auch für neue Datensätze gültig sein. Maß für die Gewißheit ist eine Funktion C (Certainty), die Audrücke aus L auf einen partiell- oder totalgeordneten Meßraum M_C abbildet. So kann einem Ausdruck E aus L über eine Untermenge $F_E \subset F$ eine Maßzahl $c = C(E, F)$ zugeordnet werden. Wenn man z.B. die Begrenzung in Abbildung 1.2 nach links verschiebt, würde die Gewißheit abnehmen, weil mehr Kredite in schlechtem Zustand in die „Kredit-Hälfte“ fallen würden.

Novel: (Neuartigkeit) Das entdeckte Muster sollte (zumindest für das System) neuartig sein. Die Neuartigkeit kann gemessen werden durch Vergleich mit alten oder zu erwartenden Daten oder durch Unterschiede zu bereits vorhandenem Wissen und entdeckten Mustern. Im allgemeinen läßt sich die Neuartigkeit mit einer Funktion $N(E, F)$ messen, die eine boolsche Funktion oder eine Maßzahl für den Grad der Unerwartetheit sein kann.

Potentially Useful: (Potentielle Nützlichkeit) Das entdeckte Muster sollte zu einer nützlichen Aktion führen können. Wenn man sich bei der Kreditvergabe aus Beispiel 1 auf das gefundene Muster stützen würde, dann könnte ein höherer Profit erzielt werden. Meßbar wird der Nutzen durch eine Funktion U von L in einem Meßbereich M_U, wobei $u = U(E, F)$. Im Beispiel könnte u die Summe sein, die die Bank durch Entscheidungsfindung nach der Abbildung zusätzlich verdient.

Understandable: (Verständlichkeit) Ein Ziel der KDD ist es, die gefundenen Muster für den Menschen verständlich zu machen und damit ein besseres Verständnis der zugrundeliegenden Daten zu ermöglichen. Die Verständlichkeit ist schwer zu messen. Ansätze dazu reichen von sehr schwammigen Definitionen wie „für Menschen leicht

verständlich“ bis zu quantitativen Aussagen wie „Länge des Musters in bits“. Wir nehmen an, die Verständlichkeit sei durch eine Funktion S meßbar, die ein $E \in L$ nach M_S abbildet, wobei $s = S(E, F)$.

Ein wichtiger Ausdruck in diesem Zusammenhang ist die **Interessantheit** (interestingness) als umfassendes Maß. In manchen KDD Systemen gibt es daher ausdrücklich eine interestingness function $i = I(E, F, C, N, U, S)$, die einen Ausdruck $E \in L$ nach M_I abbildet. Andere Systeme lösen das Problem indirekt durch eine Ordnung über die gefundenen Muster.

Mit den gegebenen Definitionen im Hinterkopf kann man nun versuchen, Wissen (knowledge) aus dem engen Blickwinkel des KDD neu zu betrachten. (Einen Versuch, Wissen neu zu definieren, sollte man wohl eher einem Philosophen überlassen.)
Wissen: Ein Muster $E \in L$ wird Wissen genannt, falls für einen nutzerspezifischen Schwellenwert $i \in M_I$ gilt: $I(E, F, C, N, U, S) > i$. Dies ist allerdings nur eine Möglichkeit, Wissen zu definieren. Zum Beispiel könnte man auch je nach Schwerpunkt bestimmte Schwellenwerte für c, s, oder u bevorzugen. Dies ist letzten Endes eine Entscheidung, die jeder Benutzer für sich selbst treffen muß.

Data-Mining ist ein Schritt im KDD-Prozeß, der aus einem bestimmten Data-Mining Algorithmus besteht und unter bestimmten Rahmenbedingungen eine Folge von Mustern erzeugt. Dies soll in Abschnitt 1.4 vertieft werden.

Der **KDD-Prozeß** ist der Prozeß der Anwendung von Data-Mining Methoden (Algorithmen), um nach bestimmten Richtlinien und Bewertungskriterien Wissen aus einer Datenbasis F zu gewinnen, mit allen dazu benötigten Vorbereitungen und Nacharbeiten.

Alle bisherigen Definitionen sehen das KDD als einen autonomen Prozeß an, was auf lange Sicht auch wünschenswert ist. Im Moment jedoch spielt der Mensch (noch) eine wesentliche Rolle (wenn nicht sogar die Schlüsselrolle) in diesem Prozeß. Daran sollte man bei den folgenden Betrachtungen denken. Alles in allem ist der KDD-Prozeß ein mensch- und prozeßorientierter Vorgang, der dem Analytiker dabei helfen sollte, Wissensgewinnuung zu verstehen und ihn dabei zu unterstützen, da KDD ohne ihn (im Moment noch) nicht möglich wäre.

1.3.3 Aufbau des KDD-Prozesses

Nachdem wir jetzt gesehen haben, was KDD ist, folgt eine Erklärung, wie der KDD-Prozeß abläuft. Er ist interaktiv und iterativ und beinhaltet zahlreiche Schritte, zwischen denen viele Entscheidungen vom Benutzer getroffen werden müssen. Die folgende Aufzählung nennt die Grundschritte. (Siehe Abbildung 1.3.)

1. Vorwissen: Zuerst muß ein Verständnis für das Anwendungsgebiet, das relevante Vorwissen und die Ziele des Endnutzers erarbeitet werden. Gerade das Finden des eigentlichen Ziels (task discovery) ist oft ein schwieriger und zeitaufwendiger Prozeß. Der Kunde wird das Ziel meist als klar und präzise formuliert ansehen, während es für den KDD-Prozeß vielleicht zuerst noch präziser formuliert werden muß. Trotz

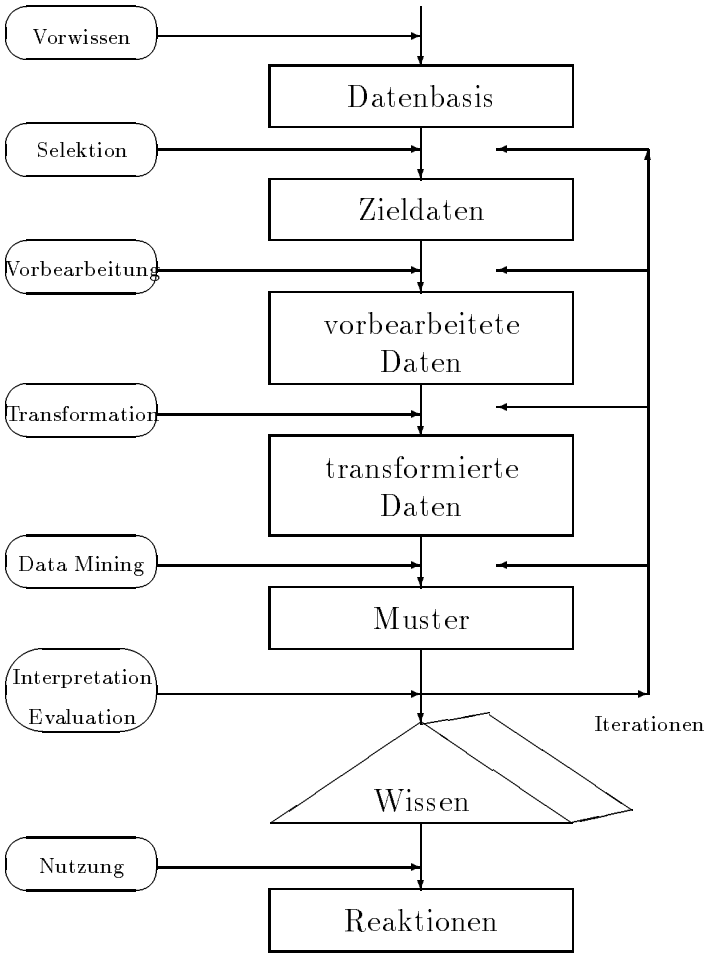


Abbildung 1.3: Der KDD-Prozeß

des Zeitaufwandes sollte man diese Teilaufgabe sorgfältig erledigen, denn sonst passiert es leicht, daß man später viel Zeit damit verbringt, die falschen Fragen zu beantworten. Außerdem ist es leicht einzusehen, wie schwierig es sein kann, sich zwecks Gewinnung von Hintergrundwissen in ein fremdes Anwendungsgebiet ausreichend tief einzuarbeiten. Eine Möglichkeit der Gewinnung von Hintergrundwissen ist auch die Nutzung von Metadaten aus dem DBMS. Manche Systeme (z.B. ReMind) können explizit genanntes Hintergrundwissen algorithmisch nutzen. Dazu ist dann noch eine Umformung in eine bestimmte Darstellung nötig.

2. Selektion: Als nächstes wählt man aus einer Unmenge „roher" Daten die aus, die am geeignetsten für die Untersuchung erscheinen. Dazu gehören Betrachtungen über Struktur und Qualität der Daten und darüber, was sie beinhalten (und vor allem was nicht). Stellt man etwa in den Daten zum Weihnachtsbeispiel fest, daß es keine Information über den Effekt von Werbung in der Weihnachtszeit gibt, so ist das Ziel dort falsch formuliert und muß neu gestellt werden.

3. Datenbereinigung: Hat man seine Zieldaten gewählt, fährt man damit fort, diese zu bereinigen. Oft sind die Kundendaten in recht unorganisierter Weise gesammelt worden, sind von falschen Eintragungen durchsetzt oder manche Felder sind ganz leer. Deshalb müssen „Rauschen" herausgefiltert und Ausreißer erkannt werden (doch Vorsicht: manchmal verbirgt sich gerade hinter einem vermeindlichen Ausreißer ein besonders interessanter Datensatz). Für die leeren Datenfelder müssen Ersetzungsstrategien gefunden werden.

   Ein Ansatz zur Datenbereiningung wird beispielweise im Kapitel 4 dieses Seminarberichtes vorgestellt.

   Der beste Ansatz zur Fehlerbearbeitung ist natürlich die Fehlervermeidung. Das läßt sich oft bewerkstelligen durch die Automatisierung der Datenerfassung. In unserem Einzelhandelsbeispiel könnte dies durch den Einsatz von Scannerkassen geschehen, der eine manuelle Aufzeichnung der Verkaufszahlen unnötig machen würde.

4. Transformation: Die nun „sauberen" Daten werden durch Reduktion und Projektion abgespeckt. Durch Dimensionsverminderung und den Einsatz von Transformationsmethoden sinkt die Anzahl der auszuwertenden Variablen. Dabei muß jedoch das Ziel des Prozesses im Auge behalten werden.

5. Data-Mining-Task Auswahl: Hiermit ist die Vorbereitung der Daten abgeschlossen. Nun wählt man zunächst die Data-Mining Aufgabe, d.h. man muß entscheiden, ob Klassifizierung, Regression, Clusterung oder etwas anderes das Ziel des KDD-Prozesses ist. Die verschiedenen Arten der Data-Mining Aufgaben werden später ausführlich beschrieben.

6. Algorithmenauswahl: Jetzt wählt man den bzw. die Data-Mining Algorithmen. Das beinhaltet die Entscheidung, welche Modelle und Parameter dem Problem angemessen sind. Außerdem muß die Data-Mining Methode zu dem Gesamtumfeld des KDD-Prozesses passen (es könnte dem Auftraggeber z.B. wichtiger sein, ein leicht verständliches Modell zu haben als eines mit besonders hoher Vorhersagefähigkeit).

7. Anwendung: Nun folgt der eigentliche Data-Mining Schritt, d.h. die Suche nach interessanten Darstellungen, wie z.B. Klassifizierungsregeln, Entscheidungsbäumen, einer Clusterung usw.. Diesen Schritt kann der Nutzer erheblich unterstützen, indem er die vorangehenden Schritte korrekt ausführt.

8. Interpretation: Die so gewonnenen Muster müssen jetzt interpretiert und bewertet werden. Möglicherweise ist die Rückkehr zu einem der Punkte 1 bis 7 erforderlich, um so eine weitere Iteration zu starten.

9. Nutzung: Den abschließenden Schritt bildet die Festigung des gewonnenen Wissens. Dazu gehören der Vergleich mit schon vorhandenen Erkenntnissen und das Erkennen potentieller Konflikte. Außerdem muß das neue Wissen dokumentiert und in geeigneter Weise den interessierten Stellen unterbreitet werden. Die Form des Output kann dabei recht unterschiedlich sein. Er kann als Text oder Grafik erfolgen. Denkbar wären aber auch das direkte Auslösen von Aktionen (z.B. bei elektronischen Börsenmakler-Systemen) oder das Einfügen von Monitoren (Alarmgebern bei bestimmten Ereignissen) in Datenbanken. (Dies leistet z.B. das System IMACS).

Abbildung 1.3 zeigt nocheinmal grafisch den Ablauf, wobei die Punkte 5 bis 7 zusammengefaßt wurden. Neben den schon genannten Positionen können an beinahe allen anderen Stellen Schleifen auftreten. Auch wenn jeder einzelne Schritt von großer Bedeutung für einen erfolgreichen KDD-Prozeß ist, so fand bisher der siebte Schritt die größte Beachtung. Ihm ist daher auch ein eigener Abschnitt (1.4) gewidmet.

1.3.4 Integration der Aufgaben

Der oben gezeigte Prozeß stellt eine Verkettung von relativ einfachen Unterprozessen dar, die jedoch selbst recht verschlungen und kompliziert sein kann. Da der Prozeß dabei an fast allen Stellen rückgekoppelt sein kann, ist es erforderlich, daß die Ausgabe einer Stufe als Eingabe bestimmter anderer Stufen dienen kann (insbesondere der nachfolgenden Stufe).
Außerdem sollte es möglich sein, Ergebnisse von Durchläufen, die schon längere Zeit zurückliegen (etwa bei zeitlich veränderlichen Daten, die periodisch untersucht werden), in den neuesten Durchlauf mit einzubeziehen. Dies und der Wunsch nach einer geschickten Benutzerführung erklärt die Forderung nach einer integrierenden Arbeitsumgebung mit einer intelligenten Buchhaltungsplattform. Die Wichtigkeit dieser Arbeitsumgebung sollte (vor allem im Bezug auf den kommerziellen Erfolg) nicht unterschätzt werden.

1.4 Data-Mining Methoden

Wie schon angesprochen, bringt der KDD-Prozeß oft den iterativen Einsatz von bestimmten Data Mining-Methoden mit sich. Zur Beschreibung dieser Methoden ist zunächst die Klärung von zwei Begriffen nötig: Muster (patterns) und Modelle (models). Ein Muster kann man sich vorstellen als Instanzierung eines Modells (z.B. ist $f(x) = 3x^2 + x$ ein Muster, wohingegen wir $f(x) = ax^2 + bx$ als Modell betrachten). Data Mining kann man dann

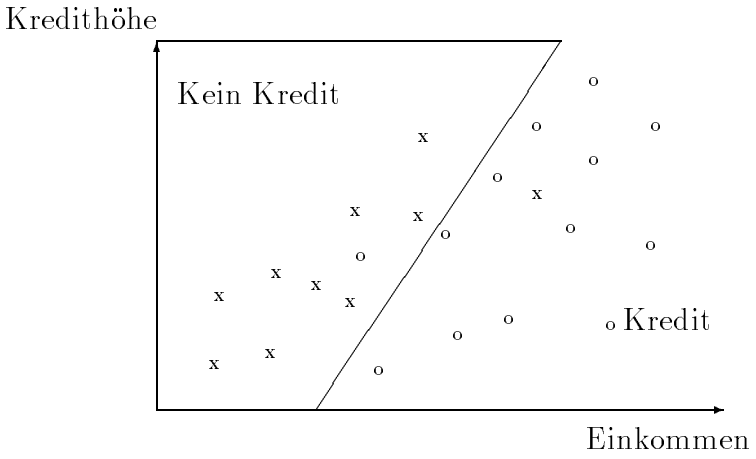


Abbildung 1.4: Klassifikation durch ziehen einer linearen Grenze

interpretieren als den Vorgang, ein Modell anzupassen oder ein Muster aus bestehenden Daten zu extrahieren. Die Entscheidung, ob das Ergebnis nützliches oder interessantes Wissen darstellt, ist eine Aufgabe des Gesamtprozesses, wozu häufig subjektive menschliche Beurteilung gebraucht wird.

1.4.1 Hauptaufgaben des DM

Die übergeordneten Ziele des Data Mining sind Vorhersage (prediction) und Beschreibung (description). Bei der Vorhersage werden Variablen oder Felder der Datenbasis dazu benutzt, unbekannte oder zukünftige Werte von interessanten Variablen zu finden. Die Beschreibung beschäftigt sich mit dem Aufspüren von Mustern, die vom Menschen interpretierbar sind. Diese beiden Ziele haben bei verschiedenen Anwendungen unterschiedliche Wichtigkeit. Sie sind zum Teil recht gegensätzlich: genauere Vorhersage geht oft nur auf Kosten der Verständlichkeit des Musters.
Die folgende Aufzählung nennt die wichtigsten Vorgehensweisen bei der Erreichung des Ziels:

Klassifikation: (Classification) Darunter versteht man das Erlernen einer Funktion, die einen Datensatz einer Klasse aus einer vordefinierten Klasseneinteilung zuordnet. Ein Anwendungsgebiet dieser Methode ist die Einordnung von Trends auf Finanzmärkten.
Abbildung 1.4 zeigt die lineare Unterteilung unseres ersten Beispiels in 2 Klassenregionen, dabei ist leicht zu erkennen, daß es auf diese Weise nicht möglich sein wird, die Klassen perfekt festzulegen.

Regression: (Regression) ist das Erlernen einer Funktion, die einen Datensatz auf eine reellwertige Vorhersagevariable abbildet. Diese Methode wird zum Beispiel in der Medizin angewandt, um die Überlebenswahrscheinlichkeit eines Patienten aufgrund bestimmter Diagnosedaten zu bestimmen.

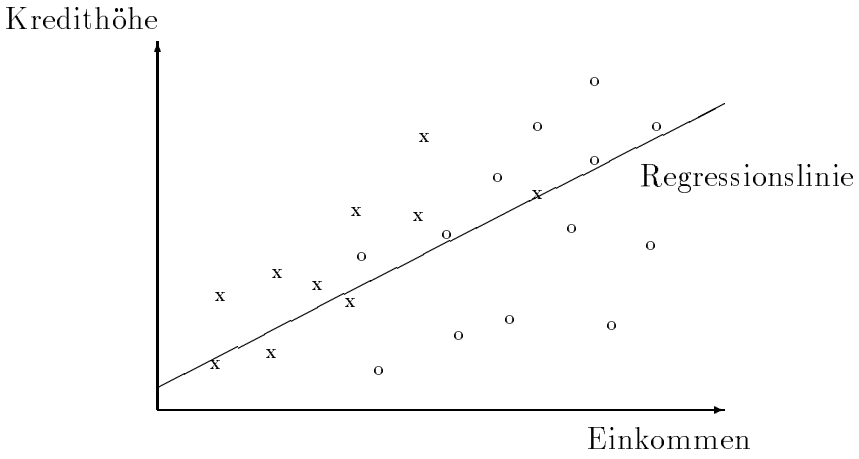


Abbildung 1.5: Einfache lineare Regression

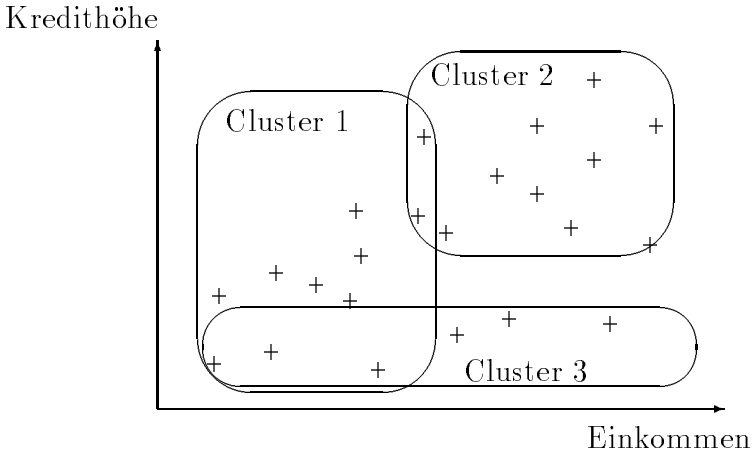


Abbildung 1.6: Clusterung im Kreditbeispiel

Abbildung 1.5 zeigt des Ergebnis einer Regression, wobei die „Kredithöhe“ als lineare Funktion des „Einkommens“ angenommen wurde: Die Qualität dieser Vorhersage ist schlecht, da die Korrelation zwischen den Variablen nur schwach ist.

Clusterung: (Clustering) Hier wird versucht, eine endliche Menge von Kategorien (Clustern) zu finden. Diese können gegenseitig ausschließend und erschöpfend, aber auch hierarchisch gegliedert oder überlappend sein.
Abbildung 1.6 zeigt eine mögliche Clusterung der Kreditdaten in 3 Cluster. Dabei ist zu beachten, daß bestimmte Datensätze in mehreren Clustern zu liegen kommen können und die x und o durch + ersetzt werden, was bedeutet, daß die bisherige Klassenzugehörigkeit keine Bedeutung mehr hat.

Aggregierung: (Summarization) Darunter versteht man Methoden zur kompakten Beschreibung einer Untermenge der Daten. So findet eine Aggregierung der Information statt. Ein einfaches Beispiel wäre die Anordnung nach dem Durchschnitt oder der

Standardabweichung über alle Felder. Solche Methoden werden oft bei der interaktiven Datenanalyse verwendet.

Modellierung von Abhängigkeiten: (Dependency Modeling) nennt man das Auffinden eines Modells, das signifikante Abhängigkeiten zwischen Variablen beschreibt. Es gibt diese Abhängigkeitsmodelle auf zwei Ebenen. Auf der strukturellen Ebene werden lokale Abhängigkeiten beschrieben, wohingegen die quantitative Ebene des Modells die Stärke der Abhängigkeit spezifiziert. Ein Anwendungsgebiet dieser Methode ist die Modellierung menschlicher Gene.

Entdeckung von Veränderungen und Abweichungen: (Change and Derivation Detection) Sie beschäftigt sich mit den auffälligsten Veränderungen in den Daten im Vergleich zu vorher gemessenen oder durchschnittlichen Werten.

1.4.2 Grundaufbau von DM-Verfahren

Wenn man die Data-Mining Algorithmen miteinander vergleicht, dann kann man bei allen drei wesentliche Komponenten erkennen. Dieser reduzierende Blickwinkel erhebt keinen Anspruch auf Vollständigkeit, aber er erhöht die Handhabbarkeit der Schlüsselkonzepte.

Modelldarstellung: (Model Representation) ist die Sprache L, mit der entdeckbare Muster beschrieben werden. Diese darf nicht zu beschränkt sein, damit bestimmte Zusammenhänge noch erfaßbar bleiben. Andererseits darf sie auch nicht zu umfangreich sein, um Überangepaßtheit (Overfitting) zu verhindern, die zu einer schlechteren Voraussagegenauigkeit führt.

Suchmethode: (Search Method) Sie besteht selbst aus zwei Komponenten, der Parametersuche (Model Fitting, Parameter Search) und der Modellsuche.
Die Parametersuche versucht, die zu den Daten und einem festen Modell besten Parameter zu finden. Dazu werden oft iterative Methoden verwendet.
Die Modellsuche erscheint als Schleife über der Parametersuche. Hier wird zwischen Modellen gewechselt und dann immer die ganze Modellfamilie betrachtet. Die Modellsuche ist der Teilschritt des KDD-Prozesses, bei dem das Hintergrundwissen einfließt, um die Menge der möglichen Modelle einzuschränken.

Modellevaluation: (Model Evaluation) Sie schätzt ab, wie gut ein bestimmtes Muster (also ein Modell mit seinen Parametern) in der Lage ist, die Anforderungen des KDD-Prozesses zu erfüllen. Dazu kann man logische und statistische Vorgehensweisen verwenden. Kriterien dabei sind die in den Definitionen gegebenen Bewertungsfunktionen. Oft werden dabei Trainings- und Testdaten unterschieden. (Z.B. ergibt ein Vorgehen nach dem maximum-likelyhood-Prinzip die Parameter für ein Modell, die am besten zu den Trainingsdaten passen.)

1.4.3 Ausgewählte Beispiele für DM-Verfahren

Die folgende Aufzählung zeigt eine Reihe populärer Data-Mining Methoden:

Entscheidungsbäume und Regeln: Entscheidungsbäume, die univariate Verzweigungen benutzen, haben eine einfache, verständliche Darstellungsform, jedoch nur gerine Mächtigkeit. Abbildung 1.2 zeigt eine solche Aufteilung. An einem Knoten des Entscheidungsbaumes folgen die beiden Äste: Einkommen $> n$ und Einkommen $\leq n$. Die Mächtigkeit wächst, indem man allgemeinere Ausdrücke an den Verzweigungen zuläßt.

Nichtlineare Regression und Klassifikationsmethoden: Darunter versteht man eine Familie von Voraussagetechniken, die lineare und nichtlineare Kombinationen von Basisfunktionen (z.B. Splines, Polynomen usw.) an Kombinationen der Inputvariablen anpaßt. Auf diese Weise könnte man im Kreditbeispiel eine nichtlineare Grenze zwischen den beiden Fällen erzeugen.

Beispielbasierte Methoden: Eine Technik aus diesem Bereich ist die Ermittlung eines nächsten Nachbarn, d.h. man sucht aus den schon vorhandenen Datensätzen den am nächsten liegenden zu dem neuen (quasi am ähnlichsten). In unserem Kreditbeispiel würde dies bedeuten, daß im Kreditbereich eine kleine „kein-Kredit-Insel“ liegt. Ein Problem bei dieser Methode kann eine wohldefinierte Metrik darstellen. Wenn man im Beispiel als dritte Dimension die Kreditlaufzeit hinzufügt, steht man vor dem Problem: „Wieviel Mark entsprechen 3 Monaten?“.

Probabilistische grafische Abhängigkeitsmodelle: In der einfachsten Form spezifiziert ein solches Modell, welche Variablen direkt voneinander abhängen. Typischerweise werden solche Modelle bei diskretwertigen Variablen verwendet, aber Erweiterungen machen sie auch für reellwertige Variablen verwendbar. Die grafische Darstellung führt zu guter Interpretierbarkeit durch den Menschen.

Induktiv logische Programmierung: Während die Darstellungsform der Entscheidungsbäume stark eingeschränkt ist, benutzt ILP (auch bekannt als relationales Lernen, da es häufig im Zusammenhang mit Relationen bzw. dem relationalen Datenmodell auftritt) die mächtigere Prädikatlogik erster Ordnung (mit Gleichheit). Im Gegensatz zu den univariaten Entscheidungsbäumen erkennt ILP so auch leicht Zusammenhänge wie x = y. ILP–Methoden werden im Kapitel 5 vorgestellt.

1.5 Die Realität

In der Geschäftswelt ist die am weitesten verbreitete Anwendung des KDD das „Database Marketing“, eine Methode zur Analysierung von Kunden-Daten, um bestimmte Vorlieben herauszufiltern. Laut *Business Week* beabsichtigen über 50% aller Händler, Database Marketing einzusetzen oder tun dies bereits. Dabei erreichte z.B. *American Express* eine 15 bis 20 prozentige Steigerung beim Kartenabsatz. Andere Anwendungsgebiete sind die Analyse von Finanzmärkten und die Aufdeckung von Betrugsdelikten.

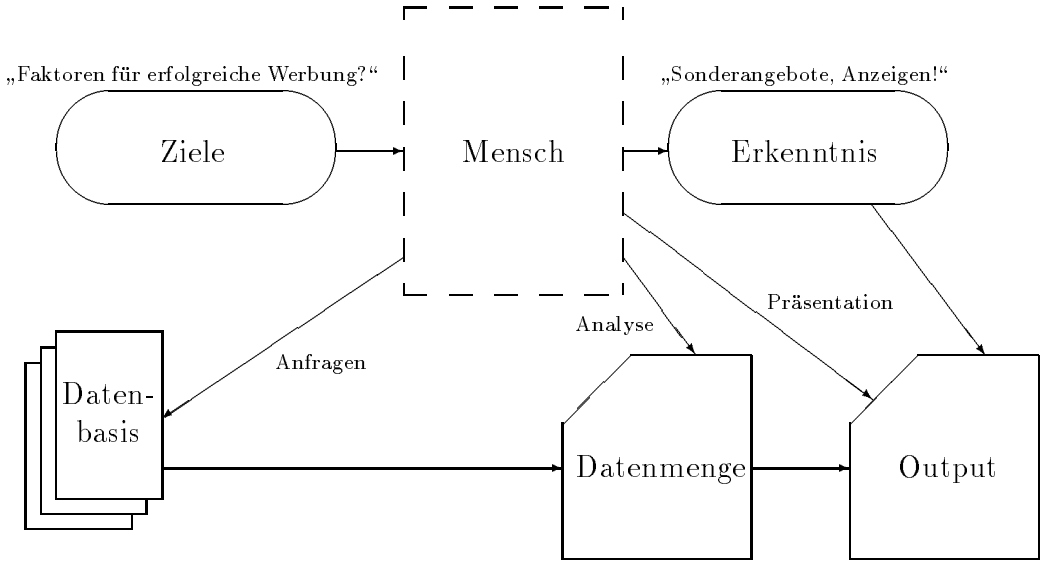


Abbildung 1.7: Rolle des Menschen im KDD-Prozeß

1.5.1 KDD als menschenorientierter Prozeß

Im Gegensatz zu der oben gefundenen Definition läßt sich eine andere Beschreibung des KDD-Prozesses finden, wenn man die realen Abläufe betrachtet:

> Knowledge Discovery ist eine wissensintensive Aufgabe, bestehend aus komplexen Interaktionen mit langwierigem Zeitaufwand zwischen einem Menschen und einer (eventuell großen) Datenbasis, die möglicherweise durch eine Menge von heterogenen Werkzeugen unterstützt wird.

Leider beruhen Neuerungen auf dem Markt der KDD-Systeme meist auf einer oder mehreren neuen Data Mining-Methoden und nicht auf den Bedürfnissen der Datenanalysten. Daher rührt auch häufig der geringe kommerzielle Erfolg der Produkte. Oft ist nicht das Data Mining, sondern die zahlreichen anfallenden Nebenaufgaben des Datenanalysten das Hauptproblem. Die zentrale Rolle des Menschen kommt auch in Abbildung 1.7 zum Ausdruck.

1.5.2 Wer ist der Benutzer?

Wie wir eben gesehen haben, spielt der Mensch (noch) eine wesentliche Rolle im KDD-Prozeß. Nun kann man aber nicht erwarten, daß jeder noch so kleine Betrieb einen Fachmann für Datenanalyse einstellen kann, nur weil er von den Vorzügen der neuen Technik profitieren will.
Dies führt zu folgender Unterscheidung: Entgegen dem allgemeinen Gebrauch ist das Ergebnis der Arbeit mit einem Knowledge Discovery System nicht Wissen, sondern man

trennt zwischen einem KDSE (**K**nowledge **D**iscovery **S**upport **E**nvironment) und einer KDA (**K**nowledge **D**iscovery **A**pplication). Der Datenanalysefachmann arbeitet mit einem KDSE und erhält als Ergebnis eine KDA, ein lauffähiges Programm, das auf dem System der Geschäftsperson installiert und betrieben wird. Diese ist dann für dessen Nutzung mit den neu eingehenden Daten und die Befolgung der vorgeschlagenen Aktionen verantwortlich.

1.5.3 Existierende Applikationen

Folgende Aufzählung nennt einige in der Benutzung befindliche Systeme, ihre Urheber oder Personen, die sich intensiv mit ihnen beschäftigt haben.

Coverstory von IRI: Eine der ältesten Anwendungen, die sich mit der Analyse und Berichterstattung über veränderliche Daten beschäftigt.

Spotlight von A.C. Nielsen: zur Analyse von Supermarktverkaufsdaten.

KEFIR von GTE: für Daten aus dem Gesundheitswesen. (Siehe dazu auch 8)

Nestor FDS: wurde für die Entdeckung von Kreditkartenbetrug konzipiert und überwacht Millionen von Kartenaccounts.

SKICAT von SPL/Caltech: wird zur automatisierten Identifizierung von Sternen und Galaxien und zu ihrer Katalogisierung und Analyse eingesetzt.

Auch in der Molekularbiologie und in der Erforschung der globalen Klimaveränderung ist KDD im Einsatz. Über die in der Betrugsverfolgung verwendeten Systeme ist aus verständlichen Gründen nur wenig publiziert.

1.5.4 Kriterien zur Auswahl eines KDD-Werkzeuges

Die Auswahlkriterien lassen sich in zwei Klassen unterteilen, praktische und technische Kriterien:
Praktische Kriterien:

- Potential des Systems, die gesteckten Ziele zu erreichen. Ziele im Sinne von Unternehmen sind Gewinnzuwachs, Zeitersparnis, Kostenminimierung oder Qualitätssteigerung. Für den wissenschaftlichen Bereich ist eher die Neuartigkeit und Qualität des entdeckten Wissens, sowie verbesserter Zugriff auf Daten durch Automatisierung manueller Analyseprozesse von Bedeutung.
- Fehlende Alternativen sind ein weiteres Kriterium, da vergleichbar gute Ergebnisse oft nur in einer bestimmten Anwendung erzielt werden können.
- Es muß in dem Unternehmen eine Person geben, die sowohl die inhaltliche Bedeutung der Daten kennt, als auch versteht, mit der neuen Technik umzugehen und den KDD-Prozeß zu unterstützen.

Technische Kriterien:

- Für die bestimmte Anwendung müssen genug Daten vorhanden sein. Die Anzahl der benötigten Daten kann zwischen verschiedenen Lösungsansätzen schwanken.
- Es ist zu berücksichtigen, wie relevant die vorhandenen Daten für das Ziel sind.
- Wie gut kann eine Anwendung mit geringer Datenqualität (Rauschen) umgehen?
- Eine verwandte Überlegung ist, ob es möglich ist, dem extrahierten Wissen ein Konfidenzintervall hinzuzufügen. Dies ermöglicht dem Benutzer die Folgeaktionen zu kalibrieren.
- Eines der wichtigsten Merkmale ist die Fähigkeit, das Vorwissen in ausreichendem Maße in den Prozeß einfließen zu lassen.
- Enge Kopplung zum DBMS (**D**ata **B**ase **M**anagement **S**ystem)
- Gemeinsame Nutzung der Datenstrukturen durch andere Schritte des Prozesses.
- Die Fähigkeit, die beste DM-Methode vorzuschlagen.
- Die Fähigkeit, den besten Graphen für Zusammenhänge auszugeben.
- Der User muß die Zusammenhänge erkennen können.

1.5.5 Datenschutzaspekte

Ein Punkt, der bis jetzt noch nicht angesprochen wurde, ist der Umgang mit sensiblen Daten. Dieses Problem tritt immer dann auf, wenn persönliche, betriebsgeheime oder regierungsgeheime Daten analysiert werden sollen. Oft ist es auch mehr eine ethische als rechtliche Fragestellung.
1990 wollte Lotus eine CD mit Daten aus über 100 Millionen amerikanischen Haushalten auf den Markt bringen. Nach stürmischen Protesten wurde diese Untersuchung verworfen. In Deutschland gibt es mittlerweile eine CD-Rom mit Daten (vor allem Adressen und Telefonnummern) aus 30 Millionen Haushalten.
Die OECD (**O**rganization for **E**conomic **C**ooperation and **D**evelopment) hat Richtlinien herausgebracht, die von der Europäischen Gemeinschaft größtenteils übernommen wurden. Ihre Kernaussage ist: „Verwende die Daten eines lebenden Individuums nur mit dessen Zustimmung!“ Außerdem sollten Daten nur mit Verfolgung eines bestimmten Ziels gesammelt werden, und die Nutzung für andere Aufgaben darf nur mit Genehmigung erfolgen. In den USA werden gerade Richtlinien zum Umgang mit den Daten aus der NII (**N**ational **I**nformation **I**nfrastructure), der sogenannten Datenautobahn entworfen.
Beim KDD ist es oft nicht nötig, Namensinformationen in den Daten zu belassen. Oft genügt die Untersuchung einer bestimmten Gruppenzugehörigkeit zum Erreichen des Ziels. Jedoch läßt sich zum Beispiel in manchen Betrieben aus der Zugehörigkeit zu mehreren bestimmten Gruppen auf das Individuum schließen.

1.6 Die Zukunft - Ein Ausblick

Im letzten Abschnitt werden noch einige der zum Teil schon erwähnten Herausforderungen aufgezeigt, die sich der Forschung im Bereich der KDD in der nächsten Zeit stellen werden.

Größere Datenbasen: Datenbasen mit hunderten von Tabellen und Millionen von Einträgen in Multi-Gigabytegröße sind keine Seltenheit mehr und die ersten Terabytegrößen erscheinen.

Hohe Dimensionalität: Nicht nur die hohe Anzahl der Einträge kann zum Problem werden, sondern auch die Anzahl der Attribute pro Eintrag. Mit ihr wächst nicht nur die Größe des Suchraumes explosionsartig, sondern auch die Gefahr der Entdeckung von Pseudoabhängigkeiten wächst.

Overfitting: Ein häufig auftretendes Problem ist die Überanpassung des entdeckten Musters und die dadurch sinkende Performance bei Testdaten.

Bewertung der statistischen Aussagekraft: Ein Problem, verwandt dem Overfitting, tritt dann auf, wenn ein System aus mehreren verschiedenen Modellen das beste suchen muß. Wo soll die Grenze sein, ein Modell als signifikant anzuerkennen oder es abzulehnen? Also muß eine Teststatistik geführt werden als eine Unterfunktion der Suche.

Temporäre Daten: Sich schnell ändernde Datenwerte stellen ein Problem dar. Daher sind inkrementelle Methoden zur Musterveränderung nötig.

Fehlende und fehlerhafte Daten: Dies Problem ist besonders in Geschäftsdatenbasen akut. Nach US-Studien sind dort bis zu 20% der Datensätze fehlerhaft.

Komplexe Beziehungen zwischen den Feldern: In der Vergangenheit wurden DM-Algorithmen nur für einfache attributwertige Felder entwickelt, doch diese konnten komplexere Zusammenhänge nicht erkennen.

Verständlichkeit der Muster: Dieses Problem ist eng verbunden mit einer guten Ausgabegenerierung (z.B. als Grafik).

Benutzerinteraktion und Einbindung des Vorwissens: Während die Benutzerinteraktion maßgeblich durch die Arbeitsumgebung unterstützt werden kann, ist die Einbindung von Vorwissen ein Problem der Algorithmen.

Integration in andere Systeme: Ein alleinstehendes Wissensgewinnungssystem ist wenig sinnvoll. Eine Anbindung an das DBMS, sowie an Tabellenkalkulations- und Visualisierungswerkzeuge sind wünschenswert.

Kapitel 2

Attributorientierte Induktion, Semantische Optimierung

Ulrike Zintz

Kurzfassung *Die attributorientierte Induktion ist eine Methode zur Wissensgewinnung aus großen Datenbasen. Sie umfaßt die Generalisierung der Attribute einer Relation entlang vorhandener Konzepthierarchien, das Zusammenfassen der verallgemeinerten Attribute zu einer generalisierten Relation und schließlich das Gewinnen von Wissen aus den so erhaltenen Datensätzen. Zusätzlich findet sie Anwendung im Bereich der semantischen Anfrageoptimierung, indem aus komplexen Anfragen, Regeln konstruiert werden. Aufgrund dieses Wissens, können ähnliche Anfragen mit wesentlich kleineren Kosten bearbeitet werden, was generell zu einem besseren Antwortverhalten führt.*

2.1 Attributorientierte Induktion

2.1.1 Einleitung

In den letzten Jahren hat die Flut von Daten, die in kommerziellen Datenbanken verwaltet werden, immer stärker zugenommen. Gleichzeitig werden die Datenbanken zur Unterstützung immer komplexerer Anwendungen eingesetzt, bei denen oft nur ein geringes Wissen über die Zusammenhänge vorhanden ist. Daher ist eine der wichtigsten Fragestellungen, wie aus den vorhandenden Daten verborgenes Wissen möglichst effizient gewonnen werden kann.
Die Methode der attributorientierten Induktion ist entwickelt worden, um aus einer relativ großen Datenbasis zusätzliche Information zu gewinnen. Sie beruht auf der Technik des *Lernen aus Beispielen* und hat als Ziel das Aufstellen von Regeln für die betrachteten Datensätze.
Einfache Beziehungen zwischen Tupeln verschiedener Relationen können über gemeinsame Attribute mit Hilfe von Join-Operatoren erhalten werden. Es stellt sich daher die Frage, ob die so erhaltenen Relationen übersichtlich genug sind, um aus ihnen brauchbare Information zu gewinnen. Oft werden Attribute in einer Datenbasis geführt, deren

Domänen eine Vielzahl von unterschiedlichen Werten zulassen. Dieses erschwert das Definieren von Regeln für die betreffenden Datensätze, da die Regeln durch die große Anzahl verschiedener Werte des gleichen Attributs unüberschaubar werden.
Dieses waren einige der Gründe, die schließlich zur Entwicklung der Methode der attributorientierten Induktion geführt haben.
Um kompakte aufschlußreiche Regeln aufstellen zu können, wird die attributorientierte Induktion auf eine Menge von Datensätzen angewandt, welche das Ergebnis einer Anfrage des Benutzers ist. Diese Anfrage hat in SQL-Notation folgende Form:

$$\begin{array}{l}\text{select } A_1, A_2, ..., A_n \\ \text{from } R_1, R_2, ..., R_n \\ \text{where Bedingung}\end{array}$$

Dadurch erzielt man eine Beschränkung des Algorithmus auf die für den Benutzer relevanten Tupel. Darauf setzt nun die Induktion auf. Für jedes vorkommende Attribut muß eine Konzepthierarchie gegeben sein, die zur Generalisierung des jeweiligen Attributes notwendig ist. Ein Beispiel einer solchen Konzepthierarchie ist in Abbildung 2.1 dargestellt.

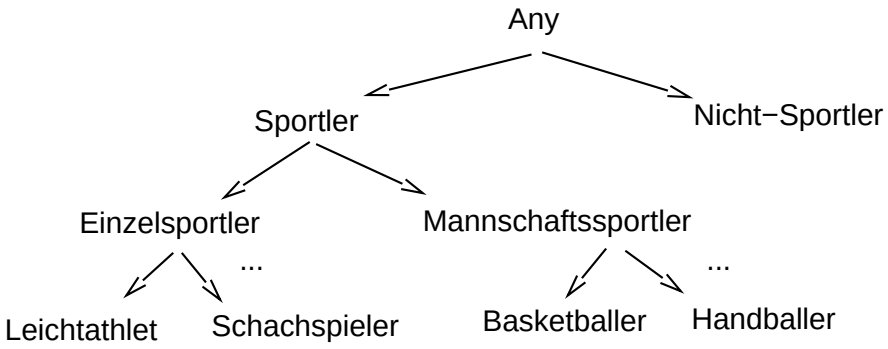


Abbildung 2.1: Graphische Darstellung einer Konzepthierarchie

Ein Tupel einer Relation kann durch ein Konjunktionsglied dargestellt werden. Für die Relation $R(A_1, A_2, ...A_n)$ gilt z.B. $A_1 \wedge A_2 ... \wedge A_n$. Um nun aus einer Menge von Tupeln eine Regel aufzustellen, werden die einzelnen Tupel, in Form von Konjunktionsgliedern, disjunktiv miteinander verknüpft.
Im Abschnitt 1.2 wird ein Algorithmus zur Erzeugung von Regeln für eine bestimmte Klasse von Datensätzen vorgestellt. Abschnitt 1.3 behandelt das Thema der Regelkonstruktion aus einem anderen Blickwinkel, indem ein Algorithmus vorgestellt wird, welcher für eine Menge von Datensätzen Zugehörigkeitsmerkmale in Form von Regeln erzeugt. Am Ende des Kapitels werden die beiden Algorithmen hinsichtlich der Eigenschaften ihrer erzeugten Regeln untersucht und verglichen.

2.1.2 LCHR-Algorithmus

Das Akronym *LCHR* steht hier für *Learning characteristic rules*. Der LCHR-Algorithmus stellt für eine Menge von Datensätzen Regeln auf, die aus der Zugehörigkeit zu dieser

Menge, auch Klasse genannt, resultieren.

LCHR-Algorithmus
Eingabe

- Relationale Datenbank
- Konzepthierarchie für jedes Attribut der Anfrage
- SQL-Anfrage
- Obere Schranke für jedes Attribut

Ausgabe

- Eine charakteristische Regel für die Menge der relevanten Datensätze

Durchführung

1. Ausführung einer SQL-Anfrage
   Im ersten Schritt wird mit Hilfe einer SQL-Anfrage die Menge der zu untersuchenden Daten reduziert. Der Algorithmus beschränkt sich also nur auf die für den Benutzer relevanten Datensätze. Für den weiteren Verlauf des Algorithmus', betrachten wir nun die erhaltenen Daten als zu einer Relation gehörig, auch wenn sie ursprünglich aus mehreren Relationen hervorgegangen sind. Diese *neue* Relation nennen wir *Ausgangsrelation.*

2. Ableitung des Generalisierungsplans für jedes Attribut
   Falls die im ersten Schritt erhaltenen Datensätze Attribute enthalten, über die der Benutzer keine Information erhalten möchte, können diese vom Algorithmus sozusagen ignoriert werden. Dieses geschieht durch eine Projektion auf die Menge der interessierenden Attribute. Als nächster Schritt erfolgt die Generalisierung der einzelnen Attribute. Sie ist notwendig, um die Anzahl der Tupel der Ausgangsrelation zu verringern. Die Generalisierung eines Attributes erfolgt anhand der vorhandenen Konzepthierarchie für das jeweilige Attribut. Oft ist eine obere Schranke, d.h. eine maximale Anzahl verschiedener Werte, für jedes Attribut vorgegeben. In dem Fall muß die Generalisierung soweit fortgeführt werden, bis alle Attribute ihre jeweiligen Schrankenbedingungen erfüllen. Gelangt man in die Situation, daß ein Attribut eine größere Anzahl an Ausprägungen auf der höchsten Ebene besitzt als seine obere Schranke es zuläßt, muß dieses Attribut in einem Generalisierungsschritt durch ANY ersetzt also entfernt werden. Diese Vorgehensweise ist notwendig, um Regeln übersichtlich zu formulieren auch dann, wenn ein Attribut eine Vielzahl von verschiedenen Werten auf der höchsten Konzepthierarchieebene besitzt.

3. Ableitung der Endrelation
   Die attributorientierte Induktion wird also durchgeführt, indem jedes Konzept eines Attributes durch ein höheres Konzept seiner Hierarchie ersetzt wird. Das Verfahren wird fortgesetzt, solange noch Attribute existieren, deren Schrankenbedingung nicht erfüllt ist. Diese Vorgehensweise führt letztendlich zu einer sogenannten *Endrelation*, die weitaus weniger Tupel enthält als unsere Ausgangsrelation.

Beispiel: Wir gehen davon aus, daß in unserer Datenbasis Angaben über Studenten einer Universität gehalten werden. Diese sind in Tabelle 1.1 aufgeführt.
Zusätzlich besitzen wir weitere Informationen über die Attribute unserer Datenbasis, in Form von Konzepthierarchien:

$\{Informatik, Mathematik, Physik, Biologie\} \subset$ Naturwissenschaften
$\{Musik, Geschichte, Philologie\} \subset$ Kunst
$\{1, 2, 3, 4\} \subset$ Vordiplom
$\{5, 6, 7, 8, 9, 10, 11, 12\} \subset$ Hauptdiplom
$\{Burnaby, Vancouver, Victoria, Richmond\} \subset$ British Columbia
$\{Calgary, Edmonton\} \subset$ Alberta
$\{Ottawa, Toronto\} \subset$ Ontario
$\{Bombay\} \subset$ India
$\{Shanghai, Nanjing\} \subset$ China
$\{China, India\} \subset$ foreign
$\{British\ Columbia, Alberta, Ontario\} \subset$ Canada
$\{1.0 \Leftrightarrow 1.5\} \subset$ sehr-gut
$\{1.6 \Leftrightarrow 2.5\} \subset$ gut
$\{2.6 \Leftrightarrow 3.5\} \subset$ befriedigend
$\{3.6 \Leftrightarrow 4.0\} \subset$ ausreichend

Der Übersichtlichkeit wegen, ist in Abbildung 2.2 die Konzepthierarchie des Attributes *Geb-Ort* als Baum dargestellt.

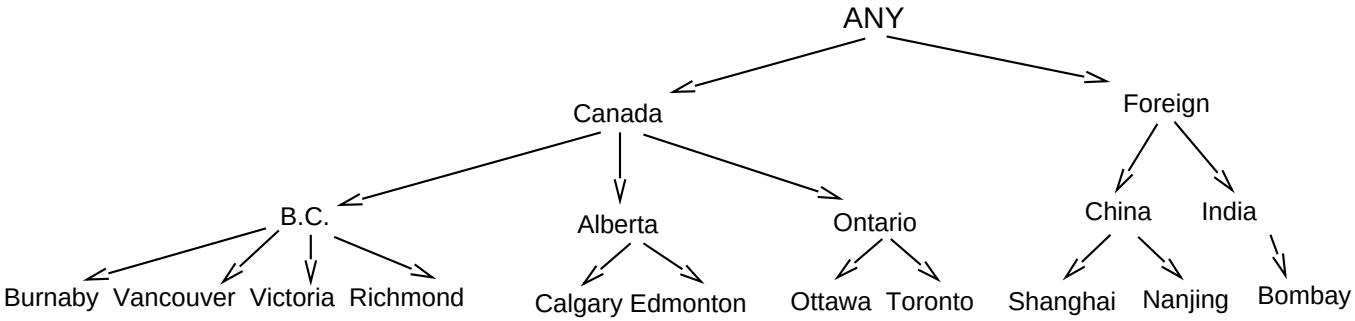


Abbildung 2.2: Konzepthierarchie des Attributes *Geb-Ort*

Wir setzen uns nun als Ziel, mit Hilfe des obigen Algorithmus, Regeln aufzustellen, die sich auf Studenten ab dem 5. Semester beziehen, wobei die relevanten Attribute Fachrichtung, Geb-Ort und Note sind.
Mit Hilfe einer SQL-Anfragen können wir aus der Datenbasis alle Tupel selektieren für die $Semester \geq 5$ gilt. Wir erhalten somit die für den Algorithmus notwendige Ausgangsrelation.

```
select *
from student
semester ≥ 5
```

Name	Semester	Fachrichtung	Geb-Ort	Note
Andersson	7	Geschichte	Vancouver	1.5
Bach	4	Mathematik	Calgary	1.3
Carey	1	Philologie	Edmonton	2.4
Fraser	6	Physik	Ottawa	1.1
Gupta	10	Mathematik	Bombay	1.7
Hart	2	Chemie	Richmond	2.3
Jackson	2	Informatik	Victoria	1.5
Liu	8	Biologie	Shanghai	1.6
Meyer	3	Musik	Burnaby	2.1
Monk	12	Informatik	Victoria	1.2
Wang	5	Mathematik	Nanjing	1.8
Wise	1	Philologie	Toronto	1

Tabelle 2.1: Datensätze *vor* Selektion

Normalerweise ist die Menge der betrachteten Tupel viel größer, aber als Beispiel reicht unsere Minidatenbasis aus. Durch Ausführung der oben angeführten SQL-Anfrage, wird der Algorithmus auf die Datensätze aus Tabelle 2.2 beschränkt.

Name	Semester	Fachrichtung	Geb-Ort	Note
Andersson	7	Geschichte	Vancouver	1.5
Fraser	6	Physik	Ottawa	1.1
Gupta	10	Mathematik	Bombay	1.7
Liu	8	Biologie	Shanghai	1.6
Monk	12	Informatik	Victoria	1.2
Wang	5	Mathematik	Nanjing	1.8

Tabelle 2.2: Datensätze *nach* Selektion

Da im weiteren Verlauf des Algorithmus die Attribute *Name* und *Semester* nicht mehr von Interesse sind, können sie durch Projektion entfernt werden. Als nächster Schritt erfolgt die Generalisierung der einzelnen Attribute. Nehmen wir an, daß für alle Attribute die obere Schranke den Wert 3 besitzt. Aus Tabelle 2.2 ist ersichtlich, daß das Attribut *Geb-Ort* 6 verschiedene Werte annimmt. In diesem Fall wird durch Generalisierung dieses Attributes, jedes vorkommende Konzept der Hierarchie, durch sein nächsthöheres Konzept ersetzt. Die generalisierte Relation sieht wie folgt aus:

Zu beachten ist, daß nach diesem Schritt, zwei Tupel identische Einträge erhalten haben, so daß eines entfernt werden konnte. Die nächste Generalisierung bezieht sich auf das Attribut *Geb-Ort* und führt zu den Veränderungen aus Tabelle 2.4.

Eine weitere Vereinfachung ist möglich, falls man in Betracht zieht, daß jedes Fach entweder der Naturwissenschaft oder der Kunst zugeordnet werden kann. Tabelle 2.5 enthält die generalisierte *Endrelation*.

Fachrichtung	Geb-Ort	Note
Kunst	B. C.	sehr-gut
Naturwissenschaft	Ontario	sehr-gut
Naturwissenschaft	B. C.	sehr-gut
Naturwissenschaft	India	gut
Naturwissenschaft	China	gut

Tabelle 2.3: Datensätze nach 1.Generalisierung des Attributes *Geb-Ort*, *Note* und *Fachrichtung*

Fachrichtung	Geb-Ort	Note
Kunst	Canada	sehr-gut
Naturwissenschaft	Canada	sehr-gut
Naturwissenschaft	foreign	gut

Tabelle 2.4: Datensätze nach der 2. Generalisierung des Attributes *Geb-Ort*

Aus dieser Endrelation läßt sich folgendes ableiten:

$\forall(x)(Semester(x) \geq 5 \implies ((Geb \Leftrightarrow Ort = Canada \wedge Note = sehr \Leftrightarrow gut) \vee (Fachrichtung(x) = Naturwissenschaft \wedge Geb \Leftrightarrow Ort = foreign \wedge Note = gut))$

2.1.3 LCLR-Algorithmus

Wie im vorigen Abschnitt bereits erläutert, wird der LCHR-Algorithmus auf eine Klasse von Datensätzen angewandt. Mit Hilfe der attributorientierten Induktion können für diese Tupelmenge charakteristische Merkmale bestimmt werden. Die durch den Algorithmus erhaltene Regel kann wie folgt interpretiert werden:

$$\text{Zielklasse(x)} \implies \text{Bedingung(x)}$$

Nun stellt sich die Frage, ob ein ähnlicher Algorithmus nicht vielleicht Klassifikationsmerkmale bestimmter Mengen von Datensätzen liefern könnte? Mit diesem Problem befaßt sich der sogenannte *Learning classification rules*-Algorithmus (LCLR). Für die erhaltene Regel, die man auch als Bedingung für die Klassenzugehörigkeit betrachten kann, gilt folgende Beziehung :

$$\text{Bedingung(x)} \implies \text{Zielklasse(x)}$$

Fachrichtung	Geb-Ort	Note
Any	Canada	sehr-gut
Naturwissenschaft	foreign	gut

Tabelle 2.5: Datensätze nach Vereinfachung durch *Any*

LCLR-Algorithmus
Eingabe

- Relationale Datenbank
- Konzepthierarchie für jedes Attribut der Anfrage
- SQL-Anfrage
- Obere Schranke für jedes Attribut

Ausgabe

- Je eine Klassifikationsregel für Ziel- und Gegenbeispielklasse

Durchführung

1. Im erstem Schritt des Algorithmus wird mit Hilfe einer Bedingung bezüglich eines oder mehrerer Attribute, die Menge der relevanten Datensätze in eine Ziel- und Gegenbeispielklasse geteilt. Die Zielklasse enthält diejenigen Tupel welche die Bedingung erfüllen, alle anderen Tupel gehören der Gegenbeispielklasse an.

2. Auf die gesamte Menge der Datensätze wird nun die attributorientierte Induktion aus Abschnitt 2.1.2 angewandt. Zusätzlich muß nach jedem Generalisierungsschritt festgestellt werden, ob durch das Ersetzen eines Attributwertes durch ein nächsthöheres Konzept dieses Attributes, Ziel- und Gegenbeispielklasse gemeinsame Tupel besitzen (d.h. es wird der Durchschnitt der beiden Klassen untersucht). Ist dieses der Fall so werden die gemeinsamen Tupel markiert. Markierte Tupel vererben *ihre Markierung* falls sie durch Generalisierung verändert werden, d.h. das neu entstandene Tupel behält die Markierung seines Vorgängers. Die folgende Prozedur systematisiert die oben erläuterte Vorgehensweise.

```
procedure Attributorientierte Induktion
    A_i-Attribute der Anfangsrelation, T-obere Schranke,
    N-Anzahl nicht markierter Tupel der Zielklasse, d_i Anzahl
    verschiedener Attributwerte von A_i
 BEGIN
      für jedes Attribut A_i DO
    BEGIN
          Berechne den Durchschnitt der Ziel- und Gegenbeispielklasse
          Markiere die identischen Tupel
      WHILE d_i > T DO
            IF es existiert kein höheres Konzept für A_i
                  THEN Entferne Attribut A_i
                  ELSE
                      BEGIN
                          Ersetze die Werte von A_i durch höhere Konzepte
                          Markiere die identischen Tupel der beiden Klassen
```

```
                    Entferne gleiche Tupel innerhalb jeder Klasse
                END
        END
        WHILE (N in der Zielklasse) > T DO
            BEGIN
              Generalisiere diejenigen Attribute mit der größten
              Anzahl verschiedener Werte, oder solche bei denen ei-
              ne effiziente Reduktion zu erwarten ist.
              Markiere die identischen Tupel der beiden Klassen
              Entferne gleiche Tupel innerhalb jeder Klasse
            END
    END {Attributorientierte Induktion}
```

3. Die markierten Tupel werden in beiden Klassen entfernt und falls möglich, wird eine Vereinfachung wie in Abschnitt 2.1.2 durchgeführt.

4. Aus der so erhaltenen Relation kann die Klassifikationsregel für die relevanten Datensätze abgeleitet werden. Eine recht anschauliche Darstellung, bietet der im vorigen Abschnitt vorgestellte Ausdruck der Prädikatenlogik 1. Stufe.

Beispiel: Wir gehen von der in Tabelle 2.1 vorgestellten Datenbasis aus und setzen uns als Ziel Klassifikationsregeln für Ziel- und Gegenbeispielklasse aufzustellen, die durch folgende SQL-Anfrage gegeben sind:

```
select Fachrichtung, Geb-Ort, Note
from student
where semester ≥ 5
```

Das Ergebnis dieser Anfrage, zeigt Tabelle 2.6.

Lernendes Konzept	Fachrichtung	Geb-Ort	Note
	Geschichte	Vancouver	1.5
	Physik	Ottawa	1.1
	Mathematik	Bombay	1.7
Semester ≥5	Biologie	Shanghai	1.6
	Informatik	Victoria	1.2
	Mathematik	Nanjing	1.8
	Mathematik	Calgary	1.3
	Philologie	Edmonton	2.4
	Chemie	Richmond	2.3
Semester <5	Informatik	Victoria	1.5
	Musik	Burnaby	2.1
	Philologie	Toronto	1

Tabelle 2.6: Datensätze nach Ausführung der SQL-Anfrage

Durch Generalisierung der Attribute *Geb-Ort*, *Name* und *Fachrichtung* erhält die Zielklasse zwei identische Tupel von denen eines entfernt werden kann. Zusätzlich fällt auf, daß Ziel- und Gegenbeispielklasse ein gemeinsames Tupel besitzen, folglich werden diese beiden Tupel markiert. Die generalisierten Datensätze sind in Tabelle 2.7 dargestellt.

Lernendes Konzept	Fachrichtung	Geb-Ort	Note
Semester $\geq$5	Kunst	B.C.	sehr-gut
	Naturwissenschaften	Ontario	sehr-gut
	Naturwissenschaften	India	gut
	Naturwissenschaften	China	gut
	Naturwissenschaften	B.C.	sehr-gut
	Naturwissenschaften	China	gut
Semester <5	Naturwissenschaften	Alberta	sehr-gut
	Kunst	Alberta	gut
	Naturwissenschaften	B.C.	gut
	Naturwissenschaften	B.C.	sehr-gut
	Kunst	B.C.	gut
	Kunst	Ontario	sehr-gut

Tabelle 2.7: Datensätze nach Generalisierung der Attribute *Geb-Ort*, *Name* und *Fachrichtung*

Eine weitere Generalisierung des Attributes *Geb-Ort* (wegen nicht erfüllter Schrankenbedingung) führt sowohl zu identischen Tupeln innerhalb jeder Klasse als auch zu identischen Tupeln der beiden Klassen. Es müßen also sowohl Tupel entfernt als auch welche markiert werden. Die veränderten Datensätze sind in Tabelle 2.8 dargestellt. Tabelle 2.9 des Beispiels erhält man durch die Vereinfachung $Naturwissenschaften \vee Kunst = ANY$.

Lernendes Konzept	Fachrichtung	Geb-Ort	Note
Semester $\geq$5	Kunst	Canada	sehr-gut
	Naturwissenschaften	Canada	sehr-gut
	Naturwissenschaften	Foreign	gut
Semester <5	Naturwissenschaften	Canada	sehr-gut
	Kunst	Canada	gut
	Naturwissenschaften	Canada	gut
	Kunst	Canada	sehr-gut

Tabelle 2.8: Datensätze nach der 2.Generalisierung des Attributes *Geb-Ort*

Lernendes Konzept	Fachrichtung	Geb-Ort	Note
Semester $\geq$5	Naturwissenschaften	Foreign	gut
Semester <5	Naturwissenschaften	Canada	sehr-gut
	ANY	Canada	gut

Tabelle 2.9: Datensätze nach Vereinfachung durch ANY

Folgende Regeln können nun aus Tabelle 2.9 abgeleitet werden:

$\forall(x)((Fachrichtung(x) = Naturwissenschaften \wedge Geb \Leftrightarrow Ort(x) = Foreign' \wedge Note(x) = gut) \Longrightarrow (Semester(x) \geq 5))$

$\forall(x)(((Geb \Leftrightarrow Ort(x) = Canada \wedge Note(x) = gut) \vee (Fachrichtung(x) = Naturwissenschaft \wedge Geb \Leftrightarrow Ort(x) = Canada \wedge Note(x) = sehr \Leftrightarrow gut)) \Longrightarrow (Semester(x) < 5))$

Zusammenfassend kann also gesagt werden, daß für die erhaltenen Regeln die man auch als Bedingungen für die Klassenzugehörigkeit betrachten kann, folgende Beziehungen gelten:

$Bedingung_1$(x) $\Longrightarrow$ Zielklasse(x)
$Bedingung_2$(x) $\Longrightarrow$ Gegenbeispielklasse(x)

Falls Ziel- und Gegenbeispielklasse keine gemeinsamen Tupel besitzen (welche im letzten Schritt des Algorithmus entfernt werden), so gilt für beide Beziehungen die Äquivalenz.

2.2 Semantische Anfrageoptimierung

2.2.1 Einführung

Das Problem der Optimierung einer Anfrage, kann sowohl unter syntaktischem als auch unter semantischem Gesichtspunkt betrachtet werden. In beiden Fällen, stellt sich die Frage, welche der betrachteten Formulierungen günstiger ist?
Bei der syntaktischen Anfrageoptimierung wird die Anfrage zunächst mit möglichst wenig Aufwand in einen Ausdruck der relationalen Algebra übersetzt. Dort muß dann der kostengünstigste Ausführungsplan ermittelt werden. Ein günstiger Plan ist dann etwa ein solcher, bei dem mit jedem Schritt in der Abarbeitungsreihenfolge, der Umfang der entstehenden Zwischenrelation minimiert wird. Dazu benötigt man allerdings Angaben zum Umfang der gespeicherten Relation und zu den Werteverteilungen unter den Attributen sowie statistische Schätzmodelle für den Umfang der aus den Einzelschritten hervorgehenden Zwischenrelationen.
Bei der semantischen Anfrageoptimierung hingegen wird versucht, aus vorangegangenen Anfragen, nützliches Wissen zu gewinnen. Anschließend kann dieses Wissen verwendet werden, um neue Anfragen möglichst effizient zu bearbeiten. Zu den Kosten tragen bei die Ein-/Ausgabeoperationen beim Zugriff auf den Hintergrundspeicher, der CPU-Verbrauch und, bei verteilten Anwendungen, das Nachrichtenaufkommen zwischen den Knoten. Die Kosten - jeweils gemessen als Zeitverbrauch - lassen sich in unterschiedlicher Weise zu Gesamtkosten zusammenfassen. Einmal zur Bearbeitungzeit, indem alle genannten Zeitverbräuche aufaddiert werden, zum anderen zur Antwortzeit, die die Zeit beschreibt, die von Beginn bis Ende der Anfragebearbeitung verstrichen ist und wegen der Möglichkeit paralleler Ausführung (in verteilten Anwendungen) unterhalb der Bearbeitungszeit liegen kann.
Im nächsten Abschnitt werden einige Anfragen und Regeln vorgestellt, wobei versucht

wird, für eine gegebene Anfrage anhand von Regeln semantisch äquivalente Anfragen zu finden. Abschnitt 2.2.3 beschreibt den Zyklus, welchen eine Anfrage in einem Datenbanksystem mit semantischem Optimierer durchläuft. Anschließend wird in Abschnitt 2.2.4 ein induktiver Lernalgorithmus zum Aufbau semantisch äquivalenter Anfragen vorgestellt.

2.2.2 Benutzer-Anfragen und Regeln

Die semantische Anfrageoptimierung ist auf verschiedene Typen von Datenbasen anwenbar. Trotzdem beschränkt sich das vorliegende Kapitel auf den Einsatz von SQO (semantic query optimization) in relationalen Modellen, da diese das breiteste Anwendungsspektrum besitzen. Der wesentliche Unterschied zwischen der konventionellen Anfrageoptimierung und dem in diesem Kapitel behandelten semantischen Ansatz besteht darin, daß die erste Methode durch eine geschickte Ausführungsreihenfolge zum Ziel kommt, wobei die letzte semantisches Wissen in Form von Regeln einsetzt, um eine möglichst hohe Effizienz zu erzielen.
Wir nehmen nun an, daß die Datenbasis aus einer Menge von Relationen besteht. Eine Relation wird als Menge von Instanzen angesehen, wobei eine Instanz ein Vektor ist, dessen einzelne Einträge Attributwerte sind. Die Anzahl der Attribute ist für alle Instanzen einer Relation gleich. Die Attributwerte sind entweder Zahlen oder Zeichenketten, der Typ muß aber festgelegt sein.
Weiter unten ist das Schema einer Beispieldatenbasis mit zwei Relationen und den zugehörigen Attributen angegeben. Anschließend sind drei semantische Regeln für die vorhandenden Relationen festgehalten. In der Beispieldatenbasis enthält die Relation *geoloc* Daten über die geographische Lage von Siedlungen, wobei das Attribut *glc-cd* (geographic location code) Schlüssel für beide Relationen ist.

Schema
geoloc (name, glc-cd, country, latitude, longitude)
seaport(name, glc-cd, storage, silo, crane, rail)

Regeln
R1:geoloc(...,*Malta*, ?latitude,...) $\Rightarrow$?latitude >= 35.89
R2:geoloc(...,?glc-cd,*Malta*,...) $\Rightarrow$ seaport(..., ?glc-cd,...)
R3:seaport(...,?glc-cd, ?storage,...) $\wedge$ geoloc(...,?glc-cd,*Malta*,...) $\Rightarrow$?storage > 2.000.000

Semantische Regeln für Anfrageoptimierung müssen in Form von Hornklauseln vorliegen, um eine relativ einfache Überprüfung dieser zu gewährleisten. Regel 1 besagt, daß in unserer Datenbasis alle Siedlungen Maltas mindestens 35.89 m über dem Meeresspiegel liegen. Regel 2 sagt aus, daß alle erfaßten Siedlungen Maltas gleichzeitig auch Häfen sind. Regel 3 besagt schließlich, daß alle Maltesischen Häfen ein Fassungsvermögen von über 2. 000. 000 ft^3 besitzen. Mit Hilfe dieser semantischen Regeln kann nun versucht werden, Anfragen zu vereinfachen, das heißt ihre Abarbeitungskosten zu reduzieren. Gegeben sei die Anfrage Q_1, welche dem *select-from-where* Konstrukt von SQL entspricht:

Q_1:answer(?name):-
geoloc(?name, ?glc-cd,*Malta*,...)
seaport(..., ?glc-cd, ?storage,...)
?storage > 1.500.000

Anhand der vorgestellten Regeln, kann das System für Q_1 äquivalente Anfragen formulieren. Anschließend müssen sie auf ihre Kosten hin untersucht werden, um festzustellen, welches die kostengünstigste ist. Folgende zu Q_1 äquivalente Anfragen, lassen sich sofort ableiten:

Q_1 *und* $R_3 \Longrightarrow Q_{21}$: answer(?name):-
geoloc(?name, ?glc-cd, *Malta*,...)
seaport(..., glc-cd,...)

Q_{21} *und* $R_2 \Longrightarrow Q_{22}$: answer(?name):-
geoloc(?name, *Malta*,...)

Q_{22} *und* $R_1 \Longrightarrow Q_{23}$:answer(?name):-
geoloc(?name, ?glc-cd, *Malta*,?latitude,...)
?latitude <= 35. 89

Möglich wäre auch, daß das System eine Anfrage ausschließlich mit Hilfe der zur Verfügung stehenden Regeln beantworten kann. In diesem Fall erzielt man mittels semantischer Anfrageoptimierung den größten Gewinn.
Für die obigen drei Anfragen muß nun das Problem der Kosten behandelt werden. Nehmen wir an, daß die Relation *geoloc* eine große Anzahl von Ausprägungen besitzt. Diese sind nach *glc-cd* sortiert, wobei die Relation *seaport* relativ klein ist. Spontan würde man behaupten, daß die Anfrage Q_{22} die kostengünstigste ist. Eine genauere Betrachtung zeigt aber, daß alle Tupel der Relation *geoloc* durchlaufen werden müssen, um die Selektionsbedingung *country* = *Malta* zu überprüfen. Auch kann die Sortierung der Relation *geoloc* nicht ausgenutzt werden, woraus man schließen kann, daß eine Verbesserung der Bearbeitungszeit durchaus möglich ist. Bei der Bearbeitung der Anfrage Q_{23} muß ebenfalls die gesamte Relation durchlaufen werden. Wegen Regel R_2 besitzt Anfrage Q_{21} die kleinsten Bearbeitungskosten und wird vom System ausgewählt, um an Stelle von Anfrage Q_1, ausgeführt zu werden.

2.2.3 Induktiver Lernalgorithmus zur Erzeugung von Regeln

Um semantische Anfrageoptimierungen durchführen zu können, benötigt ein Datenbanksystem einen Anfrageoptimierer, ein lernendes System und eine Regeldatenbank. Der Anfrageoptimierer nimmt die Anfrage Q_{Eing} entgegen, optimiert sie anhand der Regeln aus

der Regeldatenbank (d.h. es werden äquivalente Anfragen erzeugt und diejenige mit den kleinsten Kosten wird ausgewählt) und gibt sie zur Ausführung an das DBMS (database managment system) weiter. Falls da festgestellt wird, daß trotz Optimierung die Anfrage Q_{opt} komplex ist, wird sie nach ihrer Ausführung, dem lernenden System zugeführt, um daraus nützliches Wissen zu gewinnen. Die erhaltene Information wird in Form von Regeln, in die Regelbank aufgenommen. Eine graphische Veranschaulichung des Zyklus', den eine Anfrage im Regelfall durchläuft, bietet Abbildung 2.3.

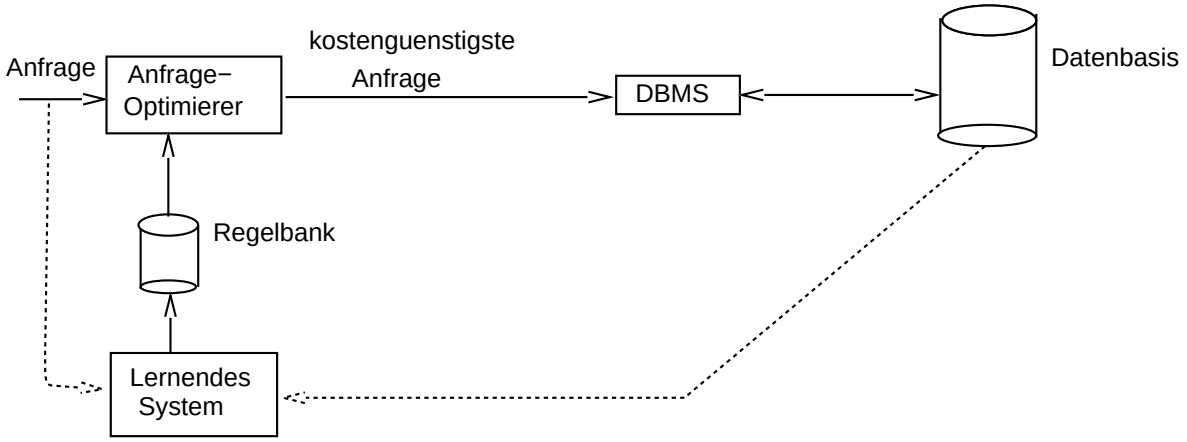


Abbildung 2.3: Struktur eines Datenbanksystems mit SQO Optimierer und lernendem System

Eine weitere interessante Fragestellung ist, wie aus der optimierten Anfrage Q_{opt} Wissen gewonnen werden kann. In einer ersten Phase, wird die Äquivalenzbeziehung der Eingabeanfrage und der optimierten Anfrage in zwei Hornklauseln umgeformt. Man erhält die Beziehungen :

$$Q_{Eing} \Longrightarrow Q_{opt}$$
$$Q_{opt} \Longrightarrow Q_{Eing}$$

Die zweite Phase befaßt sich mit der Vereinfachung der beiden Ausdrücke. Vereinfachung bedeutet in diesem Zusammenhang, die Regeln so zu gestalten, daß der Optimierer ihre Überprüfung möglichst schnell durchführen kann. Dieses ist der Fall, wenn die linke Seite einer Hornklausel eine minimale Anzahl von Literalen enthält, die konjunktiv miteinander verknüpft sind. Ein Verfahren, welches unnötige Einschränkungen eliminiert, ist z.B. der *" greedy minimum set cover"*Algorithmus (Cormen, 1989). Die erhaltenen Regeln werden in der Regelbank abgespeichert und für die Optimierung von weiteren Eingabe-Anfragen benutzt. Somit *lernt* das System im Laufe der Zeit eine Menge von Regeln, die es ihm mit großer Wahrscheinlichkeit ermöglichen, gestellte Anfragen sehr effizient zu optimieren. Gleichzeitig steigt auch die Anzahl der Anfragen, die ausschließlich mit Hilfe der Regelbank beantwortet werden können, so daß eine optimale Antwortzeit erzielt wird.

2.2.4 *Lernen* äquivalenter Anfragen

Im vorigen Abschnitt wurde das Problem der Ableitung von Regeln aus äquivalenten Anfragen behandelt. Eine andere Methode der Regelerzeugung ist durchaus möglich, wenn man an die im ersten Kapitel eingeführte attributorientierte Induktion denkt. Mit Hilfe des LCHR- oder des LCLR-Algorithmus können Regeln abgeleitet werden; dieses Verfahren setzt allerdings voraus, daß für die relevanten Attribute, Konzepthierarchien gegeben sind.
In diesem Abschnitt soll nun erläutert werden, wie zu einer gegebenen Anfrage, äquivalente Anfragen erzeugt werden. Das Verfahren beruht auf einem induktiven Lernalgorithmus, der schrittweise eine äquivalente Anfrage mit dem günstigsten Gewinn/Kosten-Verhältnis aufbaut. Der Algorithmus erhält als Eingabe eine Benutzer-Anfrage Q und eine relationale Datenbasis *DB*. Als *primäre Relation* bezeichnen wir diejenige, deren Attribute zu den Ausgabevariablen gehören. Sind mehrere Relationen an der Ausgabe beteiligt, entsteht die primäre Relation durch eine oder mehrere *Join*-Verbindungen. Anschließend werden die Instanzen der primären Relation als positiv bzw. negativ gekennzeichnet, je nachdem ob sie die Benutzeranfrage erfüllen oder nicht. Für jedes Attribut der primären Relation kann das System nun Einschränkungen einfügen, soweit diese alle positiven Instanzen abdecken. Auf diese Weise wird eine neue Anfrage aufgebaut, die letztendlich mit der Benutzeranfrage äquivalent sein soll, d.h. als Ausgabe dieselben Tupel liefert. In der Regel bieten sich einige Einschränkungen an, seien es Attributeinschränkungen oder solche, entstanden durch *Join*-Verbindungen mehrerer Relationen. Es stellt sich offensichtlich die Frage, welche der sich anbietenden Möglichkeiten die günstigste ist. Man muß leider erkennen, daß eine genaue Kostenanalyse sich vom Aufwand her nicht lohnt, so daß man sich auf die Maximierung eines Gewinn/Kosten-Verhältnisses beschränken muß. Der Gewinn ist in diesem Zusammenhang gleich der Anzahl negativ markierter Instanzen, die durch die jeweilige Einschränkung ausgeklammert werden. Die Kosten hängen von der Art des Zugriffs auf das betreffende Attribut oder die entsprechende Relation (im Falle einer Einschränkung durch *Join*-Verbindung) ab. Tabelle 2.10 enthält die Kostenfunktion für alle möglichen Fälle:

Attribut-Einschränkung, nicht index.Attribute	$\|D_1\|$
Attribut-Einschränkung, indexierte Attribute	I
Join, über zwei nicht index. Attribute	$\|D_1\|*\|D_2\|$
Join, über zwei index. Attribute	$\|D_1\|*\|D_2\|/\max\{\|I_1\|,\|I_2\|\}$

Tabelle 2.10: Geschätzte *Kosten* für Einschränkungen in einer Anfrage

Dazu bezeichnet $|D|$, die Anzahl der Tupel der Relation D, I ist die Anzahl der sich qualifizierenden Tupel einer Relation und I_1,I_2 stellen die Anzahl der unterschiedlichen Attributwerte dar, welche für die *Join*-Operation notwendig sind.

Beispiel: Gegeben seien die beiden Anfragen C1,C2 und ein Fragment einer Datenbasis in Tabelle 2.11:

geoloc(*Safaqis*,	8001,	Tunisia,	,....)	seaport(*Marsaxlokk*,	8003,	...)
geoloc(*Valletta*,	8002,	Malta,	,....)	seaport(*Grand Harbor*,	8002,	...)
geoloc(*Marsaxlokk*,	8003,	Malta,	,....)	seaport(*Marsa*,	8005,	...)
geoloc(*San Pawl*,	8004,	Malta,	,....)	seaport(*St. Pauls Bay*,	8004,	...)
geoloc(*Marsalforn*,	8005,	Malta,	,....)	seaport(*Catania*,	8016,	...)
geoloc(*Abano*,	8006,	Italy,	,....)	seaport(*Palermo*,	8012,	...)
geoloc(*Torino*,	8007,	Italy,	,....)	seaport(*Traparri*,	8015,	...)
geoloc(*Venezia*,	8008,	Italy,	,....)	seaport(*AbuKamash*,	8017,	...)

Tabelle 2.11: Fragment einer Datenbasis

C1 :geoloc(?name, *Malta*,...)
C2 :geoloc(?name,..., ?glc-cd,...)
seaport(..., ?glc-cd,...)

Wir nehmen nun an, daß $|geoloc| = 30.000$ und $|seaport| = 800$. Die Kardinalität von *glc-cd* ist für *geloc* ebenfalls 30.000 und für *seaport* auch 800 (Schlüsseleigenschaft). Falls beide Relationen einen indizierten Zugriff auf *glc-cd* erlauben, sehen die beiden Gewinn/Kosten-Verhältnisse wie folgt aus:

$$Gewinn/Kosten(C1) : \frac{30.000 \Leftrightarrow 4}{30.000} = 0.99 \tag{2.1}$$

$$Gewinn/Kosten(C2) : \frac{30.000 \Leftrightarrow 800}{800} = 36.50 \tag{2.2}$$

Das System wird sich in diesem Fall für C2 entscheiden.
Der induktive Lernalgorithmus beschreibt den systematischen Aufbau einer Anfrage, die äquivalent zu der gegebenen Benutzeranfrage sein muß und gleichzeitig die kostengünstigste darstellen soll.

Algorithmus
Eingabe : Q = Benutzeranfrage, DB = Relationale Datenbasis;

Ausgabe : Eine zu Q äquivalente Anfrage AQ (kostengünstigste Anfrage)

```
BEGIN
  LET r = Primäre Relation von Q (anfangs leer);
  LET AQ = Konstruierte Anfrage (anfangs leer);
  LET C = Menge von möglichen Einschränkungen (anfangs leer);
  Konstruiere Einschränkungen für r und füge sie C zu;
  REPEAT
     Berechne das Gewinn/Kosten-Verhältnis für alle Kandidaten aus C;
     LET c = Kandidat mit maximalen Gewinn/Kosten-Verhältnis;
     IF Gewinn(c) > 0 THEN
        Füge c zu AQ hinzu, C = C - c;
        IF AQ ⟺ Q THEN RETURN AQ;
```

```
        IF c ist eine join-Verbindung mit einer neuen Relation r'
          THEN Konstruiere Einschränkungen für r' und füge sie C zu;
        ENDIF;
      ENDIF;
    UNTIL Gewinn(c) = 0;
    RETURN Fehler, da keine zu Q äquivalente Anfrage AQ konstruiert werden konnte;
END.
```

Die in dieser Ausarbeitung vorgestellten Verfahren zeigen, daß das Wissen, welches für die Durchführung semantischer Anfrageoptimierungen notwendig ist, induktiv also schrittweise erlernt werden kann. Auslöser für diesen Lernprozeß sind die Benutzeranfragen. Experimentelle Ergebnisse zeigen, daß Anfrageoptimierung mittels semantischem Wissen erhebliche Kostenreduktionen zur Folge hat.

Kapitel 3

Unüberwachtes Lernen durch conceptual clustering

Dierk König
Dipl. Betriebswirt (BA)

Kurzfassung *Conceptual clustering ist ein Verfahren aus dem Gebiet der künstlichen Intelligenz, das maschinelles Lernen realisiert. Es folgt dabei dem Paradigma des unüberwachten Lernens, bei dem Zusammenhänge zwischen beobachteten Ereignissen selbständig erkannt werden. Dieser Beitrag stellt das Prinzip dieses Verfahrens vor und beschreibt UNIMEM als Beispiel für eine Implementierung.*
Vorangestellt wird eine Überlegung, wie conceptual clustering zur Wissensgewinnung aus Datenbasen genutzt werden könnte. Zu diesem Zweck wird gezeigt, wie eine Datenbasis als Eingabe für das Verfahren dienen kann, und wie die Ergebnisse des Verfahrens als Wissen über die Datenbasis interpretiert werden können.

3.1 Idee

3.1.1 Wissen in Datenbasen

Darstellung der Datenbasis

Für die Überlegungen dieses Beitrags sei die Datenbasis als Tabelle T gegeben. Die Zeile mit der Nummer i sei als Instanz i, in Zeichen I(i), bezeichnet. Die Spalte j der Tabelle heißt Attribut j, in Zeichen A(j). Das Tabellenelement in der Zeile i und der Spalte j sei t(i,j).

Abbildung 3.1 zeigt die Struktur einer Datenbasis T mit m Instanzen und n Attributen. Die Menge der Werte, die ein Attribut A(j) annehmen kann, ist die Domäne von A(j), in Zeichen Dom(A(j)). Die Instanzen seien paarweise disjunkt, also (I(a)=I(b)) $\Rightarrow$ (a=b).

	I\A	A(1)	A(2)	...	A(n)
T:=	I(1)	t(1,1)	t(1,2)	...	t(1,n)
	I(2)	t(2,1)	t(2,2)	...	t(2,n)
	...	...	...	...	...
	I(m)	t(m,1)	t(m,2)	...	t(m,n)

Abbildung 3.1: Struktur der Datenbasis

Die Kombination[1] von Attribut und dem Wert, den dieses Attribut hat, heißt *Merkmal* (engl. feature[2]). Die Instanz I(2) trägt also zum Beispiel das Merkmal A(2)=t(2,2).

Die obige Definition von T macht nur Sinn, wenn die zu untersuchenden kommerziellen Datenbasen auch leicht in diese Form gebracht werden können. Für relationale Datenbasen ist die Entsprechung augenfällig. Besteht eine relationale Datenbasis aus mehreren Relationen, so müssen diese geeignet zusammengefaßt werden. Attribute, die nicht in die Auswertung eingehen sollen, können durch Projektion ausgeblendet werden.

Wissen als strukturelle Abhängigkeit

Im wesentlichen ist in der Datenbasis die Information gespeichert, daß eine Instanz i existiert ($1 \leq i \leq m$), deren Attribut j ($1 \leq j \leq n$) den Wert t(i,j) hat. Diese Information wird benutzt, um die Anfragen der Benutzer zu beantworten.

Darüberhinaus kann die Struktur einer Datenbasis jedoch noch weiteres Wissen über die in ihr abgebildete Miniwelt enthalten. Wenn zum Beispiel für jedes i aus 1 bis m gilt: aus t(i,1)=x folgt t(i,2)=y, mit $x \in Dom(a(1))$ und $y \in Dom(A(2))$, so läßt dies auf eine Abhängigkeit der Merkmale A(1)=x und A(2)=y schließen. Wenn sich wie in diesem Beispiel das zweite Merkmal *zwingend* aus dem ersten ergibt, dann könnte man das zweite Merkmal als *funktional abhängig*[3] vom ersten Merkmal bezeichnen.

Diese Bezeichnung läßt sich auch leicht von einzelnen Merkmalen auf Mengen von Merkmalen ausdehnen, die man auch Kombinationen von Merkmalen nennen kann. Sei zum Beispiel MK_1 die Menge der Merkmale $A(1)=x_1$, $A(2)=x_2$ und $A(5)=x_5$ und sei MK_2 die Menge der Merkmale $A(3)=x_3$ und $A(4)=x_4$ mit jeweils $x_n \in Dom(A(n))$, so heißt MK_2 *funktional abhängig* von MK_1, wenn für alle i aus 1 bis m gilt: aus $t(i,1)=x_1$ und $t(i,2)=x_2$ und $t(i,5)=x_5$ folgt $t(i,3)=x_3$ und $t(i,4)=x_4$.

In einem weitergehenden Ansatz könnte man für die obigen Implikationen auch "Ausnahmen" zulassen. Die Abhängigkeit der Merkmale wäre dann nicht mehr "zwingend", sondern nur noch "wahrscheinlich".[4] Man könnte auch von einer *strukturellen* Abhängigkeit

[1] vgl. Gennari et al. [23], Seite 23

[2] Klatt [28], Seite 209

[3] in Anlehnung an die Bezeichnungen in der Datenbanktechnik, vgl. Lockemann et al. [34], Kapitel 11.2.3.1 *Funktionale und mehrwertige Abhängigkeiten*. Dort bezieht sich die Bezeichnung aber auf Attribute, nicht auf Merkmale.

[4] Wahrscheinlichkeit= Anzahl der Instanzen, für die Prämisse und Konklusion wahr sind, geteilt durch die Anzahl der Instanzen, für die die Prämisse wahr wird.

sprechen, von der die funktionelle Abhängigkeit ein Spezialfall mit der Wahrscheinlichkeit 1 wäre.

Nachdem man strukturelle Abhängigkeiten von Merkmalen gefunden hat, könnte man diejenigen Instanzen I(i), für die die Prämisse in der die betreffenden Implikation wahr wird, in eine Gruppe zusammenfassen und mit den so gefundenen Gruppen, die Struktur der Datenbasis beschreiben. Auch der umgekehrte Weg ist denkbar und wird im weiteren Verlauf beschritten: Man faßt Instanzen mit gemeinsamen Merkmalen zusammen und schließt auf Abhängigkeiten zwischen diesen Merkmalen.

Die Tatsache, daß bestimmte Merkmale voneinander abhängig sind, stellt ein spezifisches Wissen über die Instanzenmenge dar. Bei funktionalen Abhängigkeiten ist dieses Wissen scharf, bei lediglich strukturellen Abhängigkeiten unscharf.

Zusätzlich können für eine Datenbasis allgemeine prädikatenlogische Formeln existieren, die Wissen über die Datenbasis beschreiben. Die Gültigkeit dieser Formeln kann zusätzlich mit Wahrscheinlichkeiten behaftet sein. Solches Wissen kann mit den vorliegenden Verfahren nicht gewonnen werden.

Unüberwachtes Lernen

Lernen ist der Erwerb von Wissen. Oft besteht dieses Wissen darin, Zusammenhänge zwischen beobachteten Ereignissen zu erkennen. Dabei ist der zeitliche Zusammenhang für biologisches Lernen von besonderer Bedeutung.[5] Ein Beispiel dafür ist das Experiment von Pawlow[6]: Ein Hund hört zu jedem Fressen ein Glockensignal. Nach kurzer Lernzeit zeigt er die Freßerwartung (Speichelfluß) auch ohne Futter, wenn allein die Glocke ertönt.

Für das Lernverhalten des Hundes sind zwei Faktoren wichtig: die Häufigkeit der beobachteten Kombination (Glocke, Futter) und das Fehlen der Beobachtung (Glocke, kein Futter).

Im allgemeinen gilt für das biologische Lernen, daß gleichzeitig beobachtete Ereignisse zueinander in Beziehung gesetzt werden.

Beim überwachten, biologischen Lernen wird der Lernvorgang von einem "Lehrer" gesteuert. Der "Lehrer" bestimmt darüber, zu welchen Gruppen die Ereigniskombinationen zusammenzufassen sind. Er bestimmt, welche *Begriffe* der Schüler bilden soll. Wenn er zum Beispiel sagt: "Die Polynomfunktionen sind stetig", so soll der Schüler den Begriff "stetige Funktion" bilden und diesem unter anderen die Polynomfunktionen zuordnen.

Im Gegensatz dazu ist das *unüberwachte* Lernen ein Lernen aus Beispielen, die unstrukturiert erfahren werden. Der Schüler betrachtet zum Beispiel die Graphen vieler verschiedener Funktionen. Dabei erkennt er, daß einige Graphen "Sprünge" haben, andere nicht. Dadurch unterteilt er sie in zwei Gruppen. Die Elemente der einen Gruppe haben die Eigenschaft, stetig zu sein. Damit hat er den Begriff "stetig" selbständig erzeugt, also unüberwacht gelernt.

[5] Von anderen wichtigen Faktoren wie Reizstärke oder Lerndisposition sei hier einmal abgesehen.

[6] Ivan Petrowitsch Pawlow, 1849-1936, Physiologe und Entdecker der bedingten Konditionierung, aus [35], Seiten 320/321.

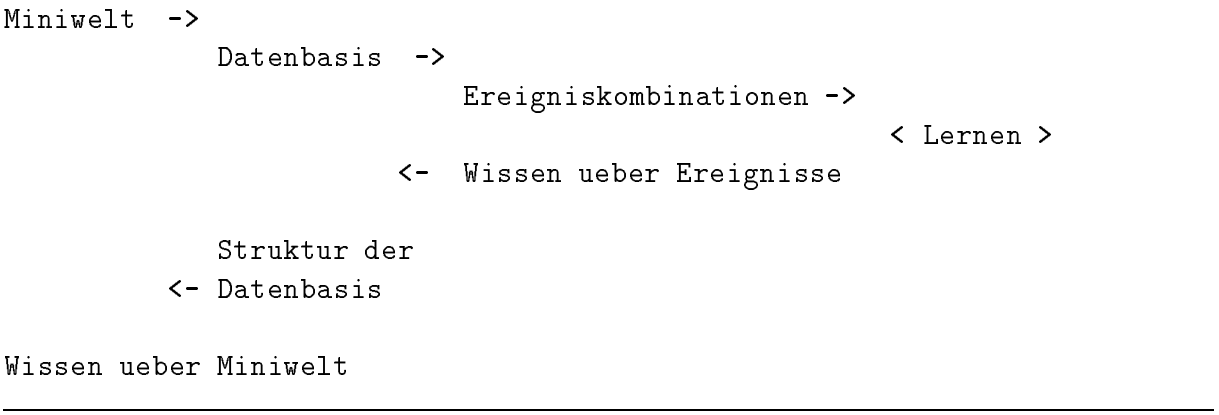


Abbildung 3.2: Lernen aus Datenbasen

3.1.2 Anwendung maschineller Lernvorgänge auf Datenbasen

In der Datenbasis ist jede Instanz eine Kombination von Merkmalen. Die Idee ist, jedes Merkmal als Ereignis anzusehen und damit jede Instanz als gleichzeitig beobachtete Ereigniskombination, die sich aus der Ausprägung ihrer Attribute ergibt. Die so gewonnenen Ereigniskombinationen werden als Eingabe für einen maschinellen Lernprozeß verwendet. Dabei hofft man, daß das damit erzeugte Wissen wichtige Strukturinformationen über die Datenbasis und damit über die abgebildete Miniwelt liefert.

Abbildung 3.2 veranschaulicht die Anwendung des Lernverfahrens auf die Datenbasis.

Ein solches Vorgehen kommt insbesondere dann in Betracht, wenn die zu untersuchende Datenbasis für einen menschlichen Auswerter zu umfangreich wird. Die angestrebte Verwendung der Lernergebnisse bedingt den Einsatz des unüberwachten Lernens als Vorgehensweise. Die Struktur der Datenbasis wird ja als unbekannt vorausgesetzt und ist erst durch den Lernvorgang zu ermitteln. Die Instanzen sind die Beispiele, deren Gemeinsamkeiten untersucht werden.

3.2 Conceptual clustering

3.2.1 Bezeichnungen

Conceptual clustering ist ein Verfahren, das beim maschinellen Lernen eingesetzt wird[7]. Die Bezeichnung ist generisch, d.h. sie bezeichnet keinen speziellen Algorithmus, sondern eine generelle Vorgehensweise.

Begriff oder "concept"

Die wörtliche Übersetzung von "conceptual clustering" bedeutet "Gruppierung mit Begriffen". Das Wort "Begriff" steht für die typisierte Vorstellung einer Sache, die von

[7] vgl. Lebowitz [33], Seite 25

den individuellen Eigenschaften der Sache abstrahiert. Dabei entsteht ein Begriff (engl. concept[8]) durch eine spezielle Merkmalskombination. So könnte in einer Personaldatenbasis mit den Attributen Name, Geburtsdatum, Geschlecht, Gehalt, Position ein Begriff definiert sein durch (Position= "Abteilungsleiter", Gehalt= "100.000 DM").

Beim menschlichen Lernen ordnet man einer als sinnvoll erkannten Merkmalskombination meist ein Wort zu, das diese Merkmale repräsentiert, in dem obigen Beispiel etwa "leitender Angestellter". Dieses Wort bezeichnet dann auch einen "Begriff". Von einer maschinellen Auswertung kann man aber billigerweise keine neuen Wortschöpfungen erwarten. Deshalb wird ein Begriff in diesem Zusammenhang nur über die Merkmalskombination definiert.

Beim conceptual clustering versucht man, aus Beispielen, Ereignissen, Datensätzen oder allgemein Instanzen, Begriffe zu bilden. Dazu durchsucht man die Instanzenmenge nach Teilmengen mit gemeinsamen Merkmalen. Aus diesen Merkmalen werden dann die Begriffe erzeugt.

Begriffsstrukturen

Beim Erwerb von Wissen werden Begriffe strukturiert, also in Unter- und Oberbegriffe eingeteilt.[9] Ein Unterbegriff erbt alle Merkmale des zugehörigen Oberbegriffes und fügt seine eigenen Merkmale hinzu. Diese Vererbung ist transitiv. Unterbegriffe sind spezieller, Oberbegriffe allgemeiner.

Ein Begriff heißt zu einer Instanz *passend*, wenn alle seine Merkmale mit der Instanz übereinstimmen. Ein passender Begriff heißt *speziellster* Begriff, wenn kein anderer passender Begriff mehr Merkmale umfaßt. In der Begriffshierarchie steht ein speziellster Begriff folglich möglichst weit "unten".

Eine Begriffsstruktur heißt *überschneidungsfrei*, wenn für jede Instanz genau ein speziellster Begriff existiert. Die Überschneidungsfreiheit kann nur gewährleistet werden, wenn die Merkmale zweier Unterbegriffe eines gemeinsamen Oberbegriffes jeweils mindestens ein gemeinsames Attribut überdecken und ihre Werte für dieses Attribut verschieden sind. Weitere Überschneidungen können dadurch entstehen, daß man zuläßt, einen Begriff auch dann als zu einer Instanz passend zu erklären, wenn sie von seinen Merkmalen geringfügig abweicht.

Geringfügige Abweichungen ergeben sich zwangsläufig bei der Verwendung von Gleitkommadomänen. Hier sind Rundungsfehler zu erwarten, die man nur dadurch ausgleichen kann, daß man zwei Werte als gleich ansieht, wenn ihre Differenz betragsmäßig kleiner ist als eine vorgegebene Schranke.

Zwei weitere Möglichkeiten der Abweichung werden durch die Unschärfe der menschlichen Begriffsbildung impliziert.[10] Ein Mensch würde eine Instanz wahrscheinlich auch dann noch dem Begriff "leitender Angestellter" zuordnen, wenn diese Instanz abweichend vom

[8] Klatt [28], Seite 118
[9] vgl. Lebowitz [33], Seite 26
[10] vgl. Lebowitz [33], Seite 30

Begriff das Merkmal Gehalt="100.001 DM" trüge. Es ist also manchmal sinnvoll, auch auf ganzzahligen Domänen einen gewissen Abstand vom Merkmal des Begriffes zu tolerieren.

Zweitens ist es denkbar, die deutliche Abweichung bezüglich eines Merkmals zu ignorieren, wenn die Instanz eine große Zahl anderer Merkmale des Begriffes erfüllt. So würde man bei einem Patienten auch dann eine Krankheit diagnostizieren, wenn er von 7 typischen Symptomen nur 6 zeigt. Außerdem können verschiedene Attribute einen unterschiedlich starken Beitrag zu Zuordnung leisten. Bei den Symptomen einer Krankheit ist das zum Beispiel der Fall.

Allgemein ist eine Abstandsfunktion[11] vorstellbar, die von den numerischen Abständen innerhalb der Merkmale und der Anzahl der erfüllten, bzw. nicht erfüllten Merkmale abhängt. Diejenigen Begriffe, die zu einer Instanz einen hinreichend geringen Abstand haben, heißen dann zu dieser passend. Hierdurch entsteht aber zwangsläufig die Möglichkeit von Überschneidungen, und zwar dann, wenn eine Instanz bezüglich zweier verschiedener Begriffe innerhalb des Maximalabstandes liegt.

Übersetzungsvorschlag

Wenn aber Überschneidungen nicht vermieden werden können, so macht es keinen Sinn von "Gruppierung" oder gar "Klassifikation" der Instanzen zu sprechen, denn diese Bezeichnungen werden meist in dem Sinn gebraucht, daß sie eine eindeutige Zuordnung ermöglichen. Aus dem gleichen Grund bilden die Begriffe auch keine "Gliederung" oder gar einen "Begriffsbaum". Sie bilden jedoch eine hierarchische Struktur über der Instanzenmenge.

Deshalb lautet mein Übersetzungsvorschlag für "conceptual clustering": hierarchische Strukturierung durch Begriffe.

Parallel dazu steht der Ausdruck "concept formation", also Begriffsbildung. Er bezeichnet den gleichen Sachverhalt, betont aber das Finden, den Aufbau, das "Lernen" der Begriffe.

3.2.2 Verfahren

Gegeben sei eine Datenbasis in der Form, die in Kapitel 3.1.1 vorgestellt wurde. Gesucht ist eine hierarchische Begriffsstruktur, die die Beziehungen in der Datenbasis möglichst gut, also *optimal* wiedergibt. Die Güte einer Begriffsstruktur kann nach mehreren Kriterien beurteilt werden, die zum Teil in Zielkonkurrenz zueinander stehen. Einige dieser Kriterien sind in Tabelle 3.1 aufgeführt.

Bergsteiger-Algorithmus

Ein mögliches Verfahren zur Lösung dieses Optimierungsproblems ist der Bergsteiger[12]- (engl. hill-climbing-) Algorithmus, dessen Grundform die Abbildung 3.3 zeigt.[13]

[11] vgl. Gennari et al. [23], Seite 25

[12] Die Namensgebung für diesen Algorithmus ist zwar sehr einprägsam, aber insofern etwas zweifelhaft, als ein Bergsteiger sich wohl kaum in dieser Weise fortbewegen würde.

[13] Nach den Überlegungen aus Gennari et al. [23], Kapitel 2.5 *Learning as incremental hill climbing*

```
Sei M(x):=Menge aller y, die in einer Entfernung s von x liegen
          mit h(y)>h(x);

Bergsteiger(x);
begin
waehle x beliebig aus E;
while M(x) nicht leer
      do begin
      x:=y aus M(x) mit h(y) maximal;
      end;
return (x);
end;
```

Abbildung 3.3: Bergsteiger-Algorithmus, Grundform

Dabei sei gegeben: eine n-dimensionale Ebene E und eine Funktion h: E $\rightarrow$ R, die jedem Punkt x aus E eine Höhe h(x) zuordnet. Weiterhin sei eine Schrittweite s gegeben.

Gesucht ist ein Optimum von h(x), für x aus E.

Der Vorteil dieses Verfahrens ist seine Effizienz und seine leichte Implementierbarkeit. Besonders angenehm ist auch sein relativ geringer Bedarf an Hauptspeicher, der im wesentlichen von der Kardinalität von M(x) abhängt.[14] Wenn man Schritte nur entlang der Koordinatenachsen zuläßt, so enthält M(x) maximal 2n Elemente.[15] Damit umgeht man die kombinatorisch explosive Berechnung von h(x) für alle x aus E, um deren Maximum zu bestimmen.

Die Schrittweite s ist jedoch ein problematischer Faktor. Ist sie zu klein, so tendiert das Verfahren dazu, an lokalen Maxima anzuhalten. Wenn s sehr groß gewählt wird, so wird zwar mit größerer Wahrscheinlichkeit das globale Maximum gefunden, aber dafür weniger genau lokalisiert, denn das tatsächliche Maximum kann immer noch in einer Entfernung vom gefundenen Optimum liegen, die gerade etwas kleiner als s ist.

Anwendung auf Begriffsstrukturen

Um den Bergsteiger-Algorithmus auf Begriffsstrukturen anwenden zu können, muß man jede mögliche Begriffsstruktur als einen Punkt in der n-dimensionalen Ebene auffassen. Dabei wird eine Begriffsstruktur in eine andere überführt, indem eine von n Operationen auf sie angewendet wird. Die Ausführung einer solchen Operation entspricht dann einem Schritt. Beginnend bei einer leeren Begriffsstruktur wird so durch die n Operationen eine n-dimensionale Ebene[16] aufgespannt.

Die möglichen Operationen umfassen zum Beispiel:

[14] vgl.Gennari et al. [23], Seiten 14/15

[15] je einen Schritt vorwärts oder rückwärts entlang jeder Koordinatenachse

[16] genauer: Es wird eine Halbebene aufgespannt, weil aus einer leeren Begriffsstruktur zum Beispiel kein Begriff gelöscht werden kann.

- das Einfügen eines Begriffes in die Struktur,
- das Löschen eines Begriffes aus der Struktur und
- das Einfügen, Löschen oder Verändern eines Merkmals in einem Begriff.
- Denkbar ist auch das Aufspalten oder Zusammenfügen von Begriffen.

Die Art der Operationen hängt stark davon ab, wie die Begriffsstruktur jeweils realisiert ist.

Weiterhin benötigt man eine Höhenfunktion, die angibt, ob bei der Operation ein Fortschritt erzielt wurde. Sie kann auch implizit dadurch gegeben sein, daß das Lernverfahren nur Operationen ausführt, die die Struktur verbessern.[17]

Es kann außerdem sinnvoll sein, den Lernvorgang inkrementell zu gestalten. Es wird also nicht mit der gesamten Datenbasis gestartet, sondern nur mit der ersten Instanz. Sie bildet den ersten, sehr speziellen Begriff. Anschließend wird immer im Wechsel eine weitere Instanz hinzugenommen und der leicht modifizierte Bergsteiger-Algorithmus gestartet. Die Modifikation besteht darin, daß der Startwert nicht beliebig ist, sondern das Ergebnis des letzten Durchlaufes als Ausgangswert genommen wird.

Spezifische Probleme

Die Angabe einer Höhenfunktion ist im allgemeinen schwierig. Ob eine bestimmte Begriffsstruktur besser oder schlechter ist, beurteilt man meist intuitiv und mit mehr Hintergrundwissen als dem Lernsystem zur Verfügung steht. Das maschinelle Lernsystem kann die Höhenfunktion aber nur nach objektiven Gütekriterien ermitteln. Diese Gütekriterien müssen gewichtet in die Höhenfunktion eingehen. Dabei sind u.a. folgende Gesichtspunkte zu berücksichtigen:

- Angestrebt wird eine Struktur, die weder zu flach noch zu tief ist, also nicht zu wenige und nicht zu viele Hierarchieebenen hat. Parallel dazu sollten die erzeugten Begriffe nicht zu speziell und nicht zu allgemein sein. Die Begriffe dürfen daher nicht zu viele und nicht zu wenige Merkmale umfassen.
- Zusätzlich ist zu bedenken, daß die optimale Tiefe der Begriffsstruktur im allgemeinen mit der Anzahl der betrachteten Instanzen steigt, und daß die optimale Anzahl der Merkmale eines Begriffes generell mit der Anzahl der Attribute steigt, die in der Datenbasis gespeichert sind. Beide Parameter sind aber vor allem hochgradig von der Semantik der betrachteten Daten abhängig. Selbst die Angaben von Bandbreiten für diese Parameter ist daher im allgemeinen schwierig.

  Die benutzerseitige Einstellung dieser Parameter für die jeweils betrachtete Datenbasis birgt wiederum die Gefahr, diejenigen Ergebnisse zu erzeugen, die man erwartet. Dadurch geht die Möglichkeit verloren, tatsächlich neues Wissen über die Datenbasis zu gewinnen.

[17] Eine neue Struktur wird dann automatisch als besser angesehen, wenn sie von einem bestimmten Algorithmus erzeugt wurde.

prägnante Merkmale	$\leftrightarrow$	gut bestätigte Begriffe
aussagefähige, d.h. nicht zu allgemeine Begriffe	$\leftrightarrow$	gut bestätigte Begriffe
umfassende, d.h. nicht zu spezielle Begriffe	$\leftrightarrow$	prägnante Merkmale
nicht zu flache Hierarchien	$\leftrightarrow$	nicht zu tiefe Hierarchien

Tabelle 3.1: konkurrierende Ziele

- Ein weiteres Gütekriterium für eine Begriffsstruktur liegt in Prägnanz der Merkmale ihrer Begriffe. Diese Merkmale sollten die gemeinsamen Eigenschaften der Instanzen umfassen, zu denen dieser Begriff paßt. Individuelle Merkmale dieser Instanzen sollen in dem Begriff möglichst nicht vorkommen. Wenn man jedoch, wie in Kapitel 3.2.1 beschrieben, für das Passen eines Begriffes einen gewissen Abstand zuläßt, so müssen nicht alle Instanzen zu denen der Begriff paßt, alle Merkmale tragen. Dann scheint es sinnvoll, ein Merkmal als besser oder prägnanter zu bezeichnen, je höher die Quote derjenigen Instanzen ist, die das Merkmal tragen. Diese Quote entspricht damit der Wahrscheinlichkeit, das eine Instanz, zu der der Begriff paßt, dieses Merkmal trägt. Eine Begriffsstruktur könnte also verbessert werden, indem die Prägnanz der Merkmale gesteigert wird, die die erzeugten Begriffe beschreiben.

- Andererseits wird man eine Begriffsstruktur bevorzugen, deren Begriffe jeweils zu einer möglichst großen Zahl von Instanzen passen, denn jede weitere Instanz, zu der ein bestimmter Begriff paßt, bestätigt diesen Begriff. Dabei wird man erwarten, daß Begriffe umso häufiger bestätigt werden, je weniger Merkmale sie umfassen und je weniger prägnant diese Merkmale sind.

  Gut bestätigte Begriffe müssen aber nicht zwangsläufig allgemein sein. Aus einer Datenbasis mit n Attributen, deren m Instanzen sich nur im Attribut A(1) unterscheiden, wird der Begriff, der die Attribute 2 bis n überdeckt, also n-1 Merkmale umfaßt, m-2 mal bestätigt.[18]

Tabelle 3.1 faßt die konkurrierenden Ziele zusammen. Wegen der Zielkonkurrenz müssen die obigen Beurteilungskriterien mit Bewertungsfunktionen versehen werden, mit denen sie vergleichbar gemacht werden können. Ihre Bewertungsfunktionen sorgen dann für ihre Gewichtung in der Höhenfunktion. Dadurch entsteht die Gefahr, nicht Vergleichbares durch willkürliche Festlegung zu vergleichen.

Dazu ein Beispiel: Ein möglicher Schritt im Bergsteiger-Algorithmus kann das Löschen eines Merkmals in einem Begriff B sein, der nun nur noch 5 statt bisher 6 Merkmale umfaßt. Dafür steigt die Zahl der Instanzen zu denen B paßt von 20 auf 30. Die Prägnanz eines verbleibenden Merkmals sinkt jedoch dramatisch. Wurde nun eine Verbesserung erzielt?

[18] Der Begriff wird erstmalig aus den Instanzen I(1) und I(2) erzeugt.

3.2.3 Freiheitsgrade

Domänen

Für die maschinelle Implementierung einer Begriffsstrukturierung ist es wünschenswert, die Domänen aller Attribute zu vereinheitlichen. So können Begriffe und deren Beziehungen leichter implementiert werden. Zur Auswahl stehen symbolische Domänen, ganzzahlige Domänen oder Gleitkommadomänen.

Symbolische Domänen eignen sich für überschneidungsfreie Strukturen mit denen Instanzen klassifiziert werden sollen. Sie eignen sich nicht für technische oder kaufmännische Datenbasen, die Zahlenwerte enthalten. Abstandsbegriffe bezüglich des Kriteriums, wie gut ein Merkmal erfüllt ist, machen auf symbolischen Domänen in der Regel[19] keinen Sinn. Beispiele für symbolische Domänen sind Wahrheitswerte, Strings und Mengen. Ein System, das auf symbolischen Domänen arbeitet ist Feigenbaums EPAM.[20]

Ganzzahlige Domänen sind mächtiger als symbolische. Symbolische Werte können in Ganzzahlen umgewandelt werden, indem man jedem Symbol eine Nummer zuordnet. Auf solcherlei umgewandelten Attributen macht der Abstandsbegriff aber weiterhin keinen Sinn. Die ursprünglich ganzzahligen Werte lassen sich aber leicht vergleichen. Sie entstehen häufig künstlich, wie zum Beispiel bei Personalnummern, Beurteilungen von 1 bis 10 oder bei Kardinal- oder Ordinalzahlen.

Gleitkommadomänen sind als echte Obermenge der Ganzzahlen am mächtigsten. Sie eignen sich zudem zum Erfassen von natürlichen oder technischen Größen, insbesondere von Meßwerten. Ihr Mangel an Genauigkeit bedingt jedoch die Einführung eines Maximalabstandes, bei dessen Unterschreitung zwei Gleitkommazahlen als gleich angesehen werden. Überschneidungsfreie Strukturen werden damit unmöglich.

Struktur

Begriffsstrukturen können mit oder ohne Möglichkeit von Überschneidungen angelegt werden. Überschneidungen haben den Vorteil, gewisse Unschärfen in der menschlichen Begriffsbildung nachzuvollziehen.

Überschneidungsfreie Strukturen geben jedoch die Möglichkeit, Instanzen oder Ereignisse genau klassifizieren zu können. Sie bilden ein "discrimination network", in dem jeder Begriff dem Test entspricht, ob eine Instanz die Merkmale dieses Begriffes trägt. Feigenbaums EPAM ist ein Beispiel für die Begriffsbildung mit überschneidungsfreien Strukturen.[21]

Aufbau

Beim Aufbau der Begriffsstruktur kann man zwei grundlegend verschiedene Methoden unterscheiden. Eine Methode operiert auf der gesamten gegebenen Datenbasis, die andere arbeitet inkrementell Instanz für Instanz.

[19] Es gibt Ausnahmen, wie phonetische oder lexikalische Abstände.
[20] Kurzbeschreibung in Gennari et al. [23] , Kapitel 3.1 ; ausführlich in Feigenbaum, Simon [16]
[21] Gennari et al. [23], Seite 17

Aufbaumethoden der ersten Art kann man im weitesten Sinn als statistische Methoden bezeichnen. Sie sind für die Anwendung auf Datenbasen prinzipiell geeignet, da hier die zu untersuchenden Instanzen oder Ereignisse nicht sukzessive anfallen, sondern fest vorgegeben sind. Sie eignen sich ebenso für off-line Auswertungen, was für kommerzielle Datenbasen günstig ist. Der Betrieb kann damit in die weniger stark belasteten Nachtstunden verlegt werden.

Inkrementelle Methoden haben den Vorteil, nach jeder weiteren betrachteten Instanz, die bis dahin bestmögliche Begriffsstruktur zu erzeugen. Dadurch kann die Auswertung mit der Entstehungszeit der Daten verzahnt werden. Bei Anwendungen des conceptual clustering, die mit Meßwerten arbeiten, können so Ausreißer erkannt und eventuell ein Alarm ausgelöst werden. Aber auch für die Anwendung auf sehr große Datenbasen ist diese Methode vorteilhaft. Es muß immer nur eine Instanz im Hauptspeicher gehalten werden, um sie zu verarbeiten. Statistische Methoden benötigen dagegen meist einen virtuellen Speicher, der viele Instanzen aufnehmen kann. Bei sehr großen Datenbasen kann das zu zeitintensiven Festspeicherzugriffen führen.

Bei inkrementellen Methoden besteht jedoch das Risiko, daß die Ergebnisse von der Reihenfolge der Verarbeitung abhängig werden. Zumindest für die Anwendung auf Datenbasen ist dieser Effekt unerwünscht. Zusätzlich erhöht sich dieses Risiko bei Datenbasen, die sortiert vorliegen.

Prägnanz der Merkmale

Nicht überschneidungsfreie Begriffsstrukturen benötigen für alle Merkmale eines Begriffes ein Maß für die Prägnanz dieses Merkmals.

Ein Maß, das bereits in Kapitel 3.2.2 erwähnt wurde, ist die Wahrscheinlichkeit, mit der dieses Merkmal bei den Instanzen beobachtet wird, zu denen der Begriff paßt. Das System COBWEB von Fisher benutzt diese Meßgröße.[22] Dabei bleibt aber unberücksichtigt, wie stark die Abweichungen der Instanzen bezüglich dieses Merkmals sind.

Dieser Mangel kann dadurch behoben werden, daß man eine statistische Verteilung, zum Beispiel die Normalverteilung, unterstellt und den Mittelwert und die Standardabweichung bestimmt. Der Mittelwert wird als Wert des untersuchten Attributes zum Merkmal. Die Standardabweichung gibt die Prägnanz dieses Merkmals an. Je kleiner die Standardabweichung, desto prägnanter das Merkmal. Gehört das Merkmal zu einem Begriff B, so gehen in die Berechnung natürlich nur diejenigen Instanzen ein, zu denen B paßt. CLASSIT[23], das Modell von Gennari, Langley und Fisher, verfolgt diesen Ansatz.

Ein eher pragmatischer Ansatz, um die Prägnanz zu bewerten, ist die Einführung eines Punktesystems. Jeder Instanz wird eine Anzahl von Punkten, ein score, zugeordnet. Instanzen, die ein Merkmal bestätigen, erhöhen den score. Instanzen, die ein Merkmal widerlegen, erniedrigen den score. Die Werte, um die erhöht oder erniedrigt wird, müssen nicht gleich groß sein.

[22] Kurzbeschreibung von COBWEB in Gennari et al. [23], Kapitel 3.3; ausführlicher in Fisher [17], [18]
[23] Gennari et al. [23] , Kapitel 4

Für die Punkte können Extremwerte vorgegeben sein. Wird das Minimum der Punktzahl erreicht, so wird das Merkmal als zu individuell erkannt. Bei Erreichen der maximalen Punktzahl kann man die weitere Untersuchung der Prägnanz dieses Merkmals einstellen. Es wird dann als sicher angesehen, daß es ein notwendiges Merkmal für den Begriff ist.

3.3 Realisierung in UNIMEM

UNIMEM ist eine Abkürzung für UNIversal MEMory model. Es ist ein System, das conceptual clustering benutzt, um aus Beispielen oder Beobachtungen Begriffe zu konstruieren.

Es wurde Mitte der 80er Jahre von Lebowitz an der Columbia Universität in New York erstellt. Obwohl es, wie auch alle weiteren hier genannten Realisierungen der Begriffsstrukturierung, ursprünglich nicht zur Wissensgewinnung aus großen Datenbasen konzipiert wurde, arbeitet es doch auf einer Sammlung von Daten, die als Menge von n-Tupeln angesehen werden kann.

Am Beispiel von UNIMEM soll eine der möglichen Vorgehensweisen des conceptual clustering exemplarisch vorgestellt werden.

3.3.1 Einordnung

Die Entwicklung von UNIMEM fand im Umfeld natürlichsprachlicher Systeme statt. Sein Vorläufer IPP analysierte Texte über terroristische Anschläge und versuchte spezifisches Wissen darüber zu erwerben. In UNIMEM wurde ein allgemeinerer Ansatz verfolgt, der es erlaubt, das System in beliebigen Umgebungen einzusetzen.

Aus der Datensammlung erzeugt UNIMEM eine nicht überschneidungsfreie Begriffshierarchie. Es arbeitet auf symbolischen und numerischen Domänen mit einem inkrementellen Verfahren. Die Höhenfunktion für den Bergsteiger-Algorithmus wird nicht explizit angegeben, sondern ergibt sich daraus, daß nur verbesserte Begriffshierarchien erzeugt werden.

Die Prägnanz der Merkmale wird mit der Punkte-Methode ermittelt. Für die Berechnung des Abstandes[24] zwischen Begriff und Instanz werden sowohl die Anzahl der passenden Merkmale als auch die numerische Differenz bei Zahlenwerten verwendet.

In UNIMEM wird ein pragmatischer Ansatz verfolgt. Alle wichtigen Parameter, die die Auswertung beeinflussen, können vom Benutzer eingestellt werden.

3.3.2 Speichermodell

UNIMEM arbeitet mit einem sogenannten Generalization Based Memory (GBM). Der Aufbau der Begriffsstruktur erfolgt durch die sukzessive Einordnung der Instanzen in den GBM.

[24] Die Abstandsfunktion ist programmtechnisch nicht als solche realisiert. Sie ist nur ein gedankliches Konstrukt, das zur Klarheit der Darstellung beitragen soll.

Der GBM bildet einen Baum.[25] Jeder innere Knoten des Baumes beschreibt einen Begriff. Die Instanzen bilden die Blätter des Baumes.[26]

Die Instanzen sind jeweils Söhne des speziellsten Begriffes, der zu ihnen paßt. Jeder Oberbegriff B_o ist Vater seines Unterbegriffes B_u erster Ordnung.[27] Oberbegriffe können gleichzeitig Väter von Unterbegriffen und Instanzen sein.

Wenn kein Begriff existiert, der Oberbegriff für alle Instanzen ist, wird ein leerer Begriff als Wurzel erzeugt.

In jedem Knoten des Baumes werden nur diejenigen Merkmale des Begriffes, beziehungsweise der Instanz, gespeichert, die nicht Merkmale der zugehörigen Oberbegriffe sind. Dadurch wird der Speicherplatz reduziert.

Diese Vorgehensweise hat Parallelen zu der Vermeidung von Redundanz bei relationalen Datenmodellen durch Normalisierung. Merkmale werden ja dann zu Begriffen zusammengefaßt, wenn sie häufig gemeinsam auftreten. Das weist auf strukturelle oder gar funktionale Abhängigkeiten der betroffenen Merkmale hin. Wenn funktionale Abhängigkeiten ganze Attribute überdecken, dann werden diese Attribute bei der Normalisierung relationaler Datenbasen in einzelne Relationen zusammengefaßt.[28] Im GBM werden sie in Begriffe zusammengefaßt.

3.3.3 Ablauf

UNIMEM speichert zu jedem Merkmal eines Begriffes zwei Werte: einen predictability score (Eignung) und einen confidence score.

Der confidence score entspricht der bereits erwähnten *Prägnanz* eines Merkmals. Er ist ein Maß für das Vertrauen (engl. confidence) darauf, daß eine Instanz, die zu diesem Begriff gehört, dieses Merkmal trägt.

Der predictability score bewertet dagegen die generelle Eignung dieses Merkmals zur Begriffsbildung. Er zählt, wie häufig das Merkmal in Unterbegriffen erster Ordnung verwendet wird. Das soll eine Vorhersage (engl. prediction) über die *Eignung* des Merkmals zur Begriffsbildung ermöglichen. Für die Eignung gilt: ist der Wert zu klein, so scheint es unsicher, ob das Merkmal gut für die Begriffsbildung geeignet ist. Wenn der Wert zu groß ist, also die Unterbegriffe den Begriff zu stark aufspalten, so ist dies ebenfalls unerwünscht.

Die obigen Bezeichnungen sind leider nicht durchgängig. Die englischen Bezeichnungen stammen von Lebowitz.[29] Gennari et al., die über UNIMEM schreiben, setzen confidence und predictability gleich[30] und bezeichnen die Prägnanz als predictability score. Die Eignung nennen sie predictiveness[31].

[25] nach Lebowitz [33], Seiten 26-28. Die Darstellung wurde leicht verändert.

[26] Genauer gesagt werden die Instanzen etwas anders gespeichert als die Begriffe. Dieser Unterschied wird zur Vereinfachung nicht beachtet.

[27] B_u ist dann Unterbegriff erster Ordnung, wenn es keinen weiteren Begriff B gibt, der Oberbegriff zu B_u ist und zu dem B_o Oberbegriff ist.

[28] Lockemann et al. [34], Kapitel 11.2.3 *Normalisierung*

[29] aus Lebowitz [33], Seiten 28/29

[30] Gennari et al. [23], Seite 23

[31] Gennari et al. [23], Seite 24

Diese Bezeichnungen eignen sich nicht gut zur Unterscheidung, weil sie das gleiche bedeuten: Vorhersagbarkeit.[32] Zur Klarheit werden bei der Beschreibung der Algorithmen daher die deutschen Begriffe verwendet.

Basisalgorithmus

Jede neue Instanz wird von UNIMEM in die bisher aufgebaute Begriffsstruktur eingeordnet. Während dieser Einordnung wird die Begriffsstruktur gleichzeitig verändert und angepaßt.[33]

Die Einordnung beginnt an der Wurzel. UNIMEM berechnet den Abstand der einzuordnenden Instanz zu allen Söhnen des aktuellen Knotens. Wenn dieser Abstand unterhalb einer benutzerdefinierten Schranke liegt, wird mit diesen Söhnen rekursiv fortgefahren.

Auf diese Weise kann eine Instanz beim Einordnen mehrere Pfade durch die Begriffshierarchie durchlaufen.

Für jeden aktuellen Knoten ruft UNIMEM die Funktion AUSWERTUNG auf. Für die Merkmale dieses Begriffes wird die Prägnanz aktualisiert.

Wenn ein aktueller Knoten außer Instanzen keine Söhne mehr hat, die zu der Instanz passen, so speichert UNIMEM diese Instanz prinzipiell als weiteren Sohn dieses Knotens und die Rekursion bricht ab. Vorher wird die Instanz jedoch mit allen anderen Instanzen verglichen, die Söhne des aktuellen Knotens sind. Wenn sie genügend viele gemeinsame Merkmale enthalten, dann erzeugt UNIMEM einen neuen Begriffsknoten, der durch die gemeinsamen Merkmale beschrieben wird. Dieser Knoten wird Sohn des aktuellen Knotens und bekommt als Söhne die beiden Instanzen, aus denen er erzeugt wurde. Weil dieser Vergleich mit allen bisherigen Sohn-Instanzen durchgeführt wird, können dabei mehrere neue Unterbegriffe entstehen.

Für alle neu entstandenen Begriffe müssen jeweils wieder die scores berechnet werden.

3.3.4 Anpassungen

Beim Verändern der Begriffsstruktur spielen die Werte Eignung und Prägnanz eine wichtige Rolle.

Die Funktion AUSWERTUNG wird für jeden aktuellen Knoten aufgerufen, d.h. die einzuordnende Instanz I paßt zu dem Begriff, der von dem aktuellen Knoten K beschrieben wird. Für alle Merkmale aus K, die noch nicht als "eingefroren" gekennzeichnet sind und die mit I übereinstimmen, wird deren Prägnanz um 1 erhöht. Die Prägnanz der anderen nicht eingefrorenen Merkmale wird um 1 erniedrigt.

[32] Die Bezeichnung stammt aus der Nutzung der Begriffe. Man versucht, beim Fehlen eines Attributes in einer Instanz, dessen Wert vorauszusagen, indem man die Instanz einem Begriff zuordnet und dessen Merkmale betrachtet. Für den Pawlow'schen Hund hat das Merkmal Futter in dem Begriff (Glocke, Futter) eine hohe Prägnanz. Wenn er die Instanz (Glocke, —) beobachtet, erwartet er deshalb Futter.

[33] Beschreibung des Algorithmus nach Gennari et al. [23]

Wenn die Prägnanz eine "Frostgrenze" wie z.B den Wert 15 überschreitet, wird das Merkmal eingefroren. Es gilt dann als sicher, daß es zur Begriffsbildung notwendig ist und sein Prägnanzwert wird in Zukunft nicht mehr verändert. Falls die Prägnanz unter einen Minimalwert sinkt, wird das Merkmal aus dem Begriff entfernt. Dadurch wird der Begriff allgemeiner. Durch das Löschen kann die Minimalzahl von Merkmalen für einen Begriff unterschritten werden. Dann wird dieser Begriff aufgelöst. Dazu werden seine Merkmale an seine Söhne weitergegeben, falls es sich bei diesen um Begriffe handelt.

Bei der Begriffsbildung in UNIMEM kann es zu dem Effekt kommen, daß Unterbegriffe in einem Merkmal von dem Merkmal eines Oberbegriffes abweichen. So kann ein Oberbegriff das Merkmal A(1)=100 und dessen Unterbegriff das Merkmal A(1)=105 tragen. Dadurch kann es beim Auflösen von Begriffen zu Konflikten kommen. In diesem Fall wird dieses Merkmal des aufgelösten Oberbegriffes nicht weitergegeben.

Die Eignung eines Merkmals wird immer mit dem neuen Anlegen eines Begriffes verändert. Der Oberbegriff des neuen Begriffes erhält neue Eignungswerte für seine Begriffe. Es werden die Eignungswerte aller übereinstimmender Merkmale erhöht. Wenn der Eignungswert eine obere Schranke erreicht, das Merkmal also zu viele Sohn-Begriffe mit diesem Merkmal hat, wird das Merkmal gestrichen.

3.3.5 Anwendungsgebiete

Lebowitz beschreibt die Anwendung von UNIMEM auf drei Anwendungsgebiete: auf Beschreibungen von Universitäten, auf das Abstimmungsverhalten von Kongreßabgeordneten und auf Daten über terroristische Anschläge, die aus Texten extrahiert wurden.[34]

Universitäten

Es wurden Daten über 230 Universitäten gesammelt. Diese Daten wurden zum Teil aus Standardquellen gewonnen und zum Teil durch Befragung der Studenten ermittelt. Neben objektiven Werten wie dem Anteil der Studenten, die finanzielle Unterstützung erhalten, gingen in die Merkmale auch subjektive Bewertungen ein. Dazu zählt etwa die Bewertung des "sozial life" auf einer Skala von 1 bis 5.

Bei der Auswahl der Attribute wurde nicht berücksichtigt, ob oder wie gut sich die Attribute zur Begriffsbildung eignen.

Nach 150 ausgewerteten Datensätzen hatte UNIMEM unter anderem einen Begriff für hochqualitative Privatuniversitäten erzeugt. Dieser Begriff bestand aus acht Merkmalen und hatte zwei Unterbegriffe, aber keine Instanzen als Söhne. Fünf der acht Merkmale waren "eingefroren", da ihre Prägnanz einen Maximalwert erreicht hatte.

Der Begriff war ursprünglich mit 10 Merkmalen angelegt worden. Zwei Merkmale wurden zwischenzeitlich gelöscht, da ihre Prägnanz unter den Minimalwert gesunken war. Auf diese Weise wurde der Begriff allgemeiner.

[34] Die Beschreibung der Anwendungsgebiete ist aus Lebowitz [33], Kapitel 3 *Example UNIMEM domains* zusammengefaßt.

Die erzeugten Unterbegriffe umfaßten fünf, beziehungsweise sechs Merkmale und verwiesen auf zwei, beziehungsweise vier Instanzen. Die Unterbegriffe ließen sich bezeichnen als teure Kunstakademien und hochselektive Universitäten. Dabei ist durchaus denkbar, daß eine Universität unter beide Begriffe einzuordnen ist.

Abtimmungsverhalten

Von den Abgeordneten im Kongreß der USA wurde festgehalten, wie sie bei wichtigen Gesetzesvorlagen abgestimmt haben. Zusätzlich wurden Daten über den Wahlbezirk berücksichtigt, den die Abgeordneten vertreten. Die Datenbasis umfaßte hundert Instanzen.

Von UNIMEM wurde erwartet, daß es Begriffe konstruiert, die ein liberales und ein konservatives Abstimmungsverhalten unterscheiden, was den politischen Gegebenheiten in den USA entspricht.

Problematisch war die Tatsache, daß man einer Gesetzesvorlage nur zustimmen oder sie ablehnen kann. Die Domänen für solche Attribute sind also nur zweiwertig. Dies führt prinzipiell zu einer starken Zersplitterung der Begriffsstruktur. Es werden zu viele Begriffe erzeugt, die zu speziell sind.

Dieses Problem wurde mit der Abstandsfunktion gelöst. Es wurde toleriert, daß eine Instanz nicht alle Merkmale eines Begriffes erfüllen muß, um diesem Begriff zugeordnet zu werden.

Terroristische Anschläge

IPP, ein Vorgänger von UNIMEM, untersuchte Texte über terroristische Anschläge, extrahierte die relevanten Daten und faßte die Anschläge nach gemeinsamen Merkmalen zusammen. In einem Experiment wurden die Daten von 370 Texten als Eingabe für UNIMEM benutzt, um die Unterschiede zu den bisherigen Auswertungen zu untersuchen.

Bei der intuitiven Beurteilung der Begriffshierarchien schnitt UNIMEM besser ab.[35] Es bildete eine umfassendere Begriffsstruktur, und die Grenzen zwischen den Einteilungen waren weniger scharf.

3.4 Stellungnahme

Die Anwendung von conceptual clustering zur Wissensgewinnung aus Datenbasen scheint mir prinzipiell möglich. Wenn das Verfahren jedoch mit wirklich großen Datenbasen funktionieren soll, so ist eine vorherige Analyse des Laufzeitverhaltens notwendig.

Ganz ohne Vorwissen über die Datenbasis ist das Verfahren leider nicht durchführbar. Mindestens über die Domänen der Attribute und die dort möglichen und sinnvollen Abstände müssen Informationen gegeben sein. Diese Informationen müssen vom Benutzer eingegeben werden. Alle benutzerdefinierten Parameter bergen allerdings die Gefahr,

[35] Lebowitz [33], Seite 32

die Werte so zu wählen, daß nicht das beste, sondern das gewollte Ergebnis erzielt wird. UNIMEM zeigt das deutlich: das Programm wurde jeweils so geändert und die Parameter so eingestellt, daß das gewünschte Ergebnis erzielt wurde. Die Vielseitigkeit der Anwendungsgebiete von UNIMEM und die Güte der dort erzielten Ergebnisse sind allerdings beeindruckend. Das läßt auf die Möglichkeit hoffen, UNIMEM auf allgemeine Datenbasen anwenden zu können.

Wenn beim conceptual clustering die menschliche Begriffsbildung nachvollzogen werden soll, so fehlt meines Erachtens eine vorherige empirische Untersuchung zum Beispiel in Form eines Experimentes: n Versuchspersonen geben zu einer Datenbasis deren Struktur an. Die Ergebnisse müßte man analysieren und untersuchen, ob sich mit maschinellen Lernverfahren ähnliche Ergebnisse erzielen lassen.

Sollte es tatsächlich gelingen, mit conceptual clustering große Datenbasen zu untersuchen, so scheint mir die Anwendung auf personenbezogene Daten problematisch. Neben den gesetzlichen Bestimmungen besteht der größte Schutz der persönlichen Daten darin, daß sie in der Masse der vorhandenen Daten untergehen. Dieser inhärente Schutz könnte verlorengehen und das Verfahren zum Mißbrauch verleiten. Da die Daten nach Gemeinsamkeiten gruppiert werden, würden im Ergebnis wohl Außenseiter von der "breiten Masse" getrennt werden. Das könnte wie bei Rasterfahndungen zu Ausgrenzung und Fehlverdächtigungen führen.

Kapitel 4

Focusing And Data Cleaning

Ralph Groß

Kurzfassung *Gemäß des Task-Modells von Wirth und Reinartz [56] ordnet man Focusing und Data Cleaning im gleichen Bereich innerhalb des Knowledge Discovery in Databases (KDD) ein : Vorverarbeitung für die Wissensgewinnung. Beide hier vorgestellten Ansätze verfolgen das Ziel, "interessante" Bereiche innerhalb der Datenbasis aufzufinden, damit nachgeschaltete Verfahren zur Wissensextraktion mit vernünftigem Aufwand durchgeführt werden können. Dabei liegt die Betonung des Data Cleanings besonders auf der Entfernung fehlerhafter oder irreführender Daten.*

4.1 Einführung

Alle Ansätze des KDD's stehen dem Problem gegenüber, daß die Verfahren zur Wissensextraktion nur eine relativ beschränkte Anzahl von Dateneinträgen mit akzeptablem Aufwand verarbeiten können. Bei den meisten Datenbasen ist es deshalb unumgänglich, den Ausgangsbereich der Algorithmen zu verkleinern. Man muß daher die Datenmenge auf die "interessantesten" Einträge einschränken, möglichst ohne dabei Wissen zu verlieren. Das Focusing nach Bhandari (siehe [3]) enthält neben diesem Schritt als wesentlichen Bestandteil die Präsentation der selektierten Daten, da hier der Mensch die Wissensgewinnung durchzuführen hat. Das Data Cleaning nach Guyon et al. (siehe [26]) beschränkt sich auf das Bestimmen der in einem noch zu definierenden Sinne informativen Dateneinträge. Diese Menge umfaßt allerdings zusätzlich fehlerhafte oder "unsaubere" Daten, die nach manueller Inspektion aus der Datenbasis entfernt werden können.

4.2 Focusing

4.2.1 Einführung

Bisherige Vorgehensweisen des KDD's konnten zumeist einer der beiden folgenden Kategorien zugeschlagen werden:

1. Automatische Wissensgewinnung durch ein Computerprogramm
2. Halbautomatische Wissensgewinnung, bei der ein durch Menschen geführtes Computerprogramm Wissen entdeckt

Die Beschäftigung mit halbautomatischen Verfahren ist notwendig, da sich abzeichnet, daß der Mensch nicht vollständig aus dem Prozeß der Wissensgewinnung eliminiert werden kann. Die Anwendung solcher Verfahren setzt allerdings explizite Kenntnisse aus dem Bereich der Datananalyse voraus. Um nun auch Laien auf diesem Gebiet adäquates Arbeiten mit ihren Datenbasen zu ermöglichen, empfiehlt sich ein dritter Ansatz:
maschinenunterstützte Wissensgewinnung

Im Unterschied zum zweiten Verfahren will man hier mit Hilfe des Rechners potentiell interessante Bereiche einer Datenbasis herausziehen und dem Anwender präsentieren. Dieser kann (soll) dann aus den vorgelegten Daten seine Schlüsse ziehen und damit neues Wissen gewinnen. Für alle Ansätze gilt, daß die Verfahren zur Wissensextraktion nicht funktionieren, wenn sie auf Datenbasen arbeiten sollen, die eine bestimmte Größe überschreiten. Es ist daher notwendig, die Datenmengen, nach Möglichkeit ohne Verlust von inhärentem Wissen, zu verkleiner. dritten Ansatz zuordnen.

4.2.2 Datenmodell

Alle weiteren Betrachtungen sollen an Hand eines relationalen Datenmodells erfolgen. Dieses in Form einer Tabelle zu veranschaulichende Modell enthält die Datentupel als Reihen und die unterschiedlichen Attribute als Spalten. Es wird vorausgesetzt, daß die Attribute diskrete Werte enthalten. Diese Forderung bedeutet allerdings keine Informationsverlust, da Verfahren existieren, die kontinuierliche Attributwerte in diskrete konvertieren.
Die vorausgesetzte Datenbasis D umfasse insgesamt N verschiedene Attribute. Zu den Attributen $A_1, A_2, \ldots, A_n$ seien die zugehörigen Mengen der Attributwerte mit $a_1, a_2, \ldots, a_n$ bezeichnet ($1 \leq n \leq N$). Damit läßt sich also eine Teilmenge der Datenbasis durch Angabe einer Liste von Attribut-Werte Paaren spezifizieren : $\{A_i = a_i : 1 \leq i \leq n\} \subset D$.
Man setzt dann DS_n als die Menge aller solcher Listen der Länge n. Damit ergibt sich DS, die Menge aller Teilmengen der Datenbasis, aus:

$$DS = \bigcup_{1 \leq n \leq N} DS_n \tag{4.1}$$

Im Folgenden genügt es, nur die Teilmengen DS_1 und DS_2 zu betrachten, wie sie in der Tabelle 4.1 veranschaulicht sind.

4.2.3 Komponenten des Focusings

Interestingness Functions

Im ersten Schritt des *Focusings* werden die Elemente von DS mit Hilfe einer (oder mehrerer) *Interestingness Functions* geordnet. Dieser vollständig vom Rechner ausgeführte

Records	A_1	A_2	A_3	...	A_n
R_1	x_1	–	–	...	–
R_2	x_2	–	–	...	–
R_3	–	–	–	...	x_3
R_4	–	x_4	–	...	–
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
R_m	–	x_m	–	...	–

Records	A_1	A_2	A_3	...	A_n
R_1	x_1	y_1	–	...	–
R_2	–	x_2	y_2	...	–
R_3	x_3	–	y_3	...	–
R_4	x_4	y_4	–	...	–
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
R_m	–	–	x_m	...	y_m

Tabelle 4.1: **Teilmengen DS_1 und DS_2.** *Man beachte, daß jeweils nur ein bzw. zwei Attribute eines Records gesetzt sind. Die x - und y - Werte seinen den entsprechenden Mengen $a_1, a_2, \ldots, a_n$ entnommen.*

Vorgang soll die Datenelemente entsprechend ihrer Wichtigkeit sortieren. Formal definiert man die Klasse I der *Interestingness Functions* wie folgt:

$$I := \{i : S \subset DS \rightarrow R^+\},$$

wobei R^+ die Menge der positiven reellen Zahlen ist.
Die Ordnung von R^+ wird mit Hilfe der Funktionen aus I auf S übertragen:

$$s_1 \; op \; s_2 :\Leftrightarrow i(s_1) \; op \; i(s_2) \text{ mit } s_1, s_2 \in S, op \in \{<, >, =\}$$

Beispielhaft seien hier zwei Funktionen aus I definiert:

- Mit Hilfe der Funktion I_1 sollen ungewöhnliche Anhäufungen eines Attributwertes erkannt werden. Dazu wird jedem Attribut-Werte Paar die Differenz zwischen der Wahrscheinlichkeit des Auftretens dieses Wertes und der bei Gleichverteilung statistisch zu erwartenden Wahrscheinlichkeit zugeordnet.

  $$I_1 : DS_1 \rightarrow R^+, \text{ Attribut } A_i, 1 \leq i \leq N$$

  Mit $N(A_i = \nu)$ als der Anzahl der Records der Datenbasis, in denen A_i den Wert ν hat und $N(A_i)$ als der Anzahl der Records, in denen A_i überhaupt einen Wert hat, definiert man die Wahrscheinlichkeit, daß A_i den Wert ν hat zu:

  $$P(A_i = \nu) := \frac{N(A_i = \nu)}{N(A_i)}. \tag{4.2}$$

  Weiterhin sei $Choice(A_i)$ die Anzahl der möglichen Werte von A_i. Damit wird nun I_1 definiert als:

  $$I_1(A_i = \nu) := \left| P(A_i = \nu) \Leftrightarrow \frac{1}{Choice(A_i)} \right| \tag{4.3}$$

- Die Funktion I_2 versucht, interessante Kombinationen aus zwei Attributen zu entdecken. Sie ordnet dazu zwei Attribut-Werte Paaren die Differenz zwischen der Wahrscheinlichkeit des gemeinsamen Auftretens und dem Produkt der Wahrscheinlichkeiten der einzelnen Attribut-Werte Paaren zu. Im Fall statistischer Unabhängigkeit der Attribute wäre diese Differenz gleich Null.

  $$I_2 : DS_2 \rightarrow R^+, \text{ Attribute } A_i, A_j, 1 \leq i, j \leq N$$

Die Bezeichnungen $N(A_i = \nu, A_j = \mu)$ und $N(A_i, A_j)$ seien analog zu oben, so daß $P(A_i = \nu, A_j = \mu)$ definiert wird zu:

$$P(A_i = \nu, A_j = \mu) := \frac{N(A_i = \nu, A_j = \mu)}{N(A_i, A_j)} \tag{4.4}$$

Dann setzt man I_2 wie folgt:

$$I_2(A_i = \nu, A_j = \mu) := |P(A_i = \nu, A_j = \mu) \Leftrightarrow P(A_i = \nu)P(A_j = \mu)| \tag{4.5}$$

Filtering Functions

Im zweiten Schritt wird die geordnete Liste der Datentupel, die aus der Anwendung einer oder mehrerer *Interestingness Functions* resultiert, weiter verarbeitet. Sogenannte *Filtering Functions* wählen die gemäß dieser Ordnung führenden Elemente aus und präsentieren sie dem Anwender in geeigneter Form.
Sei also F die Klasse der *Filtering Functions*. Ein $f \in F$ stellt die geordneten Datentupel in Form von nicht mehr als Y Tabellen mit jeweils maximal X zusammenhängenden Tupeln dar. Dabei wird mit Hilfe einer binären Funktion R festgestellt, ob zwischen zwei Datenelementen eine Verbindung besteht oder nicht: $R : DS \times DS \rightarrow \{0, 1\}$. Wenn man voraussetzt, daß Y, X und R geeignet gewählt sind, werden die Datentabellen wie folgt gebildet:
Sei L eine Liste von Elementen aus DS, wie sie aus der Anwendung einer *Interestingness Function* resultiert. Das Datum mit der höchsten Bewertung stehe dabei an erster Stelle. Die Tabelle T wird nun mit Hilfe einer rekursiven Vorschrift erstellt:

1. Entferne das erste Element e_1 aus L und plaziere es in T.
2. Durchsuche L von der Spitze aus nach Elementen $e_2, \ldots, e_X$, die mit e_1 gemäß R zusammenhängen, entferne sie aus L und schreibe sie nach T.
3. Falls noch keine Y Tabellen gebildet wurden und L nicht leer ist, beginne wieder bei 1., sonst beende den Vorgang.

Falls mehrere *Interestingness Functions* verwendet wurden, ergeben sich zwei Möglichkeiten des Vorgehens:

- Die geordneten Listen der verschiedenen Funktionen werden zu einer Liste zusammengefaßt
- Die Listen bleiben getrennt, und für jede der Listen wird ein eigener Satz von Tabellen generiert. Dabei ist Y aber die Gesamtanzahl **aller** Tabellen.

Hierbei stellen Y, X und R Parameter im Model der *Filtering Functions* dar, die für jede Anwendung individuell angepaßt werden müssen. Es lassen sich trotzdem gewisse Richtlinien für deren Wahl angeben, die sich vor allem an dem orientieren, was man über Möglichkeiten und Grenzen der menschlichen Informationsverarbeitung weiß. Demnach

sollte eine Tabelle nicht mehr als 7 Einträge enthalten und die Anzahl der Tabellen im Bereich zwischen 3 und 36 liegen (für eine Begründung der Werte siehe [3]). Damit der Anwender neue Erkenntnisse aus der Betrachtung einer Tabelle gewinnen kann, müssen deren Einträge miteinander in Beziehung stehen. Daher kann man an R sicherlich die Minimalforderung stellen, daß gemäß dieser Funktion zusammengehörige Tupel zumindest ein Attribut gemeinsam haben. Eine mögliche Realisierung von R wäre daher ein Join–Prädikat.

Interpretationsmodell

Im abschließenden Schritt des *Focusings* muß der Anwender aus den Tabellen der *Filtering Functions* geeignete Schlußfolgerungen ziehen. Um diesen Schritt nicht der puren Intuition des einzelnen anheim zu stellen, wird ein Interpretationsmodell vorgeschlagen, das es dem Analysten leichter machen soll, Wissen aus den generierten Tabellen zu gewinnen. Der Analyst muß, die Bedeutung der Attributwerte eines Tabelleneintrags ausnutzend, den von einer *Interestingness Function* gelieferten Wert mit der zugrunde liegenden Situation verbinden.
Dazu muß

(a) die **Ursache**

(b) die **Auswirkung**

der Tatsache, daß ein Element einen bestimmten Wert unter einer *Interestingness Function* aufweist, verstanden werden. Falls, beispielsweise nach Verwendung der Funktion I_2, das Paar $(A_1 = \nu)$ mit $(A_3 = \mu)$, aber nicht mit $(A_3 = \xi)$ assoziiert ist, müßte sich der Anwender fragen:

(a)
- Warum taucht $(A_1 = \nu)$ häufig mit $(A_3 = \mu)$, aber selten mit $(A_3 = \xi)$ auf ?
- Welches Ereignis kann zu solch einer Anhäufung führen ?

(b)
- Ist es wünschenswert, daß $(A_1 = \nu)$ häufig mit $(A_3 = \mu)$, aber selten mit $(A_3 = \xi)$ auftaucht ?
- Was passiert, wenn keine weiteren Schritte unternommen werden ?

Nach Betrachten einer Tabelle ergeben sich für den Anwender folgende Möglichkeiten:

1. Das neu erworbene Wissen wird direkt in Aktionen umgesetzt.
2. Es wurden lediglich bekannte Tatsachen zu Tage gefördert oder die neu entdeckten Zusammenhänge haben keine praktische Relevanz, so daß nichts unternommen werden muß.
3. Der hohe Wert eines Tabelleneintrags unter der *Interestingness Function* ist nicht verständlich. Der zugehörige Record muß deshalb näher untersucht werden.

Frage I	Frage II	gemessene Wahrsch.	erwartete Wahrsch.
Testfälle	schwache Exploitation der Funktion		
66%	32%	27%	21%
Testfälle	Problem nicht erkannt		
66%	47%	36%	31%

Tabelle 4.2: *Gefilterte Ausgabe der Interestingness Function I_2 mit*
Frage I: Was ist der beste Weg, um ähnliche Probleme zu finden ?
Frage II: Warum blieb das Problem verborgen ?

Falls die dritte Option zutrifft, ist noch vertiefende Arbeit des Analysten erforderlich. Diese Untersuchungen laufen dann aber zielgerichtet innerhalb eines bestimmten Teils der Datenbasis ab. Man kann daher mit größerer Wahrscheinlichkeit damit rechnen, daß hierbei neue Erkenntnisse gewonnen werden.
Der gesamte Vorgang des *Focusings* ist in Abbildung 4.1 veranschaulicht.

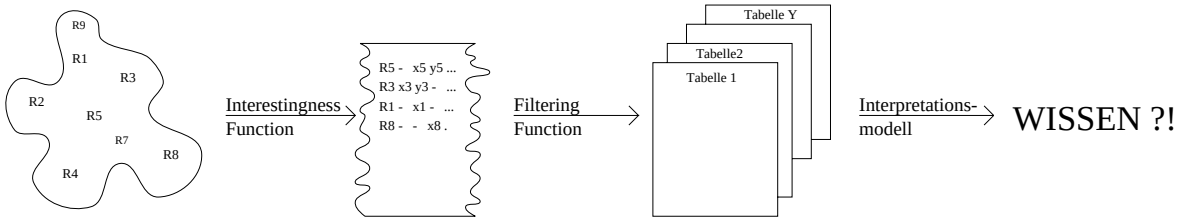


Abbildung 4.1: *Schritte des Focusings*

4.2.4 Anwendungsbeispiel

Das hier vorgestellte Verfahren wurde vom Autor mit Erfolg beim Prozeß der Softwareerstellung eingesetzt. Man verwendete dabei eine Datenbasis aus klassifizierten Fehlern, die im Laufe verschiedener Softwareprojekte bei IBM aufgetreten sind. So wurde beispielsweise für jeden, während des Betriebs der Software aufgetretenen Fehler, ein Fragebogen ausgefüllt, um Hinweise zu bekommen, wie Fehler dieser Art aus dem nächsten Release eliminiert werden können. Tabelle 4.2 enthält ausschnittsweise die gefilterte Ausgabe der *Interestingness Function I_2*.

Die erste Frage ("Was ist der beste Weg, um ähnliche Probleme zu finden ?") wurde also zu 66% mit "Testfälle", die zweite ("Warum blieb das Problem verborgen ?") zu 32% mit "schwacher Exploitation der Funktion" beantwortet. Diese beiden Antworten tauchten in 27% der Fälle gemeinsam auf, während die erwartete Prozentzahl bei $0.66 * 0.32 = 0.21 \mathrel{\hat{=}} 21\%$ liegt. Es wird also nahegelegt, daß eine Verbindung besteht zwischen Fehlern, die durch zusätzliche Testfälle gefunden werden können und der Tatsache, daß solche Fehler auf Grund schwacher Exploitation der Funktion verborgen blieben. Die **Ursache** gemäß des Interpretationsmodells liegt also in einer schwachen Abdeckung

der zu untersuchenden Funktion durch das Test-Tool. Daher würden durch Generierung zusätzlicher Testfälle mit den bisherigen Methoden keine weitere Fehler dieser Art entdeckt (**Auswirkung**). Als Schlußfolgerung verwendete man ein verbessertes Test-Tool.

4.2.5 Schlußbetrachtung

Die Untersuchungen des Autors zeigen, daß die hier vorgestellte Form des *Focusings* neues, vorher unbekanntes Wissen zu Tage fördert. Dabei ist der auf Seiten des Anwenders zu leistende Aufwand vergleichsweise gering. Die praktisch verwertbaren Ergebnisse liegen dem Autor zufolge trotzdem über dem, was durch andere Wissensgewinnungstechniken bisher gezeigt wurde. Der Grund dafür dürfte in der Art des Einsatzes des Analysten liegen. Es wurde schon von anderen Autoren erkannt (siehe [20]), daß in der näheren Zukunft Interaktive Systeme, die den Menschen in den Wissensgewinnungsprozeß einbeziehen, die besten Leistungen erbringen werden, da dieser Ansatz die Stärken sowohl des Menschen als auch der Rechner kombiniert: **menschliche Urteilskraft auf der einen und Computerleistung für Suche und Zahlenverarbeitung auf der anderen Seite.** Das Besondere im hier behandelten Ansatz liegt in der Tatsache, daß der Rechner den Menschen führt und nicht umgekehrt. Damit wird verhindert, daß der Anwender nur in bestimmten Bereichen der Datenbasis sucht. So können auch völlig unerwartete Verbindungen aufgedeckt werden. Man darf dabei die Auswirkungen der Darstellungsart der einzelnen Tabellen nicht unterschätzen. Nur wenn die vom Rechner ausgewählten Daten in der richtigen Form präsentiert werden, kann sie der Mensch sinnvoll beurteilen. Als positiv ist darüber hinaus zu vermerken, daß auch Laien das Verfahren anwenden können. Andererseits ergeben sich aber auch Kritikpunkte:

- Die verwendeten Funktionen sind sehr einfach und decken daher mathematisch gesehen nur sehr einfache Beziehungen auf.
- Die Ausrichtung auf den menschlichen Analysten macht es praktisch unmöglich, weitere maschinelle Schritte an die *Filtering Functions* anzuschließen.
- Dem Anwender wird sehr viel Verantwortung im Wissensgewinnungsprozeß übertragen.

Insgesamt gesehen halte ich den Ansatz durchaus für tauglich. Die Leistungsfähigkeit der *Interestingness Functions* müßte allerdings noch deutlich erhöht werden, um auch tieferliegende Beziehungen zwischen den Daten aufzuspüren.

4.3 Data Cleaning

4.3.1 Einführung

In Datenbasen sind oft redundante Daten vorhanden. Es wäre daher günstig, wenn man eventuell sehr große Datenmengen durch eine kleinere Teilmenge ersetzen könnte, die nur

noch die "wesentlichen" Daten enthält, die sogenannten *informativen Muster*. Es ist dabei allerdings durchaus schwierig festzulegen, was genau man darunter verstehen will. Der hier vorgestellte Ansatz legt folgende Definition zugrunde:
Ausgehend von einem Modell, das auf einer Sequenz von Mustern trainiert wurde, wird ein neues Muster als *informativ* eingestuft, wenn es schwierig ist, dieses Muster mit dem beschriebenen Modell vorherzusagen, das heißt zu klassifizieren.
Es ist einleuchtend, daß neben den wirklich interessanten Mustern, durch diese Definition auch "schlechte" Daten eingeschlossen werden, also Daten, die auf Grund von Fehlern in die Datenbasis gekommen sind. Die hier vorgestellten Algorithmen zur Identifizierung von *informativen Mustern* können dann dazu verwendet werden, derartige untaugliche Muster aus der Datenbasis zu entfernen.

4.3.2 Entdeckung von informativen Mustern

Informationstheoretische Motivation

Diesem Abschnitt sei eine konkrete Datenbasis zugrunde gelegt. Sie besteht aus handgeschriebenen, vorklassifizierten Nullen und Einsen, das heißt für jedes Datum ist bekannt, welche Ziffer es enthält. Dabei sei garantiert, daß keine fehlerhaften Daten enthalten sind. Das Ziel besteht nun darin, für jedes Muster ein Maß zu bestimmen, das angibt, wie informativ es ist. In Bezug auf die angeführte Datenbasis ist ein Muster gerade dann informativ, wenn es eine neue Form der Schreibweise darstellt, also neue Merkmalskombinationen enthält. Es sollte in solch einem Fall schwierig sein, aus der Kenntnis der vorhergehenden Muster, das gegebene als Null oder Eins zu klassifizieren.
Das gesuchte Maß läßt sich gewinnen, wenn man das Problem im Kontext der Informationstheorie betrachtet. Dazu stellt man sich vor, daß die Einträge der Datenbasis in Form einer Sequenz aufeinander folgender Muster generiert wurden. Jedes Datum kann als Ausgabe eines diskreten oder kontinuierlichen Zufallsprozesses gesehen werden. Man definiert dann den **Informationsgehalt** I einer Ausgabe x zu:

$$I(x) := \Leftrightarrow \log P(x) \tag{4.6}$$

wobei $P(x)$ die dem Ereignis a priori zugeordnete Wahrscheinlichkeit ist. Diese Begriffsbildung gilt es nun auf unsere Beispieldatenbasis zu übertragen:
Mit x_k werden die einzelnen Muster, mit $y_k \in \{0, 1\}$ die ihnen zugeordneten Klassen bezeichnet. Die a priori Wahrscheinlichkeit einer bestimmten Klasse für ein x_k entspricht dann der Wahrscheinlichkeit, mit der die richtige Klasse aus Kenntnis der vorangegangenen $k \Leftrightarrow 1$ Muster vorhergesagt wird. Wenn $\hat{y}_k$ die vorhergesagte Klasse für x_k ist, erhält man besagte Wahrscheinlichkeit als bedingte Wahrscheinlichkeit wie folgt:

$$P_k(\hat{y}_k = y_k) = P(\hat{y}_k = y_k | x_k, (x_0, y_0), (x_1, y_1), \ldots, (x_{k-1}, y_{k-1})) \tag{4.7}$$

Der Informationsgehalt eines Musters läßt sich dann natürlich analog zu Definition 4.6 setzen als:

$$I(k) = \Leftrightarrow \log P_k(\hat{y}_k = y_k) = \Leftrightarrow y_k \log P_k(\hat{y}_k = 1) \Leftrightarrow (1 \Leftrightarrow y_k) \log(1 \Leftrightarrow P_k(\hat{y}_k = 1)) \tag{4.8}$$

Die zweite Gleichung wird unmittelbar einsichtig, wenn man berücksichtigt, daß $y_k \in \{0, 1\}$ ist. $I(k)$ wird auch als $Informationskriterium$ bezeichnet.

Verfahren zur Berechnung des Informationskriteriums

Die Wahrscheinlichkeit $P_k(\hat{y}_k = 1)$ soll nun mit Hilfe von Techniken des Maschinellen Lernens approximiert werden. Wie oben werden der Maschine nacheinander Muster aus der Datenbasis vorgelegt. Sie versucht dann deren Klasse (Null oder Eins) vorauszusagen. Der dabei gemachte Fehler wird berechnet und dazu benutzt, die Genauigkeit weiterer Voraussagen durch Verändern der Parameter der Maschine zu verbessern. Man trainiert also eine einzelne Maschine, um eine Schätzung $\hat{P}_k(y_k = 1)$ der Wahrscheinlichkeit zu erhalten, daß die richtige Klasse "Eins" ist. Die Klasse $\hat{y}_k$ wird dann dementsprechend gewählt, so daß man in Gleichung 4.8 $P_k(\hat{y}_k = 1)$ durch $\hat{P}_k(y_k = 1)$ ersetzen kann.
Es bieten sich verschiedene Lerntechniken an, um diese Wahrscheinlichkeit zu berechnen:

- **k-nächster-Nachbar Klassifikator**
  Jedes neue Muster wird zunächst gespeichert. Dann erhält man $\hat{P}_k(y_k = 1)$ als den Anteil der mit Eins vorklassifizierten Muster unter den k nächsten Nachbarn von x_k.

- **Neuronales Netz**
  beispielsweise trainiert mit dem quadratischen Fehler $(y_k \Leftrightarrow \hat{P}_k(y_k = 1))^2$

Über die genannten Möglichkeiten hinaus, schlagen die Autoren (siehe [25] und [26]) eine Gruppe weiterer Verfahren, sogenannte **Minimax Algorithmen** vor, die zur Klasse der Batch Algorithmen gehören. Man unterscheidet dabei generell zwischen zwei Arten von Algorithmen:

- On-line Algorithmen

und

- Batch Algorithmen

Im ersten Fall ist das Vorgehen analog zu oben: die Muster werden in einer Sequenz präsentiert. Nach jedem Muster wird sowohl das Informationskriterium $I(k)$ berechnet als auch die Parameter der Maschine neu eingestellt. Man stuft dann ein Muster als informativ ein, wenn $I(k)$ einen vorher definierten Schwellwert überschreitet. Der Nachteil dabei ist allerdings, daß die Parametereinstellung der Maschine und damit auch das Informationskriterium von der Reihenfolge abhängt, in der die Muster vorgelegt werden. Für unterschiedliche Sequenzen erhält man unter Umständen verschiedene Mengen von informativen Mustern.
Im Unterschied dazu liegen bei Batch Algorithmen alle Daten auf einmal vor, so daß keine Abhängigkeit von einer Reihenfolge vorhanden ist. Falls die Datenbasis p Einträge enthält, müßte man p Maschinen mit $p \Leftrightarrow 1$ Mustern trainieren, um den Informationsgehalt des p-ten Musters feststellen zu können. Es leuchtet ein, daß dieses Verfahren in der Praxis nicht durchführbar ist. Man trainiert vielmehr eine einzige Maschine mit der

kompletten Datenbasis und approximiert das Informationskriterium eines einzelnen Musters mit einer Schätzung, um wieviel das über die gesamte Datenmenge aufsummierte Informationskriterium fallen würde, wenn man dieses Muster entfernt. Als kumuliertes Kriterium bietet sich in diesem Kontext natürlich der über die gesamte Datenmenge aufsummierte Klassifikationsfehler an. Der Informationsgehalt des einzelnen Musters ist sein Beitrag zu diesem Fehler. Man kann dann die Muster auf eindeutige Art und Weise nach ihrem Informationsgehalt ordnen und die gesamte Datenbasis durch die m informativsten Muster repräsentieren.

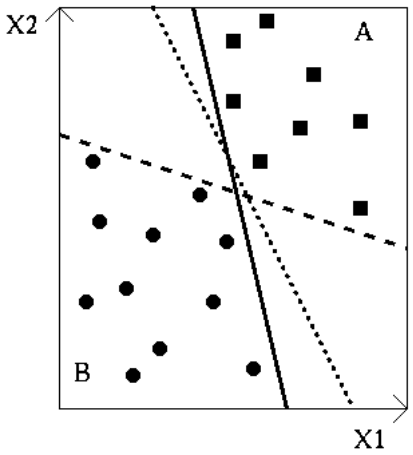


Abbildung 4.2: *Verschiedene Möglichkeiten, die Klassen A und B zu separieren*

Minimax Algorithmen

Die meisten Algorithmen trainieren eine Maschine durch Minimierung des durchschnittlichen Fehlers. Im Gegensatz dazu minimieren die Minimax Algorithmen den maximalen Fehler:

$$\min_{w} \max_{k} \; l(k) \tag{4.9}$$

wobei w den Parametervektor der Maschine darstellt, k über alle Daten läuft und $l(.)$ eine beliebige Fehlerfunktion ist. Ein auf das vorliegende Klassifikationsproblem zugeschnittener Minimax Algorithmus ist der **Optimum Margin Classifier** (OMC). Die vorliegende Aufgabe kann wie folgt formuliert werden:
Man suche eine Entscheidungsfunktion D, die für vorgelegte Muster x eines n-dimenionalen Merkmalsraumes feststellt, in welche der Klassen A oder B das Muster gehört. Im Trainingslauf werden bereits vorklassifizierte Daten vorgelegt (Klasse $g_k \in \{\Leftrightarrow 1, 1\}$), so daß die Parameter der Funktion eingestellt werden können. Danach ordnet man alle Muster x, für die $D(x) > 0$ ist, der Klasse A zu ansonsten der Klasse B. Wenn nur wenige Trainingsdaten vorhanden sind, gibt es in der Regel viele fehlerfreie Separationsmöglichkeiten (siehe Abbildung 4.2). Bei der Auswahl der Entscheidungsfunktion ist dann allerdings zu beachten, daß das eigentliche Ziel des Trainings eine gute Generalisierung ist, das heißt eine möglichst fehlerfreie Klassifikation von Mustern, die nicht in der Trainingsmenge enthalten sind. Man erwartet, daß ein neues Beispiel einer Klasse mit hoher Wahrscheinlichkeit in die konvexe Hülle der vorherigen Klassenvertreter fällt.

Daher sollte der Abstand der Trainingsmuster zur Klassengrenze maximiert werden (siehe Abbildung 4.3).

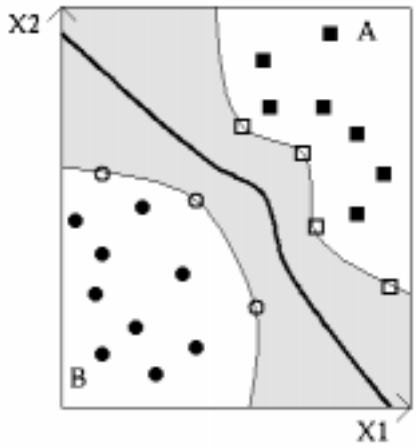


Abbildung 4.3: *Entscheidungsgrenze mit maximalem Abstand zu den Mustern. Die informativen Muster liegen auf der Grenze.*

Ein wichtige Eigenschaft dieses Ansatzes ist, daß die gewählte Grenze nur von einer beschränkten Anzahl der Trainingsmuster abhängt, und zwar von den Mustern, die am nächsten zu ihr liegen. Genau diese Muster werden dann als **informativ** bezeichnet.
Durch die Gleichung $D(x) = 0$ wird eine Hyperebene H in einem n-dimensionalen Merkmalsraum definiert. Der Parametervektor w ist ihr Normalenvektor. Es gilt nun, daß der Wert eines Merkmalvektors x unter der Entscheidungsfunktion proportional ist zum Abstand dieses Vektors von der Hyperebene H (siehe etwa [11]). Damit läßt sich der Rand, also der Abstand des am nächsten liegenden Trainingsmusters zur Hyperebene, berechnen aus:

$$M = \min_k \frac{g_k D(x_k)}{||w||} \tag{4.10}$$

Die Entscheidungsfunktion D ergibt sich dann als Lösung des folgenden Optimierungsproblems:

$$\max_w \min_k \frac{g_k D(x_k)}{||w||} \tag{4.11}$$

Das kann nun mit Blick auf die Minimax Algorithmen umgeformt werden in:

$$\min_w \max_k \left(\Leftrightarrow \frac{g_k D(x_k)}{||w||} \right) \tag{4.12}$$

Der OMC - Algorithmus löst genau dieses Problem. Man kann zeigen, daß die Lösung w^* eine Linearkombination von Basisfunktionen $\varphi(.)$ der informativen Muster ist (wobei insgesamt p Einträge in der Datenbasis vorhanden sind):

$$w^* = \sum_{k=1}^{p} g_k \alpha_k^* \varphi(x_k), \ \alpha_k^* \geq 0 \tag{4.13}$$

Dabei haben die Koeffizienten α_k^* nur für die informativen Muster einen Wert ungleich Null. Man benutzt daher die Summe der α_k^* als kumuliertes Informationskriterium:

$$I = \sum_{k=1}^{p} \alpha_k^* \tag{4.14}$$

Jedem Muster k wird dann gerade α_k^* als Informationsgehalt zugeordnet:

$$I(k) = \alpha_k^* \tag{4.15}$$

Der OMC-Algorithmus zeichnet sich schließlich noch durch folgende Eigenschaften aus:

- Der Vorhersagefehler auf einer Testmenge ist durch $\frac{m}{p}$ beschränkt, wobei m die Anzahl der informativen und p die Gesamtzahl der Muster ist. Man kann daraus ersehen, daß eine gute Generalisierung erreicht wird.
- Bei Experimenten mit einer Datenbasis aus handgeschriebenen Ziffern zeigte sich, daß die Anzahl der informativen Muster nur logarithmisch mit der Größe der Datenbasis wächst, wenn die Anzahl der informativen Muster klein ist im Vergleich zur Gesamtzahl der Muster.

4.3.3 Verfahren des Data Cleanings

Problemstellung

Die im letzten Abschnitt vorgestellte Definition von informativen Mustern liefert teilweise Ergebnisse, die sich nicht mit den Erwartungen decken, die man an sie geknüpft hat. Neben informativen Mustern können auch untaugliche Daten einen hohen Informationsgehalt zugeordnet bekommen. Während die guten Muster in der Datenbasis verbleiben sollen, würde man die schlechten Daten gerne entfernen. Ein rein automatisches Verfahren, das alle Muster mit zu hohem Informationsgehalt löscht, wäre zu gefährlich, da auch wertvolle Daten verloren gehen könnten. Daher wird im Folgenden ein computerunterstütztes Verfahren vorgestellt, bei dem ein menschlicher Operator nur die Muster untersuchen muß, die den höchsten Informationsgehalt haben und deshalb am verdächtigsten sind.

On-line und Batch Algorithmen

Für das *Data Cleaning* existieren ähnlich wie oben sowohl On-line als auch Batch Algorithmen. In Abbildung 4.4 ist die On-line Version veranschaulicht. Dabei wird der Klassifikator zunächst mit einigen sauberen Daten trainiert. Im k-ten Schritt des *Cleanings* wird das Muster x_k der Maschine präsentiert. Mit Hilfe der Vorhersage des Klassifikators ($\hat{P}_k(y_k = 1)$) und dem gewünschten Wert y_k wird das Informationskriterium $I(k)$ berechnet. Wenn dieser Wert unter einer gegebenen Schwelle Θ liegt, wird das Muster der Maschine direkt zur Parameteranpassung übergeben. Ansonsten muß es der menschliche Operator begutachten und entweder als "Müll" klassifizieren und entfernen oder ebenfalls dem Klassifikator zuführen. Wenn alle Daten verarbeitet wurden, ist sowohl das Training

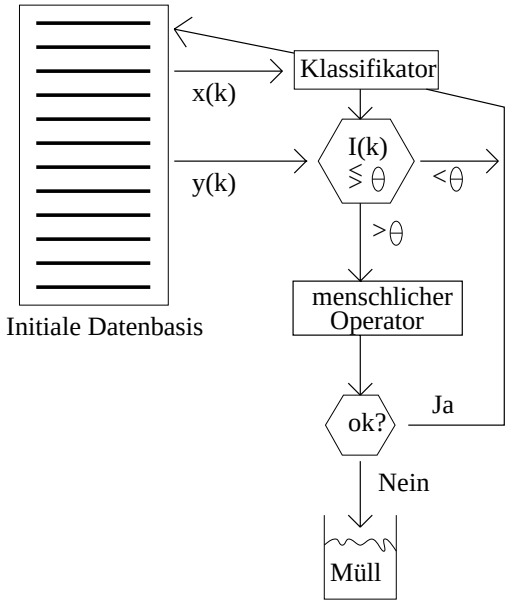


Abbildung 4.4: **Schema des On-line Cleanings.** *Im Verlauf des Lernens begutachtet ein menschlicher Operator die Muster, deren Informationskriterium die Schwelle* Θ *überschreiten. Die guten Muster verbleiben in der Datenbasis und werden dem Klassifikator zur Parameteradaption zugeführt, während man die schlechten entfernt.*

als auch das *Cleaning* abgeschlossen, da der Klassifikator nur mit sauberen Daten trainiert wurde. Falls gewünscht, kann der Klassifikator also direkt weiter eingesetzt werden. Das On-line Verfahren hat allerdings auch Nachteile:

- Man muß eine geeignete Schwelle Θ bestimmen.
- Das Verfahren hängt von der Reihenfolge der Muster ab. Es ist nicht möglich, die Entscheidung ob ein Muster gut oder schlecht ist, zu revidieren.
- Wenn das Trainieren des Klassifikators länger dauert als das Begutachten der Muster, wird die Zeit des menschliche Operators unnötig verschwendet.

Man kann aus dieser Aufstellung schon ersehen, daß — falls möglich — Batch Algorithmen vorzuziehen sind.

Dabei trainiert man, wie in Abbildung 4.5 veranschaulicht, zunächst auf allen Mustern (den schlechten eingeschlossen) mit einem beliebigen Trainingsalgorithmus. Anschließend wird, wie im letzten Abschnitt beschrieben, der Informationsgehalt jedes Musters bestimmt und die Datenbasis nach fallendem $I(k)$ sortiert. Der menschliche Operator arbeitet dann von oben, vom verdächtigsten Muster ausgehend, die Daten durch, bis die Anzahl aufeinander folgender guter Daten eine bestimmte Schwelle überschreitet. Danach wird der Klassifikator auf den gesäuberten Daten erneut trainiert. Es empfiehlt sich, als Trainingsalgorithmus den Minimax Algorithmus zu verwenden. Dann muß nicht die gesamte Datenbasis, sondern nur die informativen Muster untersucht werden.
Falls die Datenbasis korrelierte Fehler enthält, kann es notwendig sein, den gesamten Vorgang mehrfach zu wiederholen, um alle schlechten Daten zu entfernen.

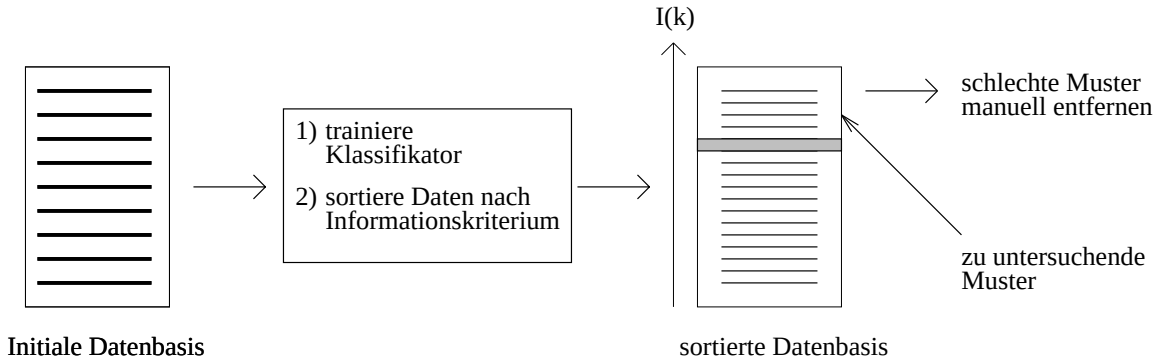


Abbildung 4.5: **Schema des Batch Cleanings.** *Der Klassifikator wird auf ungesäuberten Daten trainiert. Die Datenbasis wird dann gemäß des Informationskriteriums sortiert. Die am höchsten bewerteten Muster müssen vom menschlichen Operator untersucht werden. Nach dem Cleaning empfiehlt es sich, den Klassifikator nochmals zu trainieren.*

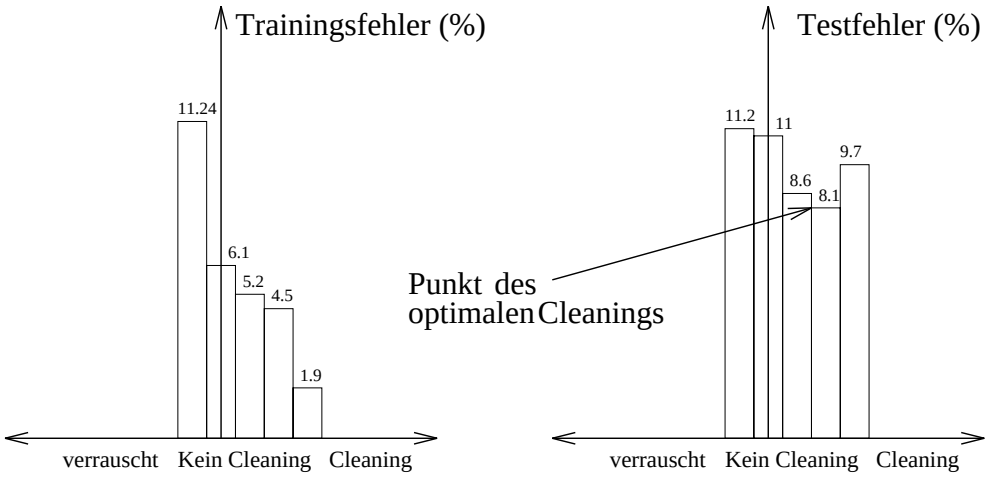


Abbildung 4.6: **Resultate des Data Cleanings mit Punkt des optimalen Cleanings**

Datenbasis	Größe	Anzahl der informativen Muster	Zeit für *Cleaning* (h)	% Fehler TM I	% Fehler TM II
OCR Zahlen	7300	690	1	15	10.5
on-line lower	16000	3130	1	11	6.5
on-line ASCII	100000	10983	5	30	6

Tabelle 4.3: ***Resultate der Anwendung des Cleanings auf mehrere Datenbasen*** *Dabei sind die Fehlerraten der letzten beiden Spalten die Fehler, die auf der Validationsmenge gemacht wurden. Im ersten Fall (TM I) wurde die unveränderte Trainingsmenge benutzt, im zweiten Fall (TM II) die Trainingsmenge nach Durchführung des Cleanings. Die Cleaningzeit ist die Zeit, die der menschliche Operator mit dem Durchsehen der vorgelegten Muster verbracht hat.*

Punkt des optimalen Cleanings

Es stellt sich jetzt natürlich die Frage, wie zuverlässig die vorgestellten Techniken funktionieren:

- Wurden alle Muster untersucht, die Kandidaten für das Entfernen sein sollten ?
- Wurden unter den begutachteten Muster zu viele oder zu wenige entfernt ?

Diese Fragen wurden von den Autoren in folgendem Kontext untersucht:
Sie trainierten ein Neuronales Netz für die Erkennung handgeschriebener Kleinbuchstaben. Die vorhandenen, gelabelten Muster wurden in eine Trainingsmenge und eine Validationsmenge geteilt, wobei man nur auf der Trainingsdatenbasis verschiedene Stufen des *Cleanings* durchführte. Zusätzlich verrauschten sie 5% der Muster dieser Menge künstlich durch Veränderung der Klassenzugehörigkeiten. Das *Cleaning* dieser Datenmenge fand nun in mehreren Schritten statt, wobei in jeder neuen Stufe die Toleranz seitens des menschlichen Operators gegenüber schlecht geschriebenen Buchstaben herabgesetzt wurde. Der Klassifikationsfehler bildete dabei das Informationskriterium. In Abbildung 4.6 sind die Fehlerraten auf der Trainingsmenge und der Validationsmenge dargestellt. Man erkennt, daß die Fehlerrate auf der Trainingsmenge mit steigender Anzahl entfernter Muster fällt. Das ist verständlich, da das zu lernende Klassifikationsverhalten einfacher wird, wenn weniger Muster zu berücksichtigen sind. Dagegen durchläuft der Fehler auf der Validationsmenge ein Minimum. Eine plausible Deutung dafür ist, daß zunächst nur wirklich schlechte Muster entfernt werden. Irgendwann beginnt man aber damit, informative Muster zu löschen, so daß dem Netz wertvolle Informationen verloren gehen. Dieses Verhalten des Validationsfehler läßt sich theoretisch vorhersagen, so daß hier nicht nur eine Einzelbeobachtung vorliegt. Man bezeichnet daher das Minimum des Fehlers auf der Validationsmenge als **Punkt des optimalen Cleanings**.
Abschließend sei noch bemerkt, daß man den Punkt des optimalen *Cleanings* auch auf andere Art und Weise bestimmen kann, ohne die Datenbasis in Trainings- und Validationsmenge unterteilen zu müssen. Man benutzt dabei die Vorhersagen der Vapnik-Chervonenkis Theorie (siehe [1] und [26]).

4.3.4 Schlußbetrachtung

In Tabelle 4.3 sind einige Ergebnisse des *Data Cleanings* auf verschiedenen Datenbasen zusammengefaßt. Man beachte dabei besonders, daß die Anzahl der informativen Muster und damit auch die Zeit zur Durchführung des *Cleanings* nur sublinear in der Anzahl der Daten wächst. Aus den Zahlen ist ersichtlich, daß die vorgestellten Techniken die Qualität der Daten, gemessen durch die Verbesserung in der Klassifikationsleistung auf einer unabhängigen Testmenge, steigert. Insbesondere die letzte Datenbasis wäre ohne Verwendung des *Cleanings* nicht brauchbar. Man muß allerdings auch bemerken, daß die Aufwandsbetrachtungen wesentlich ungünstiger ausfallen, wenn man für den eintrainierten Klassifikator über die Bestimmung des Informationsgehaltes hinaus keine Verwendung mehr hat. Insgesamt gesehen bleiben die Einsatzmöglichkeiten der vorgestellten Methoden auf den Rahmen von Klassifikationsprobleme, beschränkt.

Kapitel 5

Übersicht über Induktive logische Programmierung

Wolfgang Decker

Kurzfassung *Induktive logische Programmierung (ILP) kann als Maschinelles Lernen in einer prädikatenlogischen Sprache angesehen werden, in der Relationen im Kontext von deduktiven Datenbanken zur Verfügung stehen. Es ist auch interessant für die Wissensgewinnung in Datenbanken, da Zusammenhänge beschrieben werden können, die mehr als eine Relation miteinbeziehen. Die Ausarbeitung gibt einen Überblick über grundlegende ILP-Techniken, wie auch über andere ILP-Gebiete, die für die Wissensgewinnung in Datenbanken relevant sind. Zum Schluß wird noch auf KDD-Anwendungen von ILP eingegangen, eine davon wird dann etwas genauer besprochen: Vorhersage der sekundären Struktur von Proteinen.*

5.1 KDD und Maschinelles Lernen

Knowledge Discovery in Databases (KDD) beschäftigt sich mit dem Finden neuer und interessanter Zusammenhänge in Datenbanken und deren Beschreibung in kurzer und prägnanter Form. [22] erkannten, daß sich die Techniken des Maschinellen Lernens auch bei KDD-Problemen einsetzen lassen. Zunächst wurden attributorientierte Lernverfahren wie z.B. C4.5,ASSISTANT und CN2 auf KDD-Probleme angewandt [48, 9, 30]. Jedoch haben diese Ansätze enge Grenzen. Die von diesen Lernsystemen erkannten Zusamenhänge werden in einer aussagenlogischen Sprache ausgedrückt. Solche Sprachen sind jedoch beschränkt und erlauben keine Beschreibung komplex strukturierter Objekte oder von Beziehungen unter diesen. Das im Lernprozeß verwendbare Hintergrundwissen ist von stark beschränkter Form und es können keine zusätzlichen Relationen aus der Datenbank in den Lernprozeß miteinbezogen werden.

Neue Entwicklungen im Gebiet des Induktiven Lernens beschäftigen sich mit dem Problem, eine logische Definition einer Relation mit Hilfe von Beispieltupeln zu konstruieren, von denen man weiß, ob sie zur Relation gehören oder nicht. Hierbei können auch andere Relationen –sogenannte Hilfsrelationen– in der induzierten Definition verwendet werden.

Auf solchen Beschreibungen der Relationen basieren auch die deduktiven Datenbanksysteme. Die logischen Definitionen formen dort die sogenannte *intensionale* Datenbank. Im Gegensatz zur *extensionalen* Datenbasis müssen die Tupel einer Relation nicht einzeln aufgezählt werden, sondern können in kompakter Form als Regeln dargestellt werden. Zur Implementierung zieht man formale Systeme und logische Programmierung heran.

Die Induktion logischer Definitionen von Relationen kann man in diesem Rahmen als Synthese logischer Programme ansehen. Das Ganze nennt man dann seit kurzem Induktive logische Programmierung [37, 31].

5.2 Induktive logische Programmierung

5.2.1 Beispiel eines ILP-Problems

Trainingsbeispiele		Hintergrundwissen	
$\oplus$	Tochter(mary,ann).	Elternteil(ann,mary).	Weiblich(ann).
$\oplus$	Tochter(eve,tom).	Elternteil(ann,tom).	Weiblich(mary).
$\ominus$	Tochter(tom,ann).	Elternteil(tom,eve).	Weiblich(eve).
$\ominus$	Tochter(eve,ann).	Elternteil(tom,ian).	

Tabelle 5.1: Trainingsbeispiele und Hintergrundwissen

Veranschaulichen wir uns zunächst den Begriff ILP anhand eines einfachen Problems, dem Lernen von familiären Beziehungen. Aufgabe ist es, eine Beschreibung der Relation *Tochter(X,Y)* zu finden, welche besagt, daß die Person X eine Tochter der Person Y ist, und dies mithilfe der Relationen *Weiblich* und *Elternteil*. Diese Relationen stellen dabei Hintergrundwissen dar. Es sind zwei positive und zwei negative Beispiele für unsere *Ziel*-Relation *Tochter* gegeben. In einer logischen Programmiersprache sieht eine Definition unserer Zielrelation wie folgt aus:

$$Tochter(X,Y) \leftarrow Weiblich(X), Elternteil(Y,X).$$

Diese Definition ist eine Sicht, die die Zielrelation *intensional* hinsichtlich der Relationen des Hintergrundwissens definiert.

5.2.2 Deduktive Datenbanken und logisches Programmieren

Bevor wir nun ILP weiter besprechen, müssen wir zunächst ein paar grundlegende Ausdrücke aus den Gebieten der logischen Programmierung und der deduktiven Datenbanken einführen. Eine deduktive Datenbank (DDB) besteht aus einer Menge von *Datenbankklauseln*. Eine Datenbankklausel ist eine *typisierte Programmklausel* der Form :

$$p(X_1, \ldots, X_n) \leftarrow L_1, \ldots, L_m.$$

Hierbei ist der Kopf der Klausel ein einzelnes Prädikat. Der Rumpf der Klausel ist eine Konjunktion von positiven Literalen $q_i(Y_1, \ldots, Y_{n_i})$ und/oder negativen Literalen $\neg q_j(Y_1, \ldots, Y_{n_j})$. Der Hauptunterschied zwischen Programmklauseln und Datenbankklauseln besteht in der Verwendung von Typen. In typisierten Klauseln wird mit jeder vorkommenden Variablen ein bestimmter Typ in Verbindung gebracht. Der Typ einer Variablen bestimmt die Menge der Werte, die sie einnehmen kann.

Eine Menge von Programmklauseln mit dem selben Prädikatsymbol p im Kopf formen eine Prädikat-Definition. Ein Prädikat kann *extensional* als Menge von Fakten oder *intensional* als Menge von Datenbankklauseln beschrieben werden. Es wird vorausgesetzt, daß jedes Prädikat entweder extensional oder intensional definiert wird. In der folgenden Tabelle werden nochmal die Ausdrücke aus den Bereichen Datenbanken und logisches Programmieren gegenübergestellt.

DB-Ausdrücke	LP-Ausdrücke
Relationenname p	Prädikatsymbol p
Attribute der Relation p	Argumente des Prädikats p
Tupel $\langle a_1, \ldots, a_n \rangle$	Faktum $p(a_1, \ldots, a_n)$
Relation p –	Prädikat p –
eine Menge von Tupeln	extensional durch eine Menge von Fakten definiert
Relation q –	Prädikat q –
als Sicht definiert	intensional durch eine Menge von Regeln definiert

5.2.3 Ausprägungen von ILP

ILP-Systeme können anhand mehrerer Merkmale unterschieden werden.

1. Zuerst können sie entweder ein *einzelnes Konzept* (Prädikat) oder *mehrere Konzepte* lernen.

2. Zweitens können sie schon vor dem Lernprozeß alle Trainingsbeispiele fordern *(batch-Lerner)* oder sie akzeptieren ein Beispiel nach dem anderen *(inkrementelles Lernen)*.

3. Drittens kann sich der Lerner während des Lernprozesses auf ein Orakel verlassen, um die Gültigkeit von Verallgemeinerungen zu verifizieren oder um Beispiele, die vom Lerner erzeugt wurden, zu klassifizieren. In diesem Fall nennt man den Lerner *interaktiv*, ansonsten *nicht-interaktiv*.

4. Viertens kann ein Lerner neue Begriffe (Prädikate) erfinden. Das würde sein Vokabular erweitern, was eventuell die Aufgabe erleichtert.

5. Schließlich kann ein Lerner versuchen, ein Konzept ohne Vorgabe zu erlernen, oder er kann eine anfängliche Hypothese (Theorie) akzeptieren, die dann während des Lernprozesses überprüft wird. Letztere Systeme nennt man auch *Theorie-Beweiser*.

Obwohl diese Merkmale eigentlich unabhängig voneinander sind, liegen existierende ILP-Systeme an zwei Enden des Spektrums. Am einen Ende befinden sich nicht-interaktive

batch-Lerner die einzelne Prädikate ohne Vorgabe lernen, während am andern Ende interaktive, inkrementelle Theorie-Beweiser stehen, die mehrere Prädikate lernen. Nach [49] nennt man den ersten Typ *empirische ILP-Systeme* und den zweiten Typ *interaktive ILP-Systeme*.

MIS [53] und CLINT [49] sind typische interaktive ILP-Systeme: Sie lernen Definitionen mehrerer Prädikate aus einer kleinen Menge von Beispielen und Fragen an den Benutzer. Auf der anderen Seite lernen empirische ILP-Systeme typischerweise die Definition eines einzigen Prädikats aus einer großen Menge von Beispielen. Vertreter dieser Klasse sind z.B. FOIL [7], GOLEM [39], und LINUS [32]. Derzeit scheinen empirische ILP-Systeme wohl geeigneter für praktische Anwendungen zu sein. Deshalb werden die im weiteren beschriebenen Anwendungen auf empirische ILP-Systeme ausgerichtet sein, und wir werden ein empirisches ILP-Problem benutzen, um die grundlegenden ILP-Techniken darzustellen. Dennoch werden die beschriebenen grundlegenden Techniken sowohl bei empirischen wie auch bei interaktiven ILP-Systemen eingesetzt.

5.2.4 Empirisches induktives logisches Programmieren

Die Aufgabe des empirischen Lernens eines einzelnen Prädikats in ILP kann man formal wie folgt formulieren:

Gegeben:

- Eine Menge von Trainingsbeispielen $\mathcal{E}$, bestehend aus wahren $\mathcal{E}^+$ und falschen $\mathcal{E}^-$ Fakten eines unbekannten Prädikats,
- eine Konzept-Beschreibungssprache $\mathcal{L}$, die die Definition des Prädikats p syntaktisch beschränkt,
- Hintergrundwissen $\mathcal{B}$, welches Prädikate q_i (andere als p) bereitstellt, die bei der Definition von p verwendet werden können

Finde:

- eine Definition $\mathcal{H}$ für p, ausgedrückt in $\mathcal{L}$, so daß $\mathcal{H}$ komplett,
  d.h., $\forall e \in \mathcal{E}^+ : \mathcal{B} \wedge \mathcal{H} \vdash e$ und konsistent, d.h., $\forall e \in \mathcal{E}^- : \mathcal{B} \wedge \mathcal{H} \nvdash e$ ist.

Gewöhnlich bezeichnet man die wahren Fakten $\mathcal{E}^+$ als *positive ($\oplus$) Beispiele*, die falschen Fakten $\mathcal{E}^-$ als *negative ($\ominus$) Beispiele* und die Definition von p als Definition der *Ziel*relation. Positive Beispiele sind Tupel von denen man weiß, daß sie zur Zielrelation gehören; von den negativen Beispielen weiß man, daß sie nicht zur Zielrelation gehören. Es wird eine Definition der Zielrelation gesucht, die alle Beispiele richtig charakterisiert (d.h. alle positiven und keine negativen Beispiele umfaßt). Wenn man von nicht-perfekten (noisy) Beispielen lernt, müssen die Kriterien der Komplettheit und Konsistenz gelockert werden, d.h. eine annähernde Charakterisierung der Beispiele wird als hinreichend angesehen.

Das Symbol $\vdash$ bezeichnet dabei die Ableitbarkeit unter der Annahme einer bestimmten Schlußregel wie z.B. der Resolution: $T \vdash e$ bedeutet, daß e mit Hilfe dieser Regel von T abgeleitet werden kann. In Datenbank-Sprache würde $T \vdash e$ bedeuten, daß die Anfrage e Erfolg hat, wenn man sie an die Datenbank T richtet.

Übertragen in die Sprache der Datenbankwelt ist die Aufgabe von empirischem ILP also, eine intensionale Definition (Sicht) der Zielrelation p mit Hilfe anderer Relationen aus einer gegebenen Datenbank zu konstruieren. Ein möglicher Nutzen der konstruierten Sicht ist es, die Fakten (Beispiele) von p zu ersetzen um dadurch Speicherplatz zu sparen.

Die Konzept-Beschreibungssprache $\mathcal{L}$ wird gewöhnlich auch *Hypothesen-Sprache* genannt. Sie ist typischerweise eine Untermenge der Sprache der Programm-Klauseln. Die Komplexität des Lernens wächst mit der Ausdruckskraft der Hypothesen-Sprache $\mathcal{L}$. Deshalb wird $\mathcal{L}$ oft durch bestimmte Einschränkungen begrenzt. Eine typische Einschränkung ist z.B. das Verbot von Rekursionen in einer Klausel.

5.3 Grundlegende ILP-Techniken

ILP-Techniken suchen nach Klauseln in der Hypothesen-Sprache, die die Trainingsbeispiele überdecken, d.h. konsistent mit diesen sind. Der Raum der Klauseln wird durch eine Allgemeinheitsrelation zwischen Klauseln gegliedert, der sogenannten θ-subsumption. Eine Klausel c ist allgemeiner als eine Klausel d, falls sich c an eine Untermenge von d anpassen läßt. So ist zum Beispiel die Klausel $Tochter(X,Y) \leftarrow Weiblich(X)$ allgemeiner als die Klausel $Tochter(ann,Y) \leftarrow Weiblich(ann), Elternteil(Y,ann)$. Die hier besprochenen grundlegenden ILP-Techniken wurden alle schon in empirischen ILP-Systemen eingesetzt und sind deshalb für KDD anwendbar. Einige der Techniken wurden auch in interaktiven ILP-Systemen angewandt, oder dort sogar zuerst eingeführt. Die unten beschriebenen Techniken sind: relative least general generalisation, inverse Resolution, Durchsuchen von Verfeinerungsgraphen, Regelbasierte Modelle, und die Umwandlung von ILP Problemen in aussagenlogische Form.

5.3.1 Relative least general generalisation

Plotkin führte 1969 den Begriff der least general generalisation (*lgg*) ein [47], der für ILP sehr wichtig ist, da er die Basis für eine vorsichtige Verallgemeinerung bildet. Vorsichtig bedeutet hier, daß wenn zwei Klauseln c_1 und c_2 wahr sind, daß dann wahrscheinlich auch ihre speziellste Verallgemeinerung $lgg(c_1,c_2)$ wahr ist. Die *lgg* zweier Klauseln wird berechnet, indem man die *lgg* von jedem Paar von Literalen sowohl in den Köpfen als auch in den Rümpfen berechnet. Die *lgg* zweier Literale wird berechnet, indem man sie vergleicht und die sich unterscheidenden Teile durch Variablen ersetzt. Wir werden hier nicht die Definition der *lgg* geben, sondern sie anhand eines Beispiels erläutern. Sei $c_1 = Tochter(mary,ann) \leftarrow Weiblich(mary), Elternteil(ann,mary)$ und $c_2 = Tochter(eve,tom) \leftarrow$ *Weiblich(eve), Elternteil(tom,eve)*. Dann ist $lgg(c_1,c_2) = Tochter(X,Y) \leftarrow$ *Weiblich(X), Elternteil(Y,X)* (X steht für *lgg(mary,eve)* und Y steht für *lgg(ann,tom)).*

Die *relative least general generalisation* zweier Klauseln c_1 und c_2 ist die lgg von c_1 und c_2, bedingt (*relative*) durch das Hintergrundwissen $\mathcal{B}$. Relative least general generalisation ist die im ILP-System GOLEM [39] verwendete Grundtechnik, wobei das Hintergrundwissen auf Fakten beschränkt ist. Falls K die Konjunktion all dieser Fakten bezeichnet, definiert man den *rlgg* zweier Grundatome A_1 und A_2 (positive Beispiele) wie folgt:

$$rlgg(A_1, A_2) = lgg((A_1 \leftarrow K), (A_2 \leftarrow K))$$

Betrachten wir die Berechnung der rlgg für unsere Beispielrelation. Gegeben sind die positiven Beispiele e_1 = *Tochter(mary,ann)* und e_2 = *Tochter(eve,tom)* sowie das Hintergrundwissen $\mathcal{B}$ aus Tabelle 5.2.1. Die rlgg wird dann wie folgt berechnet:

$$rlgg(e_1, e_2) = lgg((e_1 \leftarrow K), (e_2 \leftarrow K))$$

wobei K für die Konjunktion der Literale *Elternteil(ann,mary), Elternteil(ann,tom), Elternteil(tom,eve), Elternteil(tom,ian), Weiblich(ann), Weiblich(mary),* und *Weiblich(eve)* steht. Zur Vereinfachung verwenden wir im weiteren folgende Abkürzungen: *T-Tochter, E-Elternteil, W-Weiblich, a-ann, e-eve, m-mary, t-tom, i-ian.* Die Konjunktion der Fakten des Hintergrundwissens ist:

$$K = E(a,m),\ E(a,t),\ E(t,e),\ E(t,i),\ W(a),\ W(m),\ W(e).$$

Die Berechnung der rlgg ergibt folgende Klausel:

$$\begin{aligned} T(V_{m,e},V_{a,t}) \leftarrow \quad & E(a,m),\ E(a,t),\ E(t,e),\ E(t,i),\ W(a),\ W(m),\ W(e), \\ & E(a,V_{m,t}),\ E(V_{a,t},V_{m,e}),\ E(V_{a,t},V_{m,i}),\ E(V_{a,t},V_{t,e}), \\ & E(V_{a,t},V_{t,i}),\ E(t,V_{e,i}),\ W(V_{a,m}),\ W(V_{a,e}),\ W(V_{m,e}). \end{aligned}$$

$V_{x,y}$ steht hierbei für *rlgg(x,y)*, d.h. sowohl für x als auch für y. Im Allgemeinen kann eine *rlgg* einer Menge von Trainingsbeispielen unendlich viele Literale enthalten, oder zumindest exponentiell mit der Anzahl der Beispiele wachsen. GOLEM geht deshalb bei der Aufnahme neuer Variablen in den Rumpf der *rlgg* nach festen Regeln vor. Doch trotz dieser Beschränkung sind *rlgg*s gewöhnlich sehr lange Klauseln mit vielen irrelevanten Literalen.

Entfernen wir in unserem Beispiel die irrelevanten Literale, so erhalten wir die Klausel $T(V_{m,e},V_{a,t}) \leftarrow E(V_{a,t},V_{m,e}),\ W(V_{m,e})$, welche für *Tochter(X,Y)* ← *Weiblich(X), Elternteil(Y,X)* steht.

Das auf *rlgg* basierende ILP-System GOLEM wurde auf mehrere praktische Probleme angewandt, wie zum Beispiel die Vorhersage der sekundären Struktur von Proteinen [40]. Dieses Beispiel werden wir uns später auch noch einmal genauer ansehen.

5.3.2 Inverse Resolution

Die Grundidee bei der inversen Resolution ist, die *Resolution* als Ableitungsregel umzukehren.

Bei der prädikatenlogischen Resolution werden zunächst zwei Klauseln betrachtet. Durch eine geeignete *Substitution* wird versucht, in den beiden Klauseln zwei sich widersprechende Literale zu erzeugen. Dabei wird eine Variable durch einen Term ersetzt. Die *Resolvente* ist dann die Disjunktion/Vereinigung der beiden Klauseln ohne die beiden sich widersprechenden Literale.

Inverse Resolution, wie sie zum Beispiel in CIGOL [38] implementiert wurde, benutzt einen Verallgemeinerungsoperator, der auf inverser Substitution beruht. Bei einer gegebenen Klausel W ist die *inverse Substitution* θ^{-1} einer Substitution θ eine Funktion, die Terme in $\theta(W)$ durch Variablen ersetzt, so daß $\theta^{-1}(\theta(W)) = W$. Sei $c = Tochter(X,Y) \leftarrow Weiblich(X), Elternteil(Y,X)$ und die Substitution $\theta = \{X/mary, Y/ann\}$:

$$c' = \theta(c) = Tochter(mary, ann) \leftarrow Weiblich(mary), Elternteil(ann, mary)$$

Indem man darauf die inverse Substitution $\theta^{-1} = \{mary/X, ann/Y\}$ anwendet, erhält man die ursprüngliche Klausel c:

$$c = \theta^{-1}(c') = Tochter(X,Y) \leftarrow Weiblich(X), Elternteil(Y,X)$$

Im allgemeinen ist inverse Resolution jedoch wesentlich komplexer. Sie bezieht die *Orte* der Terme mit ein, um sicherzustellen, daß die Variablen in der ursprünglichen Klausel W auch richtig in $\theta^{-1}(\theta(W))$ wiederhergestellt werden.

Ein Beispiel soll dies verdeutlichen: Das Hintergrundwissen $\mathcal{B}$ enthalte die zwei Fakten (Klauseln) b_1 = *Weiblich(mary)* und b_2 = *Elternteil(ann,mary)*. Die Anfangs-Hypothese sei $\mathcal{H} = \emptyset$. Der Lerner bekommt nun das positive Beispiel e_1 = *Tochter(mary,ann)*. Damit läuft inverse Resolution dann wie folgt ab:

- Im ersten Schritt versucht die Inverse Resolution eine Klausel c_1 zu finden, die e_1 mit Hilfe von b_2 folgert, und die der laufenden Hypothese $\mathcal{H}$ anstatt von e_1 hinzugefügt werden kann. Mit Hilfe der inversen Substitution $\theta_2^{-1} = \{ann/Y\}$, wird die Klausel $c_1 = ires(b_2, e_1) = Tochter(mary, Y) \leftarrow$ *Elternteil(Y,mary)* hergeleitet. Klausel c_1 wird zur laufenden Hypothese, so daß $\{b_2\} \cup \mathcal{H} \models e_1$.

- Die Inverse Resolution nimmt dann b_1 = *Weiblich(mary)* und die laufende Hypothese $\mathcal{H} = \{c_1\} = \{Tochter(mary, Y) \leftarrow Elternteil(Y, mary)\}$. Durch die inverse Substitution $\theta_1^{-1} = \{mary/X\}$, erhält man die Klausel $c' = ires(b_1, c_1) = Tochter(X,Y) \leftarrow Weiblich(X), Elternteil(Y,X)$. In der laufenden Hypothese $\mathcal{H}$ kann nun die Klausel c_1 durch die allgemeinere Klausel c' ersetzt werden, welche mit Hilfe von $\mathcal{B}$ das Beispiel e_1 folgert. Die induzierte Hypothese ist also $\mathcal{H} = \{Tochter(X,Y) \leftarrow Weiblich(X), Elternteil(Y,X)\}$.

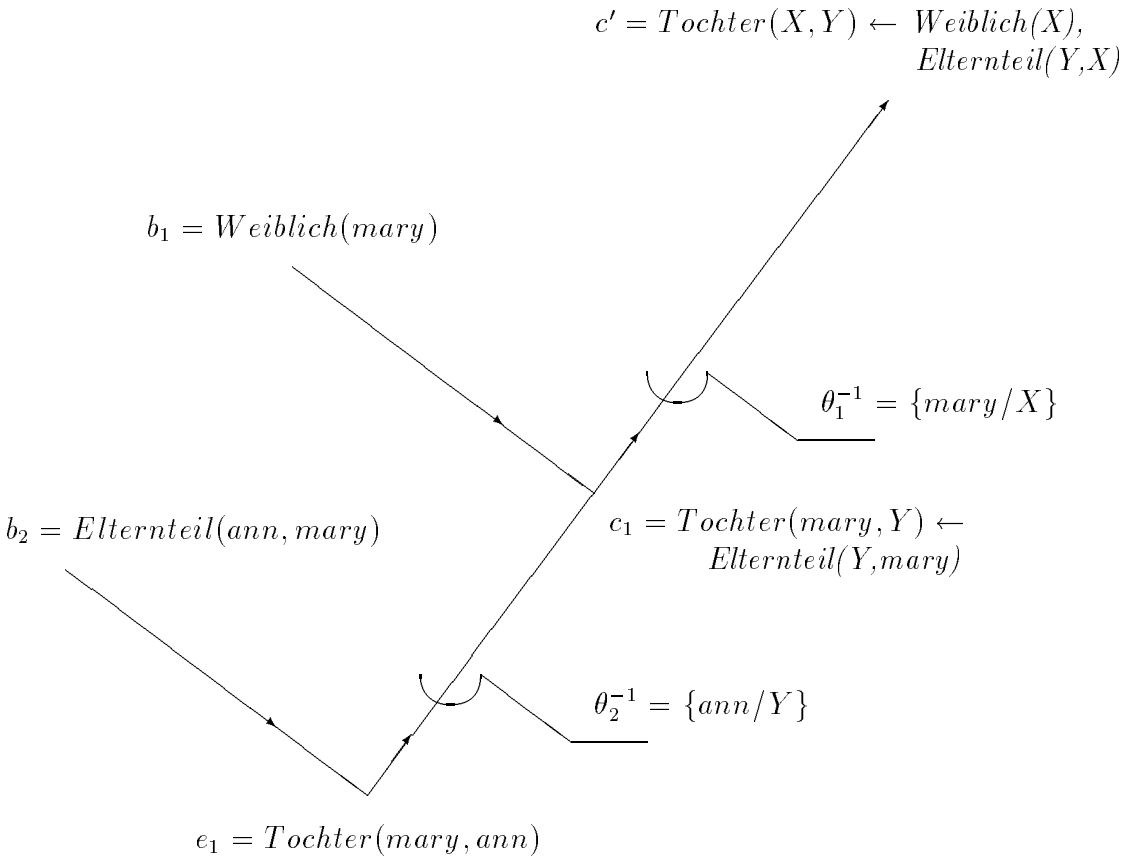


Abbildung 5.1: Ein inverser linearer Ableitungsbaum

Während die meisten auf inverser Resolution basierenden Systeme interaktiv sind, wurde erst in jüngster Zeit das empirische System PROGOL [41] entwickelt, das auch schon auf ein praktisches Problem der Chemie erfolgreich angewandt wurde.

5.3.3 Durchsuchen von Verfeinerungsgraphen

Während man in den vorangegangenen Abschnitten mit einer *bottom-up* Methode so lange verallgemeinert hat, bis alle positiven Beispiele überdeckt sind, geht man hier nun den umgekehrten Weg. Man startet bei der allgemeinsten Klausel und spezialisiert sie so lange, bis sie keine negativen Beispiele mehr überdeckt. Die grundlegende spezialisierende ILP-Technik ist dabei die *top-down Suche durch Verfeinerungsgraphen*. Während der Suche wird sichergestellt, daß mindestens ein positives Beispiel überdeckt ist. Diese ILP-Technik wurde das erste Mal im interaktiven ILP-System MIS [53] angewandt. Sie wird ebenfalls von dem empirischen ILP-System FOIL [7] und dessen Abkömmlingen mFOIL [12] und FOCL [42] benutzt.

Ein Verfeinerungsgraph kann als gerichteter azyklischer Graph definiert werden, wobei die *Knoten* Programmklauseln darstellen und die *Kanten* für eine der grundlegenden Verfeinerungsoperationen stehen: Ersetzung einer Variablen durch einen Term und Hinzufügen eines Literals zum Rumpf der Klausel. Ein Teil des Verfeinerungsgraphen für unser Beispiel mit den familiären Beziehungen ist in Bild 2 dargestellt. Die Suche nach einer Klausel startet mit der allgemeinsten Klausel $Tochter(X,Y) \leftarrow$, die zwei negative Beispiele überdeckt. Man schaut sich dann die Verfeinerungen dieser Klausel an, von denen $Tochter(X,Y) \leftarrow Weiblich(X)$ und $Tochter(X,Y) \leftarrow Elternteil(Y,X)$ jeweils nur ein negatives Beispiel überdecken. Diese beiden Klauseln haben eine gemeinsame Verfeinerung $Tochter(X,Y) \leftarrow Weiblich(X), Elternteil(Y,X)$, welche kein negatives Beispiel mehr überdeckt.

5.3.4 Regelbasierte Modelle

Ein regelbasiertes Modell wurde das erste Mal im System MOBAL [36] eingesetzt. Regelbasierte Modelle legen explizit die Form der Klauseln fest, die in der Hypothese vorkommen können. Diese Regeln werden als Schablonen für die Konstruktion von Hypothesen benutzt. Sie werden vom Benutzer bereitgestellt oder sie können auch aus zuvor gelernten Regeln abgeleitet werden.

Um die *Tochter*-Relation aus unserem Beispiel zu lernen, muß der Benutzer zunächst folgende Regel definieren:

$$P(X,Y) \leftarrow R(X), Q(Y,X)$$

In diesem Fall kann dann die korrekte Klausel $Tochter(X,Y) \leftarrow$ *Weiblich(X), Elternteil(Y,X)* induziert werden. Die obere Regel könnte auch für die Herleitung der Klausel $Sohn(X,Y) \leftarrow$*Männlich(X), Elternteil(Y,X)* verwendet werden. Man sieht, daß eine Regel beim Lösen mehrerer Lernprobleme nützlich sein kann. Der durch die Regeln festgelegte

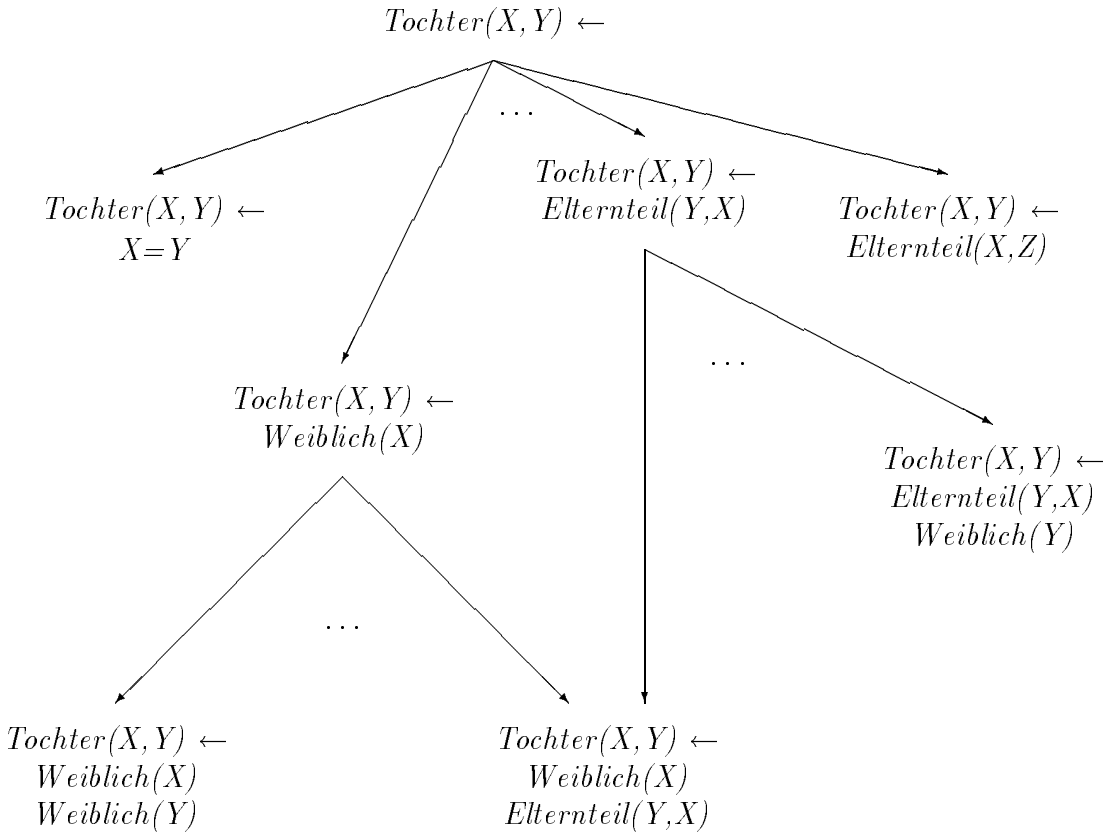


Abbildung 5.2: Teil des Verfeinerungsgraphen zum Familienproblem

Raum für Klauseln ist immer noch sehr groß. Für jede Regel werden alle möglichen Ersetzungen von Prädikatvariablen systematisch durchgegangen und die sich ergebenden Klauseln werden mit den Beispielen und dem Hintergrundwissen getestet. Eine hierarchische Ordnung der Regeln, basierend auf einer erweiterten θ-subsumption, erlaubt MOBAL, Teile des Hypothesenraums wegfallen zu lassen.

MOBAL wurde auf das praktische Problem angewandt, Zugangsregeln für Telekommunikationseinrichtungen in einem sicheren Umfeld zu lernen [36].

5.3.5 Umwandlung von ILP Problemen in aussagenlogische Form

Das ILP-System LINUS [32] basiert auf der folgenden Idee: Aus dem Hintergrundwissen können neue Attribute erzeugt werden, die für attributorientiertes Lernen geeignet sind. Ein ILP-Problem wird dabei von der relationalen in eine attributorientierte Form umgewandelt und dann von einem attributorientierten Lernverfahren gelöst. Dieser Ansatz ist jedoch nur bei einer eingeschränkten Klasse von ILP-Problemen anwendbar. Dabei wird die Hypothesensprache eingeschränkt auf funktionsfreie Programmklauseln, die *typisiert* (jede Variable ist mit einer vorbestimmten Menge von Werten verbunden), *in ihrer Form eingeschränkt* (alle Variablen im Rumpf einer Klausel kommen auch im Kopf vor) und *nicht-rekursiv* (das Prädikatsymbol aus dem Kopf kommt nicht im Rumpf vor) sind. Man nennt solche Klauseln auch funktionsfreie, in ihrer Form eingeschränkte DHDB-Klauseln (DHDB steht für Deduktive Hierarchische Datenbank).

Der LINUS-Algorithmus besteht aus den folgenden drei Schritten:

- Das zu lernende Problem wird von relationaler in attributorientierte Form umgewandelt.
- Das transformierte Problem wird von einem attributorientierten Verfahren gelöst.
- Die induzierte Hypothese wird in relationale Form rücktransformiert.

Wir veranschaulichen uns den Algorithmus am Beispiel unserer familiären Beziehungen. Aufgabe ist es, eine Definition für die Relation *Tochter(X,Y)* zu finden, und das hinsichtlich der Relationen des Hintergrundwissens *Weiblich, Männlich und Elternteil*. Alle Variablen sind vom Typ *Person*, der folgendermaßen definiert ist: *Person* = {*ann, eve, ian, mary, tom*}. Es gibt zwei positive und zwei negative Beispiele der Zielrelation (siehe Tabelle 5.2.1).

Der erste Schritt des Algorithmus sieht jetzt folgendermaßen aus: Zuerst werden alle möglichen Anwendungen der Hintergrund-Prädikate auf die Argumente der Zielrelation bestimmt. Jede dieser Möglichkeiten führt ein neues Attribut ein. Ein Attribut wird als *wahr* gewertet, falls es einem Tupel aus dem Hintergrundwissen entspricht, ansonsten ist es *falsch*. Dadurch ergibt sich in unserem Beispiel Tabelle 2.1.

Im zweiten Schritt berechnet ein attributorientiertes Lernprogramm aus den Werten in der Tabelle die nachfolgende if-then-Regel.

Klasse	Variablen		Werte							
	X	Y	W(X)	W(Y)	M(X)	M(Y)	E(X,X)	E(X,Y)	E(Y,X)	E(Y,Y)
$\oplus$	mary	ann	wahr	wahr	falsch	falsch	falsch	falsch	wahr	falsch
$\oplus$	eve	tom	wahr	falsch	falsch	wahr	falsch	falsch	wahr	falsch
$\ominus$	tom	ann	falsch	wahr	wahr	falsch	falsch	falsch	wahr	falsch
$\ominus$	eve	ann	wahr	wahr	falsch	falsch	falsch	falsch	falsch	falsch

Tabelle 5.2: Aussagenlogische Form des *Tochter*-Problems

$$Klasse = \oplus \qquad \textbf{\textit{if}} \qquad [Weiblich(X) = wahr] \wedge [Elternteil(Y, X) = wahr]$$

Im letzten Schritt werden die induzierten if-then-Regeln in DHDB-Klauseln umgewandelt. In unserem Beispiel erhalten wir die Klausel $Tochter(X, Y) \leftarrow$ *Weiblich(X), Elternteil(Y,X).*

Man beachte, daß in einem einzigen ILP-System auch mehrere Techniken zum Einsatz kommen können. So nutzt MOBAL [36] zum Beispiel ein regelbasiertes Modell in Verbindung mit Top-down-Suche. Bidirektionale Suche, welche Top-down- und Bottom-up-Suche miteinander kombiniert, kann ebenfalls in einem ILP-System angewandt werden.

5.4 Neue, für KDD einsetzbare Entwicklungen in ILP

Dieser Abschnitt gibt einen Überblick über drei Entwicklungen in ILP, die für KDD eingesetzt werden können: mehrfaches Prädikat-Lernen, induktives Data-Engineering, und Klauselermittlung.

5.4.1 Mehrfaches Prädikat-Lernen

Hier werden empirische Systeme vor die Aufgabe gestellt, die logischen Definitionen von m verschiedenen Prädikaten $p_1, \ldots, p_m$ zu lernen. Zur Verfügung stehen ihnen eine Menge von Trainingsbeispielen $\mathcal{E}$ dieser Prädikate und ein Hintergrundwissen $\mathcal{B}$, welches Prädikatdefinitionen von $q_1, \ldots, q_l$ enthält. Diese Aufgabe ist dem Lernen eines einzelnen Prädikats sehr ähnlich, bezieht jedoch zusätzliche Probleme mit ein, die auftreten, wenn man wechselseitig abhängige Prädikate lernt. Methoden zur Lösung dieser Probleme sind zum Beispiel:

- Betrachtung von Klauseln nur auf einem lokalen Level,
- Festlegung der Reihenfolge der zu lernenden Prädikate,
- zu starke Verallgemeinerung und

- wechselseitige Rekursion.

Das ILP-System MPL [51] wurde speziell dafür entworfen, mehrere Prädikate zu lernen. Es basiert auf Top-Down-Suche und betrachtet Klauseln sowohl lokal als auch global. Falls Inkonsistenzen entdeckt werden, springt das System zurück und schlägt einen anderen Weg ein (*Backtracking*).

5.4.2 Induktives Data-Engineering

Der Begriff Induktives Data-Engineering (IDE) wird in Verbindung mit der interaktiven Umstrukturierung von Datenbanksystemen mittels Induktion benutzt. Die Hauptidee dabei ist, mittels Induktion gültige Integritätsbedingungen in einer Datenbank zu bestimmen, mit deren Hilfe dann die Datenbank zerlegt (umstrukturiert) wird. Die in dem IDE-System INDEX [19] verwendeten Integritätsbedingungen enthalten funktionale und mehrwertige Abhängigkeiten.

Eine unstrukturierte Datenbasis enthält typischerweise redundante Information. INDEX zerlegt eine solche Relation in mehrere einfache Relationen, indem es Abhängigkeiten von Attributen nutzt. Es definiert die ursprüngliche Relation als intensionale Relation hinsichtlich der einfachen Relationen. Übertragen in die Sprache von ILP heißt das, INDEX erfindet neue Prädikate. Während des Umstrukturierens bezieht es den Benutzer interaktiv mit ein, zum Beispiel bei der Auswahl der Abhängigkeit nach der die Relation zerlegt werden soll.

5.4.3 Klauselermittlung

Das in Abschnitt 5.2 beschriebene Umfeld für ILP, welches interaktives und empirisches ILP enthält, wird als das normale ILP-Umfeld angesehen. Hier werden Regeln zur Beschreibung eines oder mehrerer Prädikate durch Induktion gefunden. Falls die induzierten Regeln ausreichend genau sind, können sie die Beispiele ersetzen. Man kann diese nämlich jetzt aus den Regeln ableiten. Ein komplett anderer Ansatz wird im sogenannten *nichtmonotonen* ILP-Umfeld verfolgt: Gegeben ist eine komplette Datenbasis und gesucht ist eine Menge von Bedingungen oder Zusammenhängen, die in der Datenbasis vorhanden sind. Da diese Zusammenhänge die Form von prädikatenlogischen Klauseln haben, wird dieser Ansatz auch als *Klauselermittlung* bezeichnet.

Um die Klauselermittlung im nicht-monotonen ILP-Umfeld zu erläutern, schauen wir uns wieder eine Datenbank mit Fakten zu familiären Beziehungen an. Dies seien *Mutter(X,Y)*, *Vater(X,Y)*, und *Elternteil(X,Y)*. Das ILP-System CLAUDIEN [50] fand unter anderem die folgenden Integritätsbedingungen:

$$
\begin{aligned}
Mutter(X,Y) \vee Vater(X,Y) &\leftarrow Elternteil(X,Y),\\
Elternteil(X,Y) &\leftarrow Vater(X,Y),\\
&\leftarrow Mutter(X,Y) \wedge Vater(X,Y), \text{ und}\\
&\leftarrow Elternteil(X,X).
\end{aligned}
$$

Die dritte und vierte Bedingung besagen, daß es unmöglich ist, daß X zugleich Vater und

auch Mutter von Y ist, und daß es für eine Person unmöglich ist zugleich sein eigener Elternteil zu sein. Der Unterschied zum normalen ILP-Umfeld besteht darin, daß wenn wir die Datenbank wegschmeissen, wir sie dann nicht mehr mit der hier gelernten Theorie rekonstruieren könnten.

Um die Bedingungen und Zusammenhänge zu finden, durchsucht CLAUDIEN einen Verfeierungsgraph nach vollen Klauseln, d.h. Klauseln, die jeden Eintrag in der Datenbank überdecken.

5.5 KDD-Anwendungen von ILP

In jüngster Zeit sind mehrere Anwendungen von ILP aufgetaucht. Sie variieren beträchtlich im Grad ihrer Erfahrenheit und Bedeutung im Hinblick auf das jeweilige Anwendungsgebiet. Die meisten davon sind eher Labordemonstrationen als praktisch nutzbare Anwendungen. Anwendungen von ILP auf Probleme der Wissensgewinnung aus Datenbasen waren zum Beispiel das Erlernen von Regeln für eine frühe Diagnose von rheumatischen Erkrankungen aus einer medizinischen Datenbank und die biologische Einstufung der Wasserqualität von Flüssen aus einer Umwelt-Datenbank.

ILP wurde auch auf verschiedene Aufgaben der Wissensgewinnung in biomolekularen Datenbanken angewandt, wo die gewonnenen Erkenntnisse schon fast praktische Bedeutung erlangten. Wir werden uns hierzu deshalb ein Beispiel näher ansehen: Vorhersage der sekundären Struktur von Proteinen. Der ILP-Ansatz erzeugte hier Regeln, die genausogut oder sogar besser arbeiten als konventionelle Ansätze (z.B. neuronale Netze), und die zusätzlich Einsicht in wichtige chemische Zusammenhänge vermitteln.

5.5.1 Vorhersage der sekundären Struktur von Proteinen

Wir sehen uns hier nun kurz das Problem an, Regeln für die Vorhersage der sekundären Struktur von Proteinen zu lernen, auf welches das ILP-System GOLEM angewandt wurde [40]. Zuerst schauen wir uns das Problem näher an, dann das vorhandene Hintergrundwissen, und schließlich die erzielten Ergebnisse.

Ein Protein ist im Grunde genommen eine Kette von Aminosäuren. Die dreidimensionale Gestalt von Proteinen anhand ihrer Aminosäuresequenz zu bestimmen wird in weiten Kreisen als eines der größten ungelösten Probleme in der Molekularbiologie angesehen. Es ist ebenfalls von großem Interesse für die pharmazeutische Industrie, da die Gestalt eines Proteins im allgemeinen dessen Funktion bestimmt.

Die Reihenfolge der Aminosäuren nennt man die *primäre Struktur* eines Proteins. Räumlich sind die Aminosäuren auf verschiedene Arten angeordnet (Spiralen, Windungen, flache Abschnitte, usw.). Die dreidimensionale Gestalt eines Proteins wird als dessen *sekundäre Struktur* bezeichnet. Beim Versuch, die Gestalt (*sekundäre Struktur*) vorherzusagen, ist es einfacher, nur ein bestimmtes Muster zu betrachten, anstatt die Vielzahl aller Möglichkeiten. Von GOLEM wurde das Muster der α-Helix (Spirale) betrachtet.

Die Zielrelation *alpha(Protein,Position)* besagt, daß die Aminosäure an Position *Position* im Protein *Protein* zu einer α-Helix gehört. Negative Beispiele sind alle Positionen von Aminosäuren in bestimmten Proteinen, die nicht zu einer α-Helix gehören.

Für die Lösung des Problems stand GOLEM folgende Informationen als Hintergrundwissen zur Verfügung.

- Die Relation *position(A,B,C)* besagt, daß im Protein A an Position B die Aminosäure C steht. Es gibt 20 mögliche Aminosäuren, welche alle durch einen Kleinbuchstaben charakterisiert werden. Das Faktum *position(1HMQ,111,g)* besagt demnach, daß die Aminosäure an Position 111 im Protein 1HMQ Glycin ist.
- Die arithmetische Relation *oct f(A,B,C,D,E,F,G,H,I)* erlaubt die Durchnummerierung der Aminokette. Sie muß explizit bereitgestellt werden, da GOLEM keine eingebaute Arithmetik hat. Die Relation *oct f(A,B,C,D,E,F,G,H,I)* besagt, daß die Positionen A bis I in einer Reihe vorkommen. Ein Faktum dieser Relation wäre zum Beispiel *oct f(19,20,21,22,23,24,25,26,27)*.
- Physikalische und Chemische Eigenschaften von Aminosäuren werden mit einstellige Prädikaten beschrieben. Zu diesen Eigenschaften gehören zum Beispiel Hydrophobheit (Grad der Wasserlöslichkeit), Ladung, Größe, Polarität, ob eine Aminosäure aromatisch oder aliphatisch ist, usw. Die Eigenschaften werden durch Konstanten repräsentiert. So steht die Konstante *polar0* für Polarität Null. Es werden auch Relationen zwischen den Konstanten zur Verfügung gestellt, wie zum Beispiel *kleiner(polar0,polar1)*.

Kommen wir nun zu einer kurzen Beschreibung des Versuchsaufbaus und der Ergebnisse. Es wurden sechzehn Proteine aus der Brookhaven-Datenbank [2] verwendet, zwölf als Trainingsbeispiele und vier zum Testen der Ergebnisse. Ohne auf den näheren Ablauf der Induktion einzugehen sei erwähnt, daß jede der induzierten Regeln bis zu zehn Beispiele falsch einordnen durfte. Die gefundenen Regeln überdeckten um 60 % der Testbeispiele.

Um die Überdeckungsrate dieser vorläufigen Regeln weiter zu verbessern, wurde der Lernprozeß wiederholt. Die durch die ersten Regeln (Level 0 Regeln) gefundenen Positionen wurden dem Hintergrundwissen hinzugefügt. GOLEM wurde dann noch einmal darauf angesetzt, um neue (Level 1) Regeln zu finden. Das war nötig, da die Level 0 Voraussagen stark gestreut waren, d.h. es wurden nur kurze α-Helix Ketten vorhergesagt. Die Level 1 Regeln filterten tatsächlich die gestreuten Vorhersagen und fügten kurze Ketten von α-Helix Vorhersagen zusammen. Der Lernprozeß wurde dann noch einmal wiederholt, und man erhielt noch Level 2 Regeln.

Angewandt auf die Trainingsbeispiele produzierte GOLEM einundzwanzig Level 0 Regeln, fünf gleichartige Level 1 Regeln und zwei gleichartige Level 2 Regeln. Die Tabelle zeigt eine Regel jedes Levels. Die hergeleiteten Regeln erreichten Genauigkeiten von 78 % bei den Trainingsbeispielen und sogar 81 % bei den Testbeispielen. Zum Vergleich: Das beste zuvor erreichte Ergebnis waren 76 %, erzielt mit Hilfe eines neuronalen Netzes [29]. Die von GOLEM erzeugten Regeln haben im Gegensatz zu den neuronalen Netzen aber auch den Vorteil, verständlicher zu sein.

Regeln zur Vorhersage einer α-Helix Sekundärstruktur

Level 0 Regel

alpha0(A,B) $\leftarrow$ *oct f(D,E,F,G,B,H,I,J,K),*
position(A,D,Q), not_p(Q), not_k(Q),
position(A,E,O), not_aromatic(O), small_or_polar(O),
position(A,F,R),
position(A,G,P), not_aromatic(P),
position(A,B,C), very_hydrophobic(C), not_aromatic(C),
position(A,H,M), large(M), not_aromatic(M),
position(A,I,L), hydrophobic(L),
position(A,K,N), large(N), ltv(N,R).

Level 1 Regel

alpha1(A,B) $\leftarrow$ *oct f(D,E,F,G,B,H,I,J,K),*
alpha0(A,F), alpha0(A,G).

Level 2 Regel

alpha2(A,B) $\leftarrow$ *oct f(C,D,E,F,B,G,H,I,J),*
alpha1(A,B), alpha1(A,G), alpha1(A,H).

Tabelle 5.3: Regeln zur Vorhersage einer α-Helix Sekundärstruktur

5.6 Schlußfolgerung

ILP befaßt sich mit der induktiven Herleitung prädikatenlogischer Regeln in Form von Klauseln oder logischen Programmen. Dieser Prozeß findet im Kontext von deduktiven Datenbanksystemen statt, d.h. einer Menge von intensionaler und extensionaler Relationen. ILP kann deshalb benutzt werden um Muster zu erkennen, die mehrere Relationen miteinbeziehen. Obwohl man beim Einsatz von ILP für KDD Effizienz- und Speicherprobleme erwartet, stellen neue ILP Systeme mächtige Werkzeuge zur Verfügung, um diesen Problemen zu begegnen. Auch den Umgang mit nicht-perfekten (noisy) Daten findet man immer mehr in ILP Systemen. In Ihren Anstrengungen, den praktischen Wert ihrer Systeme zu zeigen, sind sich die ILP Entwickler einig, KDD Anwendungen zu verwenden. In Zusammenhang mit den Arbeiten von KDD-Experten könnte hier neues und nützliches Wissen entdeckt werden.

Kapitel 6

Maschinelles Lernen mit graphischen Modellen

Sven Burk

Kurzfassung *Im Bereich maschinelles Lernen wurde in der Vergangenheit eine Vielzahl an Algorithmen und Wissensrepräsentationsmethoden entwickelt. Viele dieser Algorithmen basieren auf Modifikationen einiger Grundmechanismen. Aus Anwendersicht stellt sich die Frage nach Stärken und Schwächen verschiedener Verfahren. Könnte durch geeignete Kombination und Modifikation bestimmter Standardverfahren die Entwicklungszeit für Lernalgorithmen verkürzt werden und für unterschiedlichste Anwendungsbereiche „optimale" Ergebnisse erzielt werden? Buntine [4] schlägt Wahrscheinlichkeitsmodelle, die Wissen in Form eines Abhängigkeitsgraphen veranschaulichen, als ein Konzept zur Lösung dieser Fragen vor.*

6.1 Einleitung

Maschinelles Lernen ist seit ca. 30 Jahren eine Forschungsrichtung im Umfeld der KI-Forschung. Gemeinsames Ziel in diesem Bereich entwickelter Verfahren ist es, aus Beispielen allgemeinere Zusammenhänge zu generieren. Bei Beispielen aus dem medizinischen Bereich kann dieses Wissen, z.B. bei Diagnoseproblemen, entscheidungsunterstützend eingesetzt werden. Für Beispiele, die aus Datenbanken entnommen sind, hat sich in jüngster Zeit der Begriff Knowledge Discovering in Databases (KDD) eingeprägt. Da aus der historischen Entwicklung eine Vielzahl an Verfahren entstanden sind, gab es schon früh Versuche, diese zu klassifizieren. Graphische Modelle finden sich in einem Klassifikationsvorschlag in [52, S. 11] wieder, der nach der Art der Wissensrepräsentation unterscheidet. Andere Wissensrepräsentationsarten sind Entscheidungsbäume, Frames und Produktionsregeln.

Steuernagel [54, S. 30] weist darauf hin, daß „aus Sicht des KDD ... kein einzelnes Verfahren existiert, welches gleichzeitig auf unterschiedliche Arten und Typen von Beispieldaten, unterschiedliche Zeit- und sonstige Ressourcenbeschränkungen und unterschiedliche Zielvorgaben hin allgemeingültig „optimal" arbeitet." Der Wissensgewinnungsprozeß erfolgt

in der Regel in enger Zusammenarbeit des Knowledge Ingenieurs mit den Experten, wobei verschiedene Verfahren und Verfahrensabwandlungen zum Einsatz kommen. Um den Modellierungsprozeß zu verkürzen, ist es wichtig, über Techniken zu verfügen, die flexibel sind in der Art, Wissen zu codieren, darzustellen und zu gewinnen. Probabilistische graphische Modelle, wie Bayes-Netzwerke, Influenz-Diagramme oder Markov-Netzwerke, eignen sich nach Buntine [4] aus folgenden Gründen besonders gut als Modellierungstechnik:

- Graphische Modelle stellen die Variablen eines Problems und die Beziehungen zwischen den Variablen aufgrund des wahrscheinlichkeitstheoretischen Begriff der Abhängigkeit bzw. Unabhängigkeit dar. Sie sind intuitiv begreifbar.
- Ohne auf mathematische Details einzugehen, wird die Struktur des Problems klar. Schlüsselabhängigkeiten werden ausgedrückt, irrelevante Tatbestände ignoriert.
- Graphische Modelle bilden ein einheitliches Rahmenkonzept für überwachte und nichtüberwachte Lernverfahren, neuronale Netzwerke und Kombinationen verschiedener Wissensrepräsentationsformen, sogenannte Hybride.
- Eignung für Schlüsselaufgaben des Wissensgewinnungsprozesses wie:
  - Problemzerlegung
  - Entwurf von Lernalgorithmen
  - Identifikation von wertvollem Wissen
  - Generierung von Erklärungen

In der Seminararbeit findet eine Beschränkung auf Bayes-Netzwerke als ein einfaches Beispiel für graphische Modelle statt. Abschnitt 3.2 gibt eine Einführung in graphische Modelle. Das Grundberechnungsschema und Eigenschaften von Bayes-Netzwerken werden anhand elementarer Beispiele behandelt. Abschnitt 3.3 behandelt zwei Anwendungen graphischer Modelle: Problemzerlegung und Wissensverfeinerung. Abschnitt 3.4 gibt einen Überblick über verschiedene Lern- und Wissensgewinnungsverfahren unter Betonung des Aspekts der Strukturgleichheit. Abschnitt 3.5 stellt allgemeine Klassen von Lernalgorithmen vor und erklärt diese kurz. Abschnitt 3.6 enthält Schlußfolgerungen.

6.2 Einführung in graphische Modelle

6.2.1 Begriffsklärung

Whittaker [55, S. 12] definiert graphische Modelle als Wahrscheinlichkeitsmodelle für multivariate Zufallsvariablen, deren Abhängigkeitsstruktur durch einen bedingten Abhängigkeitsgraph charakterisiert ist. Ein Graph ist definiert durch eine endliche Anzahl an Knoten und Kanten. Die Knoten des Graphen stellen die für das Anwendungsmodell (*domain model*) notwendigen Variablen dar. Diese können aus Datenbeständen stammende bekannte Werte, unbekannte Werte, wie z.B. unbekannte Klassen von Objekten in Datenbanken, oder Parameter, wie die Standardabweichung einer Normalverteilung, sein.

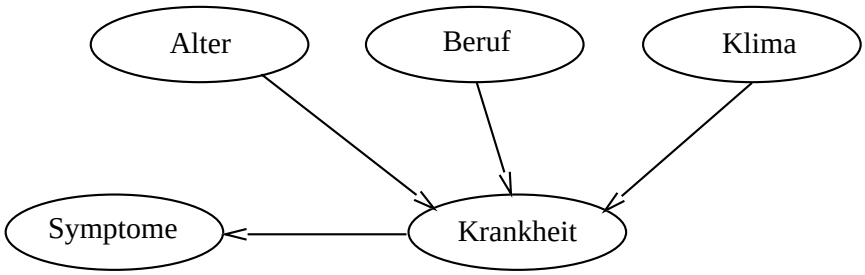


Abbildung 6.1: Ein vereinfachtes medizinisches Problem

Die Kanten stellen stellen wahrscheinlichkeitstheoretische Abhängigkeiten zwischen Variablen dar. Ein einfaches Beispiel für ein graphisches Modell in Form eines Bayes-Netzwerkes zeigt Abbildung 6.1. Ein Bayes-Netzwerk ist ein zyklenfreier gerichteter Graph, d.h. keine Pfeilfolge kann in Pfeilrichtung so durchlaufen werden, daß Anfangsknoten und Endknoten zusammenfallen. Unser Beispielgraph könnte so interpretiert werden: Alter, Beruf und Klima sind Faktoren, die eine Krankheit auslösen können. Ist eine Krankheit hervorgerufen worden, so verursacht sie bestimmte Symptome.

6.2.2 Grundberechnungsschema und Aussagekraft von Bayes-Netzwerken

Die Berechnung von Bayes-Netzwerken beruht auf vier wahrscheinlichkeitstheoretischen Definitionen. Oft ist bei einem Zufallsvorgang A von Interesse, ob die Wahrscheinlichkeit des Eintretens von A durch ein anderes Ereignis B beeinflußt wird. Dieser Sachverhalt wird durch bedingte Wahrscheinlichkeiten ausgedrückt, die wie folgt definiert sind:

$$P(A \mid B) = \frac{P(A, B)}{P(B)} \tag{6.1}$$

$P(B) > 0$ wird vorausgesetzt. Zwei Ereignisse A und B sind voneinander unabhängig, wenn gilt:

$$P(A, B) = P(A)\, P(B) \tag{6.2}$$

Mit dem Satz von der totalen Wahrscheinlichkeit kann die Wahrscheinlichkeit für ein Ereignis B unter anderen Ereignissen A_i mit $P(A_i) > 0$ berechnet werden:

$$P(B) = \sum_i P(B \mid A_i)\, P(A_i) \tag{6.3}$$

Falls $P(B) > 0$ ist, gilt ferner die Formel von Bayes, die besagt:

$$P(A_j \mid B) = \frac{P(B \mid A_j)\, P(A_j)}{\sum_i P(B \mid A_i)\, P(A_i)} \tag{6.4}$$

Im Folgenden soll anhand einiger Beispiele, die Vorgehensweise bei der Berechnung von Bayes-Netzwerken anhand von Fragestellungen aus dem medizinischen Bereich veranschaulicht werden.

a) Angenommen für ein medizinisches Diagnoseproblem ist die Wahrscheinlichkeit $P(H)$ für das Auftreten einer Krankheit H bekannt. Ebenso sind die Wahrscheinlichkeiten $P(e \mid H)$ und $P(e \mid \neg H)$ bekannt. $P(e \mid H)$ kann als Wahrscheinlichkeit für das Erscheinen eines Symptoms e unter der Bedingung, daß die Krankheit H diagnostiziert wurde, interpretiert werden. Wie kann umgekehrt die Wahrscheinlichkeit $P(H \mid e)$ prognostiziert werden ? Durch Division der Definition der bedingten Wahrscheinlichkeit durch die

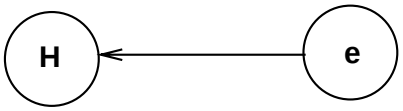


Abbildung 6.2: Diagnose bei einem Symptom

komplementäre Form $P(\neg H \mid e)$ erhält man:

$$\frac{P(H \mid e)}{P(\neg H \mid e)} = \frac{P(e \mid H)}{P(e \mid \neg H)} \frac{P(H)}{P(\neg H)} \tag{6.5}$$

Definiert man das Verhältnis der Vorgängerknoten als

$$O(H) = \frac{P(H)}{P(\neg H)} = \frac{P(H)}{1 \Leftrightarrow P(H)} \tag{6.6}$$

das Wahrscheinlichkeitsverhältnis als

$$L(e \mid H) = \frac{P(e \mid H)}{P(e \mid \neg H)} \tag{6.7}$$

und das Verhältnis der Nachfolgerknoten als

$$O(H \mid e) = \frac{P(H \mid e)}{P(\neg H \mid e)} \tag{6.8}$$

so kann man Gleichung 6.5 als Produkt schreiben:

$$O(H \mid e) = L(e \mid H) O(H) \tag{6.9}$$

Des weiteren berechnet sich $P(H \mid e)$ nach Umformungen aus:

$$P(H \mid e) = \frac{O(H \mid e)}{1 + O(H \mid e)} \tag{6.10}$$

Für die Aussagekraft des Ergebnisses ist es wichtig, daß beide Produktglieder in Gleichung 6.9 gut feststellbare relative Häufigkeiten sind. Liegt eine Patientendatenbank vor, so kann sowohl die Wahrscheinlichkeit für das Auftreten einer Krankheit als auch die Wahrscheinlichkeit für das Auftretens eines Symptoms unter der Voraussetzung, daß die Krankheit richtig diagnostiziert wurde, einfach bestimmt werden. Zum anderen ist $L(e \mid H)$ eine „lokale“ Größe, die nicht von anderem verstecktem Wissen beeinflußt wird, wenn man davon ausgeht, daß ein Symptom ein stabiles Kennzeichen einer Krankheit ist.

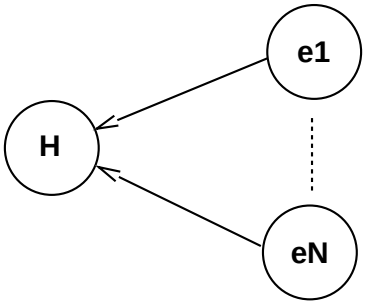


Abbildung 6.3: Diagnose bei mehreren Symptomen

b) Angenommen einer Krankheit liegen mehrere Symptome zugrunde.

Die Wahrscheinlichkeit für die Hypothese H berechnet sich jetzt als:

$$O\left(H \mid e_1, e_2 \ \ldots\ e_N\right) = O\left(H\right) L\left(e_1, e_2, \ \ldots\ e_N \mid H\right) \tag{6.11}$$

Die Berechnung der Gleichung vereinfacht sich unter der Annahme, daß die Symptome unabhängig voneinander sind. Dies führt auf folgenden Ausdruck:

$$O\left(H \mid e_1, \ \ldots, e_N\right) = O\left(H\right) \prod_{k=1}^{N} L\left(e_k \mid H\right) \tag{6.12}$$

Es genügt, die individuellen Charakteristika der Symptome zu kennen, um den Einfluß einer bestimmten Anzahl von Symptomen berechnen zu können.

c) Wie ist das Bayes-Netzwerk zu verändern, wenn angenommen zwei Symptome nicht direkt, sondern über einen medizinischen Test festgestellt werden, der mit einer gewissen Unsicherheit behaftet ist? Zudem soll das Testergebnis mehrere Zustände besitzen können. Abbildung 6.4 zeigt den veränderten Graph. Die Berechnung von $P\left(H \mid e_1, e_2\right)$ ändert sich,

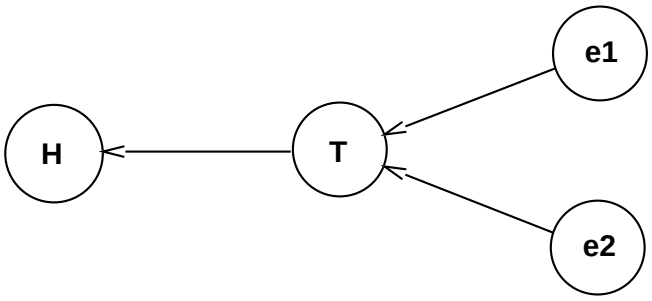


Abbildung 6.4: Graph bei Auftreten von Unsicherheit

da e_1 und e_2 nicht mehr direkt von H abhängen.

$$\begin{aligned} P\left(H \mid e_1, e_2\right) &= \alpha P\left(H\right) P\left(e_1, e_2 \mid H\right) \\ &= \alpha \sum_j P\left(T_j \mid H\right) P\left(e_1, e_2 \mid H, T_j\right) \end{aligned} \tag{6.13}$$

Unter der Annahme, daß e_1,e_2 und H bezüglich der Knotenvariablen T_j unabhängig sind, gilt:

$$P\left(e_1, e_2 \mid H, T_j\right) = P\left(e_1 \mid T_j\right) P\left(e_2 \mid T_j\right) \tag{6.14}$$

Gleichung 6.13 lautet jetzt

$$P(e_1, e_2 \mid H, T_j) = \alpha P(H) \sum_j P(T_j \mid H) P(e_1 \mid T_j) P(e_2 \mid T_j) \tag{6.15}$$

Der Berechnungsprozeß kann als dreistufiger Prozeß interpretiert werden. Zuerst werden auf der untersten Ebene die Vektoren $P(e_1 \mid T_j)$ und $P(e_2 \mid T_j)$ multipliziert. Durch Mulptiplikation dieses Vektors mit $P(T_j \mid H)$ erhält man einen Vektor $P(e \mid H)$. Am Ende ergibt sich die Formel für den Satz von Bayes. α ist dabei die normierende Größe, die aus der Forderung $\sum P(H \mid e_1, \ldots, e_N) = 1$ resultiert. Diese Möglichkeit, ein Problem top-down zu zerlegen und die Lösung rekursiv über einfache lineare Algebra zu berechnen, ist eine große Stärke von Bayes-Netzwerken. Die Eigenschaft einen Abhängigkeitsgraph in voneinander unabhängige Teilgraphen zu zerlegen, wird auch als Markov-Eigenschaft bezeichnet. Für eine tiefergehende Behandlung von Bayes-Techniken ist Pearl [43] sehr zu empfehlen.

In der Praxis ist die Berechnung von Bayes-Netzwerken keine triviale Aufgabe. Ist keine Baumstruktur vorhanden wie in Abbildung 6.5, so versucht man durch Clusteringtech-

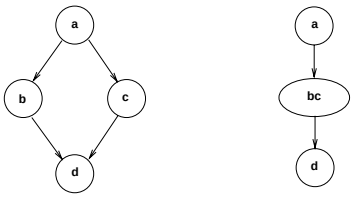


Abbildung 6.5:

niken, die Knoten b und c zusammenzuführen. Charniak [8] weist darauf hin, daß die exakte Berechnung von Bayes-Netzwerken ohne Baumstruktur ein $\mathcal{NP}$ -hartes Problem mit exponentieller Laufzeit ist.

6.3 Anwendungen graphischer Modelle

6.3.1 Problemzerlegung

Der Bereich maschinelles Lernen hat in einigen Richtungen starke Querbezüge zur klassischen Statistik. „Die überwachten Verfahren (*supervised learning*) besitzen im Gegensatz zu den nichtüberwachten Verfahren eine Zuordnung des Beispiels zu einer bestimmten Klasse .... Die Lernaufgabe besteht in dem Aufbau eines Klassifikators, der für Beispiele, deren Zugehörigkeit noch nicht bekannt ist, diese Zugehörigkeit vorhersagt [54, S. 32]." In der Statistik ist dieses Problem als Diskriminationsproblem bekannt.

Das Verfahren der Problemzerlegung soll an einem Beispiel aus [4] motiviert werden. Eine Nachrichtenagentur produziert jährlich Zehntausende von Artikeln zu ca. 90 verschiedenen Themengebieten. Die Artikel bestehen aus maximal 400 Wörtern, der verwendete Wortschatz beläuft sich auf 11 000 Wörter. Ohne Kenntnis der Überschrift des Artikels soll eine Einordnung des Artikels zu einem oder mehreren Themengebieten erfolgen. Der

große Eingaberaum und das Vorliegen eines mehrklassigen Entscheidungsproblems ist typisch für Probleme aus dem Bereich des überwachten Lernens. Derartige Probleme werden oft noch ohne Einsatz des Computers zerlegt. Man versucht eine Baumstruktur mit abnehmendem Abstraktionsniveau aufzubauen, die mit den 90 Themenklassen beginnt und in den Wörtern endet. Dabei wird die Struktur des Wortschatzes ausgenutzt, indem man z.B. Worte wie Silber, Platin, Gold zu Edelmetallen zusammenfaßt. Buntine [4] schlägt auf graphischen Modellen basierende Lernverfahren für eine automatisierte Problemzerlegung vor, räumt aber ein, daß in diesem Bereich noch einige Forschungsfragen offen sind. Prinzipiell eignen sich graphische Modelle, da top-down beginnend auf der obersten Ebene die Wahrscheinlichkeit $P(H)$ für ein Thema aufgrund von Daten aus der Vergangenheit feststellbar sind, ebenso die Wahrscheinlichkeiten für ein Unterthema e unter einem Thema $P(e \mid H)$ u.s.w. Liegt ein konkreter Artikel vor, so kann die Wahrscheinlichkeit für bestimmte Themen rekursiv berechnet werden. An dieser Stelle wird auch klar, wie wichtig sinnvolle Beispiele sind, aus denen gelernt werden kann.

6.3.2 Wissensverfeinerung

Wissensverfeinerung ist eine Anwendung für nichtüberwachte Lernverfahren. „Nichtüberwachte Verfahren (*unsupervised learning*) finden in den Beispielen noch nicht bekannte Klassen ...und ordnen die Beispiele diesen neu gefundenen Klassen zu. [54, S. 32]“. In der Statistik wird dieses Problem als Klassifikationsproblem bezeichnet.

Bekannte Software für Probleme aus diesem Bereich sind AutoClass und SNOB. In der Regel erfolgt die Wissensverfeinerung ausgehend von einem einfachen Modell, welches schrittweise, oft unter Modifikation der verwendeten Software, verfeinert wird. Im graphischen Modell werden die versteckten Klassen als „versteckte“ Knoten dargestellt. Ein Anwendungsbeispiel wäre die Feststellung von Klassen in einer Datenbank über Autounfälle. Graph 6.6 stellt eine mögliche Problemstellung dar. Es könnten z.B. Zusammenhänge

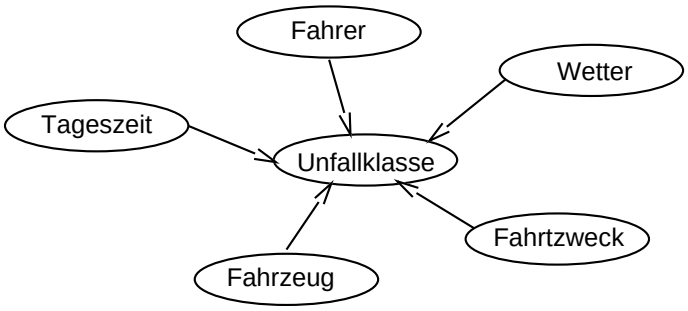


Abbildung 6.6: Einfaches nichtüberwachtes Modell für das Gebiet Autounfälle

auftauchen in der Form, daß ältere Fahrer mit schnellen Wagen bei ungünstigen Straßenverhältnissen eine Klasse bilden, weil sie die Technik überschätzen. Sind solche Zusammenhänge schon im voraus bekannt, so kann der Graph entsprechend abgeändert werden. Der Lernalgorithmus muß dann nur noch versuchen, die Reste des Modells zu verfeinern.

6.4 Modelle zur Wissensgewinnung

In diesem Abschnitt soll die Vielseitigkeit von graphischen Modellen als einheitliche „Sprache“ für unterschiedliche Verfahren aus der Statistik und des maschinellen Lernens aufgezeigt werden. Bei den die Beispiele illustrierenden Graphen wird angenommen, daß die Modellparameter, z.B. bezüglich einer Verteilungsannahme, bekannt sind. Bekannte Werte werden als schattierte Knoten dargestellt.

6.4.1 Lineare Regression

Die lineare Regression ist ein Standardverfahren aus der Statistik. Es beruht auf der Annahme, daß ein funktionaler Zusammenhang besteht zwischen einem quantitativen Merkmal Y mit Ausprägungsvariable y und quantitativen Merkmalen X_1, ...,X_n mit Ausprägungsvariablen $x_1, \ldots, x_n$. Der Erwartungswert $E(Y \mid X)$ berechnet sich aus

$$E(Y \mid X) = y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n \tag{6.16}$$

Das Modell ist linear, da die unbekannten Regressionskoeffizienten β_i linear eingehen. Ein Beispiel für ein nichtlineares Modell ist $y = \beta_0 + exp(\beta_1 x)$, da der Parameter β_1 nichtlinear in das Modell eingeht. Die Inputvariablen können auch nichtlineare orthogonale Funktionen wie Legendre-Polynome sein. Diese M Funktionen werden im folgenden Basisfunktionen genannt mit $basis_i = f(x_j)$ mit $i = 1, \ldots, M$, $j = 1, \ldots, n$.

Die Wahrscheinlichkeit $P(y \mid x_1, \ldots, x_n, \beta, s)$ bei bekannter Standardabweichung s und Erwartungswert $m = \sum_{i=1}^{M} \beta_i basis_i(x)$ unter der Annahme der Normalverteilung für y läßt sich berechnen als

$$P(y \mid x_1, \ldots, x_n, \beta, s) = \frac{1}{\sqrt{2\pi}s} e^{\frac{-(y-m)^2}{2s^2}} \tag{6.17}$$

oder als graphisches Modell wie in Abbildung 6.7 darstellen. Dieses Beispiel ist auch

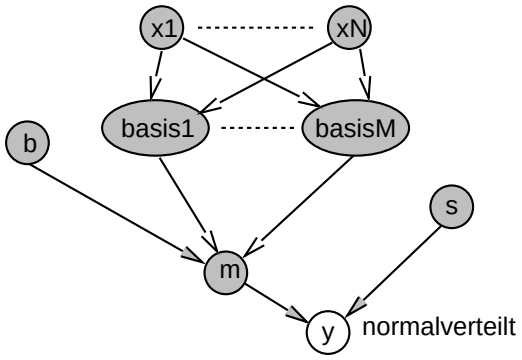


Abbildung 6.7: Lineare Regression auf Normalverteilungsannahme beruhend

ein Beispiel für ein verallgemeinertes lineares Modell. Bei diesen Modellen kommt außer der Linearität des Regressionsmodells noch die Annahme hinzu, daß die zugrundegelegte

Verteilung der Exponentialfamilie zugerechnet werden kann. Diese Familie von Verteilungen ist weitestgehend durch ihren Erwartungswert charakterisiert. Da die Normalverteilung bezüglich des Erwartungswertes symmetrisch ist, muß die Regressionsgleichung nicht durch eine Link-Funktion transformiert werden. Bei bezüglich des Erwartungswertes schiefen Verteilungen wird oft eine logistische oder logarithmische Transformation durchgeführt.

6.4.2 Gewichtete regelbasierte Systeme

Gewichtete regelbasierte Systeme sind Wissensrepräsentationsformen, die im KI-Bereich, bei neuronalen Netzen und in der Statistik verwendet werden. Ein Beispiel für eine diskrete Version eines solchen Modells ist Abbildung 6.8. Der Graph ist aufgrund seines linearen

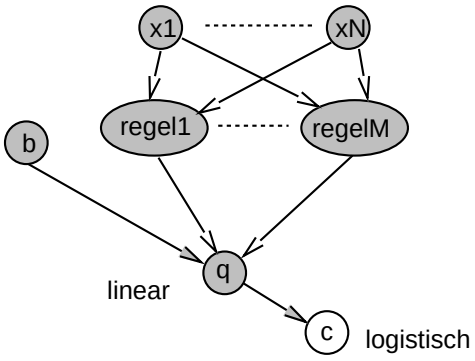


Abbildung 6.8: Ein gewichtetes regelbasiertes Netzwerk

Aufbaus ebenfalls ein Beispiel für ein verallgemeinertes lineares Modell. Die Basisfunktionen sind hier boolesche Funktionen, auch Regeln genannt, mit Werten 1, falls die Regel wahr ist und 0 falls nicht. Diese werden mit einem Gewichtungsfaktor b aufaddiert. Das Ziel dieses Verfahrens ist es, die Zugehörigkeit zu einer Klasse zu bestimmen.

Für den binären Klassifikationsfall berechnet sich die Wahrscheinlichkeit für die Klasse $c = 1$, falls mehrere Regeln den Wahrheitswert 1 besitzen durch folgende Transformation:

$$P\left(c = 1 \,|\, x_1, \ldots, x_n, b\right) = Logistic^{-1}\left(\sum_{i=1}^{M} regel_i \, b_i\right) \tag{6.18}$$

Die logistische Funktion eignet sich nach [27] „zur Repräsentation von Wachstumskurven. Sie hat gegenüber anderen Wachstumskurven den Vorteil, daß sie nicht mit der Zeit unendlich wird, sondern einer Asymptote, der sogenannten Sättigungsgrenze, zustrebt." Dies ist bei einer linearen Regression, bei der y eine Wahrscheinlichkeit darstellt, wichtig, da Wahrscheinlichkeiten nie größer als 1 sein dürfen. Allgemein hat der auf einer logistischen Verteilungsannahme beruhende Knoten c folgende Verteilungsfunktion:

$$P\left(c = 1 \,|\, u\right) = \frac{e^u}{1 + e^u} = 1 \Leftrightarrow Sigmoid\left(u\right) = Logistic^{-1}\left(u\right) \tag{6.19}$$

Die Funktion ist die Inverse der logistischen oder Logitfunktion, die bei verallgemeinerten linearen Modellen zur Symmetrisierung des Erwartungswertes benutzt wird. Sie ist ebenfalls mit der Sigmoidfunktion verknüpft, die bei neuronalen Netzen eine Rolle spielt.

6.4.3 Nichtüberwachte Lernverfahren

Nichtüberwachte Lernverfahren existieren in der Statistik, bei neuronalen Netzen und in der KI. Graphische Modelle eignen sich auch hier zur Darstellung der Abhängigkeitsstruktur. Noch versteckte Klassen werden in Form von „versteckten" Knoten dargestellt. Das Lernen von Bayes-Netzwerken aus Daten gehört prinzipiell auch in diesen Bereich wird oft aber exakter als Modellernen bezeichnet. Ein Beispiel für ein einfaches AutoClassmodell ist Abbildung 6.9. In diesem Modell sind Var_1,Var_2 und Var_3 boolesche Variablen, die

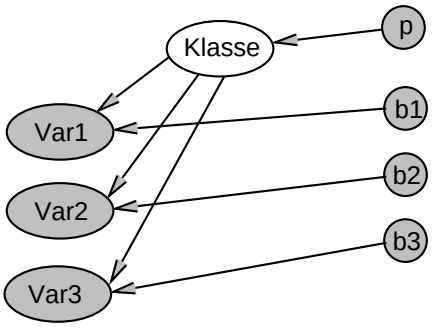


Abbildung 6.9: Explizite Parameter für ein einfaches AutoClassmodell

klassifiziert werden sollen. Gibt es angenommen zehn Klassen, so ist der Parameter P ein Wahrscheinlichkeitsvektor mit P_c als Wahrscheinlichkeit für die Zugehörigkeit zur Klasse c. Die Parameter b_1, b_2 und b_3 drücken aus, wie die Variablen innerhalb der Klassen verteilt sind. Ist Var_1 eine binäre Variable, dann wäre b_1 ein 10-Tupel von Wahrscheinlichkeitswerten. Ist die Klasse c bekannt, so wäre die Wahrscheinlichkeit dafür, daß Var_1 wahr ist, durch $b_{1,c}$ gegeben. Viele andere Modelle aus dem Bereich der nichtüberwachten Lernverfahren können ebenfalls durch bedingte Abhängigkeitsgraphen dargestellt werden.

6.5 Lernalgorithmen

Durch die Entwicklung nichtüberwachter Lernverfahren wurden Methoden entwickelt , um einfache diskrete sowie auf einer Normalverteilungsannahme basierende Bayes-Netzwerke automatisch zu generieren. Wie Buntine [4] betont, ist es trotz der Darstellbarkeit der linearen Regression und gewichteter regelbasierter Systeme als graphische Modelle mit diesen Verfahren nicht möglich, automatisch die Baumstruktur zu erzeugen. Für diese Zwecke existieren übergeordnete Klassen von Lernalgorithmen, die für unterschiedliche Problemstellungen kombiniert werden können. Abbildung 6.10 zeigt vier allgemeine Kategorien von Lernalgorithmen. Die einfachste Kategorie von Lernalgorithmen besitzt eine exakte, geschlossene Lösung für das Lernproblem, keine weitere numerische Optimierung ist notwendig. Die zugrundegelegte Verteilungsannahme gehört zur Exponentialfamilie wie z.B. die Normalverteilung, die Poissonverteilung und die multinomiale Verteilung. Die Wahrscheinlichkeit für die Daten bei bekannten Parametern ist $P(X \mid b)$.

Zweite weitere Kategorien von Lernmodellen basieren auf der Exponentialfamilie. Das partielle Exponentialmodell in Abbildung 6.10 b) gehört dazu. Der eine Teil des Modells der auf dem Exponentialmodell beruht kann in geschlossener Form bestimmt, der andere Teil

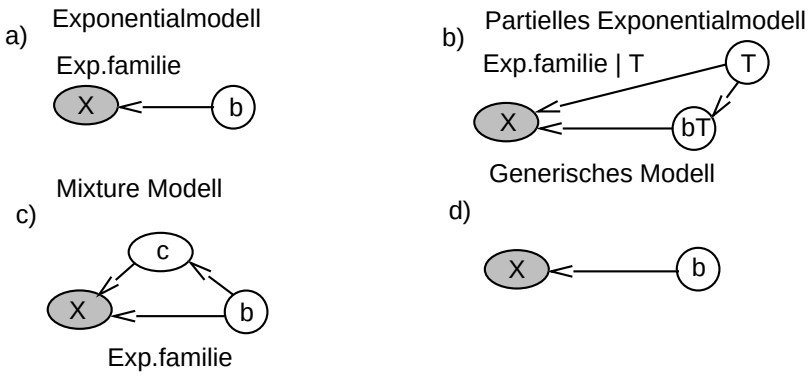


Abbildung 6.10: Vier Kategorien von Modellen

muß approximativ behandelt werden. Zu dieser Kategorie gehören Entscheidungsbäume und Bayes-Netzwerke, die auf einer Multinomial- oder Normalverteilung beruhen und bei denen die Struktur des Baumes bzw. Netzwerks nicht bekannt ist. Dies ist bei der linearen Regression der Fall, da Teilmengen von das Problem erklärenden Variablen herausgefunden werden sollen. Ist diese Struktur T bekannt, so gehört das Wahrscheinlichkeitsmodell $P(X \mid b_T, T)$ zur Exponentialfamilie. Das Mixture Modell beinhaltet 'versteckte' Knoten für noch unbekannte Klassen c. Sind diese Klassen bekannt, so gehört dieses Modell ebenfalls zur Exponentialfamilie. Ein Beispiel für diese Kategorie war die Klassifizierung von Autounfällen.

Generische Modelle bilden die vierte Kategorie von Lernalgorithmen. Sie sind eine Überbau für die drei anderen Kategorien, da sie auf keiner Verteilungsannahme bezüglich X beruhen. Daher können Lernalgorithmen aus diesem Bereich auch für die anderen Kategorien verwendet werden. Gewichtete regelbasierte Systeme werden dieser Kategorie zugerechnet.

6.6 Schlußfolgerungen

Graphische Modelle wurden in dieser Seminararbeit vor allem zur Veranschaulichung der beschriebenen Modelle benutzt. Dadurch wurde eine strukturelle Betrachtung der zugrundegelegten Wahrscheinlichkeitsmodelle ermöglicht und herausgestellt, daß graphische Modelle eine gemeinsame „Sprache“ für viele Verfahren bilden. Graphische Modelle eignen sich für viele Bereiche in denen Inferenzprobleme eine zentrale Rolle spielen. Diagnose, Informationsbeschaffung, Generierung von Erklärungen, Problemzerlegung und Wissensverfeinerung sind nur einige Beispiele für Bereiche, in denen Lernalgorithmen Verwendung finden. Durch graphische Modelle ist der direkte Einsatz von Wahrscheinlichkeitstheorie über Inferenzalgorithmen wie Maximum-Likelihood möglich. Andere Algorithmen wurden in Abschnitt 6.4 vorgestellt.

Als stärkstes Argument führt Buntine[4] die Möglichkeit der Verknüpfung unterschiedlichster Verfahren durch das gleiche Rahmenkonzept und die Umsetzung des Ergebnisses in Softwarecode an. Ihm schwebt dabei eine Art Softwaregenerator vor, der die Fähigkeit besitzt, für spezifische Problemstellungen maßgeschneiderte Lernalgorithmen zu schaffen, indem er unterschiedlichste Kombinationen von Lernalgorithmen verbindet. Von der

Vorgehensweise würden zuerst Methoden aus der Statistik und der Entscheidungstheorie benutzt, um das Problem in eine berechenbare Programmierungsvorschrift zu zerlegen. Bestehende Lücken werden dann durch den Einsatz von Optimierungs- und Suchmethoden aufgefüllt. Buntine weist darauf hin, daß ein Softwaretool von Gilks [24] existiert, welches für einen speziellen Bereich diese Vorgehensweise benutzt.

Im Rahmen des Forschungsprojekts CAKE Tool wurde am Institut für Werkzeugmaschinen und Betriebstechnik der Universität Karlsruhe ebenfalls Entwicklungen in diese Richtung vorangetrieben mit dem Ziel Entscheidungen in der Produktionstechnik zu unterstützen. In der Dissertation von Steuernagel [54] wird auch das Problem von Stärken und Schwächen verschiedener Lernverfahren erörtert im Unterschied zum Beitrag von Buntine. Dort wird nur die prinzipielle Eignung und Kombinierbarkeit von wahrscheinlichkeitstheoretischen Modellen für verschiedene Problemstellungen unter dem Konzept der graphischen Modelle diskutiert, ohne auf Bewertungsfragen von Lernverfahren bezüglich bestimmter Zielkriterien einzugehen. Steuernagel ging in seiner Arbeit nach der Analyse unterschiedlicher Lernverfahren auf Vor- und Nachteile den Weg über den Aufbau einer möglichst redundanzfreien, objektorientierten Bibliothek von induktiven Lernverfahren. Großer Wert wurde dabei auf eine benutzerfreundliche Handhabung gelegt. Nach dem Einsatz verschiedener Lernverfahren ist es mit CAKE Tool ebenfalls möglich die Ergebnisse in isoliert einsetzbaren Softwarecode umzuwandeln.

Kapitel 7

Integration heterogener Schemata

Ingo Redeke

Kurzfassung *Durch attributorientierte Lernverfahren zur Erzeugung von Klassifikations- und Charakterisierungsregeln sowie deren Anwendung werden bei der dargestellten Methode zur Integration heterogener Datenbankschemata bestimmte semantische Verwandtschaften zwischen Attributen ermittelt.*

Die Regeln, die für zwei oder mehr Attribute aus verschiedenen Datebanken erzeugt werden, liefern dabei mit Hilfe aufsteigender Suchbaumtechnik die Vergleichsmöglichkeiten auf höhren Ebenen. Um den Integrationsprozeß voranzutreiben, werden Beziehungen in und zwischen ganzen Attributsmengen herangezogen.

7.1 Einleitung

Im Zuge einer Globalisierung und Dezentralisierung von Datenbanken ist ein Verfahren gefordert, das den Datenaustausch zwischen ihren häufig semantisch heterogenen Komponenten übernimmt. Data-Mining Verfahren dienen hier dem Ermitteln von Wissen, wobei der im weiteren vorgestellte Algorithmus für eine Identifizierung und Auflösung semantischer Heterogenität sorgt. Das Hauptaugenmerk dabei richtet sich auf das Bestimmen äquivalenter Attribute in den Komponenten.

Die Schemaintegration wird deshalb durch ein Lernverfahren begründet, das die Werte, die die Attribute verschiedener Datenbanken annehmen können zueinander in Beziehung setzt und induktiv auf höhere Konzepte reduziert. Nicht in jedem Fall entstehen dadurch Äquivalenzaussagen, sondern nur hinreichende oder notwendige Bedingungen für ein bestimmtes Konzept.

7.2 Attributorientiertes Induktionsverfahren

Für das Verständnis des Data-Mining-Prozesses ist vorab eine Darstellung des zugrundeliegenden induktiven Lernverfahrens notwendig.

Ziel des Induktionsverfahrens ist es, Klassifikations- und Charakterisierungsregeln für die Attribute von Einträgen in relationalen Datenbanken aufzustellen. Wie bei induktiven Verfahren üblich, werden zur Lösung dieser Aufgabe eine Menge von Beispielen und eine konzeptuelle Voreinstellung (bias) herangezogen (Lernen aus Beispielen).

Die Menge der Beispiele sind hier die für die spezielle Lernaufgabe relevanten Daten (als Tupel) der Datenbank D, die bezüglich er Zielklasse positive als auch negative Daten enthält. Da die Trennung beider Anteile nicht explizit in einer Datenbank gegeben ist, müssen die Merkmale der positiven Daten für die Zielklasse und implizit dazu die der negativen Daten für die kontrastierende Klasse herangezogen werden.

In [6] wird dazu das folgende Beispiel angegeben, bei dem eine Regel für graduierte Studenten gefunden werden soll.

Name	Grad	Hauptfach	Geburtsort	$\emptyset$
Anderson	M.A.	Geschichte	Vancouver	3,5
Bach	junior	Mathematik	Calgary	3,7
Carey	junior	Literatur	Edmonton	2,6
Fraser	M.S.	Physik	Ottawa	3,9
Gupta	Ph.D.	Mathematik	Bombay	3,3
Hart	sophomore	Chemie	Richmond	2,7
Jackson	senior	Informatik	Victoria	3,5
Liu	Ph.D.	Biologie	Shanghai	3,4
Meyer	sophomore	Musik	Burnaby	2,9
Monk	Ph.D.	Informatik	Victoria	3,8
Wang	M.S.	Statistik	Nanjing	3,2
Wise	freshman	Literatur	Toronto	3,9

Tabelle 7.1: Beispiel eines Datesatzes

Die Zeilen in der Tabelle stellen jeweils die in der Datenbank enthaltenen Tupel dar.

Die Voreinstellung (bias) C der Konzepte wird durch eine Taxonomie geordnet. Als praktisch hat sich dabei die hierachische Struktur der IS_A - Bäume erwiesen, deren allgemeinstes Konzept „ANY“ und deren speziellstes Konzept der Eintrag eines bestimmten Attributwertes ist. C wird als Konzepthierarchie-Tabelle dargestellt, indem die gegebene Baumstruktur zur Mengen- und Untermengenvereinbarung genutzt wird. Sie kann mit Hilfe der in einer relationalen Datenbank herrschenden Verhältnisse automatisch oder halbautomatisch erzeugt werden (vgl. dazu den Vorschlag in [5] zur Erhöhung der Granularität bei Datenclustern).
Als Ergebnis entsteht aus dem Beispiel 7.1:

{ Informatik, Mathematik, Biologie, Statistik, Physik } $\subset$ Naturwissenschaft
{ Musik, Geschichte, Literatur } $\subset$ Geisteswissenschaft
{ freshman, sophomore, junior, senior } $\subset$ nichtgraduiert
{ M.S., M.A., Ph.D. } $\subset$ graduiert
{ Burnaby, Vancouver, Victoria, Richmond } $\subset$ British Columbia
{ Calgary, Edmonton } $\subset$ Alberta
{ Ottawa, Toronto } $\subset$ Ontario
{ Bombay } $\subset$ Indien
{ Shanghai, Nanjing } $\subset$ China
{ China, Indien } $\subset$ Ausland
{ British Columbia, Alberta, Ontario } $\subset$ Kanada
{ 2,0 - 2,9 } $\subset$ durchschnittlich
{ 3,0 - 3,4 } $\subset$ gut
{ 3,5 - 4,0 } $\subset$ sehr gut

Am Ende des Lernprozesses sollen die gesuchten Regeln in einer Sprache Λ darstellt werden. Im vorliegenden Fall ist Λ die Prädikatenlogik der ersten Stufe.

Die gegebenen Tupel werden in konjunktive Normalform transformiert und disjunktiv miteinander disjunktiv verknüpft. So ist, z.B., in Tabelle 7.1 der erste Eintrag:

$$\exists x((Name(x) = Anderson) \wedge (Grad(x) = M.A.) \wedge (Hauptfach(x) = Geschichte) \wedge$$

$$(Geburtsort(x) = Vancouver) \wedge (Durchschnitt(x) = 3,5))$$

Um ein sinnvolles Limit bei der aufsteigenden Suchbaumtechnik anzugeben, wird ein Schwellwert S eingeführt, bis zu dem die Generalisierung auf der disjunkten Tupelmenge ausgeführt wird. Er gibt die maximale Anzahl der Tupel in der abgeleiteten generalisierten Relation an. Da es kein algorithmisches Verfahren zu seiner Festlegung gibt, ist man gezwungen, einen Kompromiß zwischen einer groben Verallgemeinerung (S zu klein) und einem unbrauchbar hohen Anteil an Redundanzinformation (S zu groß) zu treffen.

7.2.1 Charakterisierungsregel

Der Algorithmus zum Finden der Charakterisierungsregel (im folgenden kurz **chr**) ersetzt den jeweils aktuellen Wert eines Attributes aller betrachteten Tupel gleichzeitig durch das nächsthöhere Konzept. Die daraus folgende Verallgemeinerung erzeugt u.U. im Datensatz gleiche Tupel, von denen alle bis auf eines eliminiert werden müssen. Dieser Vorgang wird solange für alle Attribute wiederholt, bis die Anzahl der Tupel den Schwellwert S erreicht.

Am Ende erhält man eine kleine Menge von Tupeln, die leicht in eine logische Formel überführt werden kann. Der Generalisierungsalgorithmus besteht aus folgenden vier

Schritten:
Schritt 1: Auswahl der Aufgabenrelevanten Daten
Durch Anwendung der Operationen *selektion*, *projektion* und *join* der relationalen Datenbank werden die relevanten Daten extrahiert.

Weiter kann das Attribut, auf das hin gelernt werden soll, aus der Tabelle entfernt werden, weil es nicht zur eigentlichen Lernaufgabe beiträgt. Im vorgestellten Beispiel betrifft das das Attribut „Grad".

Schritt 2: Attributorientierte Induktion
Sind für ein Attribut viele verschiedene Werte eingetragen und gibt es durch die Konzepthierarchie dafür übergeordnete Konzepte, wird darauf der eigentliche Generalisierungsschritt angewendet, wobei sich wiederholende Exemplare in eines vereinigt werden. Gibt es dagegen kein höheres Konzept für eine große Menge an Werten, sollte das ganze Attribut gelöscht werden.

Im Beispiel betrifft das das Attribut „Name". Tabelle 7.2 zeigt das Resultat dieses Verfahrens nach dem ersten Durchlauf.

Hauptfach	Geburtsort	$\emptyset$
Geisteswissenschaft	British Columbia	sehr gut
Naturwissenschaft	Ontario	sehr gut
Naturwissenschaft	British Columbia	sehr gut
Naturwissenschaft	Indien	gut
Naturwissenschaft	China	gut

Tabelle 7.2: Generalisierte Relation

Die Generalisierung wird solange wiederholt, bis die Anzahl der Tupel der resultierenden Relation nicht mehr größer als S ist. Für das Beispiel sei $S = 3$, d.h. das Attribut „Geburtsort", das für das Überschreiten von S verantwortlich ist, muß erneut generalisiert werden (Tabelle 7.3).

Man beachte, daß die letztendliche Anzahl verschiedener Werte pro Attribut geringer als S, aber damit nicht unbedingt die Anzahl der Tupel auch $\leq S$ sein muß! In dem Fall müssen weitere Attribute generalisiert werden, bis die Tupelanzahl die gewünschte Bedingung erfüllt.

Hauptfach	Geburtsort	$\emptyset$
Geisteswissenschaft	Kanada	sehr gut
Naturwissenschaft	Kanada	sehr gut
Naturwissenschaft	Ausland	gut

Tabelle 7.3: Weiter generalisierte Relation

Schritt 3: Simplifikation
Auf der generalisierten Relation sollte nach Möglichkeit eine Simplifikation angewandt werden, etwa, wenn die Werte aller Attribute bis auf eines gleich sind. Dann können die betreffenden Werte des verbliebenen Attributes durch das nächsthöhere Konzept ersetzt werden, ohne das die anderen Tupel berücksichtigt werden müssen.

Wie in Tabelle 7.4 gezeigt, können im Beispiel die Hauptfächer „Geisteswissenschaften“ und „Naturwissenschaften“ der ersten beiden Tupel zu „ANY“ vereinfacht werden, weil ihre Trennung für den Charakterisierungsprozess redundant ist.

Hauptfach	Geburtsort	$\emptyset$
ANY	Kanada	sehr gut
Naturwissenschaft	Ausland	gut

Tabelle 7.4: Simplifikation

Schritt 4: Überführung der generalisierten Relation in eine logische Formel
Jedes Tupel wird in konjunktive Normalform gebracht und alle zusammen in disjunktiver Normalform miteinander verknüpft. Aus dem Beispiel 7.4 ergibt sich dann umgangssprachlich, daß ein graduierter Student entweder Kanadier mit sehr gutem Punktedurchschnitt ist, oder ein ausländischer Student mit gutem Durchschnitt und naturwissenschaftlichem Hauptfach.

$$\forall(x) graduiert(x) \Longrightarrow (Geburtsort(x) \in Kanada \wedge Durchschnitt(x) \in sehrgut) \vee$$

$$(Hauptfach(x) \in Naturwissenschaft \wedge Geburtsort(x) \in Ausland \wedge$$

$$Durchschnitt(x) \in sehrgut)$$

7.2.2 Klassifikationsregel

Die Vorgehensweise bei der Erstellung der Klassifikationsregel (**klr**) ist ähnlich zu **chr**. Auch hier wird wieder das attribut-orientierte Lernverfahren angewendet. Um eine Zielklasse von der sie kontrastierenden Klasse zu unterscheiden, müssen die beide Klassen überlappenden Anteile eliminiert werden.

Zur Illustration sei das Beispiel in Tabelle 7.1 verwendet, diesmal jedoch, um eine **klr** aufzustellen, die graduierte von nicht-graduierten Studenten unterscheidet.

Schritt 1: Auswahl der Aufgabenrelevanten Daten
Wie in der **chr** werden auch hier alle negativen und positiven Daten aus der Datenbank extrahiert, die für die gestellte Aufgabe von Bedeutung sind. Anders als bei der **chr** werden jedoch nun die Tupel der nicht-graduierten mit in Betracht gezogen.

Schritt 2: Attributorientierte Induktion
Dieser Schritt verläuft äquivalent zu dem für **chr**. D.h., die Attribute „Name“ und „Grad“ entfallen und alle anderen werden generalisiert.

Weil nun jedoch eine **klr** erzeugt werden soll, müssen die Tupel, die mehrere Klassen überdecken gesondert behandelt werden. Zunächst werden diese überlappenden Tupel bei jeder Stufe des Hierarchiebaumes speziell markiert (vgl. Tabelle 7.5).

Lernkonzept	Hauptfach	Geburtsort	$\emptyset$	Marke
graduiert	Geisteswissenschaft	Kanada	sehr gut	*
	Naturwissenschaft	Kanada	sehr gut	*
	Naturwissenschaft	Ausland	gut	
nicht-graduiert	Naturwissenschaft	Kanada	sehr gut	*
	Geisteswissenschaft	Kanada	durchschnittlich	
	Naturwissenschaft	Kanada	durchschnittlich	
	Geisteswissenschaft	Kanada	sehr gut	*

Tabelle 7.5: Generalisierte Relation der **klr**

Das weitere Vorgehen (Generalisierung und Attributsentfernung in jedem Induktionsschritt) bezieht sich dann nur auf noch nicht markierte Tupel, also auch der Vergleich der Anzahl freier Tupel mit dem gegebenen Schwellwert. Die markierten Tupel müssen weiter mitgeführt und ihre Attributswerte im Zuge der Verallgemeinerungen ersetzt werden, um mögliche Markierungen auf einer höheren Ebene zu gewährleisten.

Schritt 3: Simplifikation
Die markierten Tupel werden entfernt und die verbliebenen wie bei **chr** vereinfacht.

Schritt 4: Transformation in logische Formel
Wie in **chr**. Die resultierenden Formeln sind im Gegensatz zu **chr** jedoch nur hinreichend, aber nicht notwendig:

$\forall(x) graduiert(x) \Longleftarrow Hauptfach(x) \in Naturwissenschaft \wedge Geburtsort(x) \in Ausland \wedge Durchschnitt(x) \in gut$

Und für nicht-graduierte:

$\forall(x) nichtgraduiert(x) \Longleftarrow Geburtsort(x) \in Kanada \wedge Durchschnitt(x) \in durchschnittlich$

7.3 Anwendung der Regeln

Im folgenden wird gezeigt, wie die oben dargestellten Verfahren zum Auffinden von Charakterisierungs-, wie Klassifikationsregeln über den Attributen hetergogener Datenbanken dienen. Anders als bei den üblichen Herangehensweisen, werden in diesem Fall ganze Attributsklassen verschiedener Datenbanken betrachtet.

Die in [10] dargestellte Arbeit stützt sich auf ein wissensbasiertes Schemaintegrations-Tool (*KBSIT*) als Teil eines intelligenten Verwaltungsprozesses für heterogene Datenbanken, sowie auf eine Implementierung der attributorientierten Induktion (*DBMiner*).

Vom KBSIT wird eine Mining-Anfrage an DBMiner geschickt, wenn das Wissen über die Beziehung zweier oder mehrerer Attribute gefordert wird. Der DBMiner erstellt daraufhin die aus den Datenbanken gewonnene **chr** und/oder **klr** und gibt sie an KBSIT zurück, das aus ihnen Vorschläge zu den interessierenden Beziehungen berechnet.

Formalisiert werden die KBSIT-Anfragen durch die Funktionen h und l. So ist $h(B = b|\{A_i\})$ die Anfrage nach der **chr** für Ausprägung b des Attributes B im Hinblick auf alle Attribute $\{A_i\}$. $h(B|\{A_i\})$ ist die gleiche Anfrage für alle Ausprägungen von B. Entsprechend **klr**, also $l(B = b|\{A_i\})$ und $l(B|\{A_i\})$ für **klr**.

7.3.1 Anwendbarkeit der Regeln

Chr als auch **klr** implizieren in der obenstehenden Anwendungsform selbst einen Satz Regeln für Art und Bestand von Attributsverwandschaften. Sie entstehen beim Vergleich der durch das Mining aufgefundenen Einzelregeln.

Charakteristische Regel

Gegeben sei das Attribut B_1 aus der Datenbank DB_1 und die Menge $\{CHR_{1j}\}$ mit $CHR_{1j} = h(B_1 = b_{1j}|\{A_i\})$, $j = 1 \ldots n$ der n erhaltenen Charakterisierungsregeln für B_1. Dabei ist n Die Anzahl der Ausprägungen von B_1.

Entsprechend sei B_2 aus DB_2 mit $\{CHR_{2k}\}$ und $CHR_{2k} = h(B_2 = b_{2k}|\{A_i\})$ für $k = 1 \ldots m$.

Wie aus dem im vorhergehenden Abschnitt dargestellten Beispiel zur Charakterisierung von graduierten Studenten hervorgeht, liefert **chr** stets eine notwendige Bedingung für die Zielklasse. $(\forall(x) : Zielklasse(x) \Longrightarrow Bedingung(x))$. D.h. jedes Tupel, das die gegebene Regel CHR_{rs} verletzt, kann nicht zur Zielklasse $\{B_r = b_{rs}\}$ gehören, während über alle die, die sie erfüllen keine Aussage gemacht werden kann. Durch den Vergleich von CHR_{1j} und CHR_{2k} erhält man eine Regel über Beziehung zwischen B_1 und B_2:

chr-1: Ist $CHR_{1j} \cap CHR_{2k} = \emptyset$, so folgt, daß b_{1j} und b_{2k} keine Darstellungen der semantisch kompatiblen Ausprägung sein, d.h. Äquivalenz, Ober- oder Unterklasse und zwar unabhängig davon, ob $b_{1j} = b_{2k}$ ist oder nicht.
Sind weiter $CHR_{1j} \cap CHR_{2k} = \emptyset$, $\forall(j,k)$, kann man voraussetzen, daß B_1 und B_2 semantisch inkompatibel sind.

Klassifikationsregel

Es seien wieder B_1 aus der Datenbank DB_1 und B_2 aus DB_2 mit der Menge der Klassifikationsregeln $\{CLR_{1j}\}$ und $CLR_{1j} = l(B_1 = b_{1j}|\{A_i\})$, sowie $\{CLR_{2k}\}$ mit $CLR_{2k} = l(B_1 = b_{2k}|\{A_i\})$ wobei wieder $j = 1 \ldots n$ und $k = 1 \ldots m$ gilt.

Im Gegensatz zur **chr** liefern die **klr** hinreichende Bedingungen der Tupel für die gegebene Zielklasse. $(\forall(x) : Zielklasse(x) \Longleftarrow Bedingung(x))$. Der Vergleich von CLR_{1j} mit CLR_{2k} impliziert Regeln zur Bestimmung, ob B_1 und B_2 ähnliche Attributsemantiken ergeben.

klr-1: Sei $CLR_{1j} = CLR_{2k}$ (jedes Tupel, das CLR_{1j} erfüllt, erfüllt auch CLR_{2k}). Dann gilt für

1. $b_{1j} = b_{2k} \rightarrow B_1$ ist äquivalent zu B_2
2. $b_{1j} \neq b_{2k} \rightarrow B_1$ und B_2 sind nur dann äquivalent, wenn es eine Abbildung zwischen b_{1j} und b_{2k} gibt, die bei unterschiedlichen Darstellungen semantische Äquivalenz erzeugt.

klr-2: Sei $CLR_{1j} < CLR_{2k}$ (entsprechend $CLR_{1j} > CLR_{2k}$), also jedes Tupel, das CLR_{1j} erfüllt, erfüllt auch CLR_{2k}, aber nicht umgekehrt. Dann gilt für

1. $b_{1j} = b_{2k} \rightarrow B_1$ ist eine Unterklasse von B_2
2. $b_{1j} \neq b_{2k} \rightarrow B_1$ ist nur dann eine Unterklasse von B_2, wenn es Abbildung M zwischen b_{1j} und b_{2k} gibt, derart, daß die Abbildung von b_{2k} äquivalent zu, oder eine Oberklasse der Abbildung von b_{1j} ist ($M(b_{1j}) \leq M(b_{2k})$).

klr-3: Sei $CLR_{1j} \cap CLR_{2k} \neq \emptyset$ (d.h. es gibt Tupel, die zwar Untermengen beider Regelmengen, aber nicht beide ganz erfüllen). Dann kann man annehmen, daß B_1 und B_2 „Ableger“ eines übergeordneten, aber noch unbekannten Konzeptes sind. Diese Annahme muß dann durch weitergehende Maßnahmen überprüft werden, bevor eine Integration stattfinden kann.

klr-4: Sei $CLR_{1j} \cap CLR_{2k} = \emptyset$ aber $b_{1j} = b_{2k}$, dann gibt es keine Übereinstimmung zwischen den Klassifikationsregeln bei gleichen Ausprägungen. In diesem Fall steht B_1 nur dann in einer Äquivalenz-, Oberklassen- oder Unterklassenbeziehung zu B_2, wenn es eine Abbildung von den Ausprägungen von B_1 und B_2 in eine kompatible Darstellung gibt.

7.3.2 Vorgang der Schemaintegration

Aufgabe des KBSIT ist die eigentliche Integration der heterogenen Schemata auf der Grundlage der korrespondierenden Attribute. Dazu muß es bereits bekannte Beziehungen zwischen den Attributen zweier oder mehrerer Datenbanken in einer eigenen Wissensbasis halten (Das sind $\{A_i\}$ in den obengenannten Anfragefunktionen, die Attribute also, die für beide Datebanken in gleicher Weise konzeptualisiert worden sind).

Eine Data-Mining-Anfrage an DBMiner wird notwendig, wenn KBSIT zwischen zwei heterogenen Attributen keine Beziehung bestimmen kann. Die dann vom DBMiner gelernten Regeln (**CHR** und **KLR**) werden auf eine allgemeine Aussage über die semantische Übereinstimmung der Attribute reduziert und das Ergebnis als Vorschlag für eine Beziehung angesehen, der in die bereits vorhandene Wissensbasis eingegliedert werden kann. Man sieht also, daß das eigentliche Ziel der Integration die Verschmelzung von Konzepthierarchien (Bildung neuer Konzepte) ist, während die Daten selbst unberührt bleiben.

Man kann nun Die Ergebnisregeln, die DBMiner liefert in einer Matrix zusammenfassen und durch Anwendung spezieller Reduktionsfunktionen ($H(B_k = b_{kj}|\{A_i\})$ und $L(B_k = b_{kj}|\{A_i\})$) durch einfache Ausdrücke ersetzen. Ebenso lassen sich von vornherein Anfragen, die eigentlich die Angabe der Potenzmenge der schon vorhandenen Attribute notwendig

macht, durch ein bottom-up Verfahren sukzessiv aus allen Teilmengen dieser Potenzmenge gewinnen und dadurch vereinfachen.

Das Beispiel nach dem folgenden Abschnitt soll den Vorgang beschreiben.

Reduzierung der Miningergebnisse

CHR:
Um aus allen Regeln aus CHR eine einzige verwertbare Aussage zu ziehen, werden in einer $n \times m$-Matrix $chrTBL(i,j)$ die Ergebnisse der Vergleiche von CHR_{1i} mit CHR_{2j} eingetragen.

Auf die Matrix wird anschließend eine Funktion H angewendet, die eine Verallgemeinerung der h-Regel des vorigen Abschnitts ist. Somit ist H definiert als $H(B_1/B_2|\{A_i\}) \in \{inc, eq\_inc, null, \emptyset\}$ mit

inc	$\hat{=}$	B_1 und B_2 sind inkompatibel
eq_inc	$\hat{=}$	B_1 und B_2 benutzen keine äquivalenten Darstellungen für Attributsausprägungen
$null$	$\hat{=}$	es gibt keine konsistente Reduktion
$\emptyset$	$\hat{=}$	$\{A_i\}$ enthält nicht genug Informationen für eine Mininganfrage nach B_1 und B_2

$$H(B_1/B_2|\{A_i\}) = \begin{cases} inc, & \forall(i,j) : chrTBL(i,j) = \emptyset \\ eq\_inc, & \forall(i) : chrTBL(i,i) = \emptyset \\ null, & \exists i,j,k,l : (i,j) \neq (k,l),\ chrTBL(i,j) = \emptyset,\ chrTBL(k,l) \neq \emptyset \\ \emptyset, & \text{(zuwenig Informationen)} \end{cases}$$

KLR:
Für die Klassifikationsregeln ergibt sich entsprechend eine Matrix $klrTBL(i,j)$ mit der Reduktionsfunktion $L(B_1/B_2|\{A_i\}) \in \{eq, sup, sub, sibl, null, \emptyset\}$ mit

eq	$\hat{=}$	B_1 und B_2 sind äquivalent
sup	$\hat{=}$	B_1 ist Oberklasse von B_2
sub	$\hat{=}$	B_1 ist Unterklasse von B_2
$sibl$	$\hat{=}$	B_1 ist ein Ableger von B_2
$null$	$\hat{=}$	es gibt keine konsistente Reduktion
$\emptyset$	$\hat{=}$	$\{A_i\}$ enthält nicht genug Informationen für eine Mininganfrage nach B_1 und B_2

$$L(B_1/B_2|\{A_i\}) = \begin{cases} eq, & \textbf{klr-1} \text{ ist in jeder Zeile und jeder Spalte mindestens einmal erfüllt} \\ sub, & \textbf{klr-1} \text{ oder } \textbf{klr-2} \text{ (mit „<“) ist in jeder Zeile und jeder Spalte mindestens einmal erfüllt} \\ sup, & \textbf{klr-1} \text{ oder } \textbf{klr-2} \text{ (mit „>“) ist in jeder Zeile und jeder Spalte mindestens einmal erfüllt} \\ sibl, & \textbf{klr-1}, \textbf{klr-2} \text{ oder } \textbf{klr-3} \text{ ist in jeder Zeile und jeder Spalte mindestens einmal erfüllt} \\ null, & \text{die Bedingungen für } sub \text{ und } sup \text{ gelten beide zugleich} \\ \emptyset, & \text{sonst} \end{cases}$$

Reduzierung der Anfragemenge

Im konkreten Fall habe KBSIT eine Menge bereits relationierter Attribute CA und soll Aussagen über den Grad der Beziehung zwischen B_1 aus DB_1 und B_2 aus DB_2 machen, für die CA nicht ausreicht. Die Mining-Anfrage wird also durch $L(B_1/B_2|CA)$, bzw. $H(B_1/B_2|CA)$ an DBMiner gestellt.

Um zu verhindern, daß es zu Fehlentscheidungen bei der Aufstellung der Beziehung kommt, wenn etwa bereits Untermengen von CA die B_1/B_2 im Grunde schon ausreichend abdecken, muß die Potenzmenge von CA herangezogen werden. Allerdings ist dann die Menge der Anfragen (alle Untermengen der Potenzmenge müssen betrachtet werden) im allgemeinen nicht praktikabel.

Abhilfe schafft dabei eine Reduzierung der Anfragemenge auf eine Annäherung an das ideale Szenario. Dazu werden eine Ordnung und eine Vergleichsfunktion auf den ermittelten Beziehungen definiert:

Es gilt $eq > sup > sibl$, $eq > sub > sibl$ und $sibl > null > \emptyset$ sowie $sup \neq sub$.

Für die Vergleichsfunktion auf den Beziehungen (r_i) gilt:

$$compare(r_1, r_2) = \begin{cases} positiv, & r_1 \geq r_2 \\ negativ, & r_1 < r_2 \end{cases}$$

Statt der ganzen Potenzmenge werden einzelne Attribute A_i aus CA sukzessiv in den Mining-Anfragen verwendet: $L(B_1/B_2|\{A_i\})$ [1], dann $L(B_1/B_2|\{A_i, A_j\})$ mit $i \neq j, j = 1 \ldots |CA|$. Das Verfahren wird in einem Suchzweig abgebrochen, wenn eine Anfrage $L(B_1/B_2|\{A_i, A_j\})$ sich nicht mehr positiv mit $L(B_1/B_2|\{A_i\})$ vergleichen läßt. Für alle positiven Schritte wird dann von $L(B_1/B_2|\{A_i, A_j\})$ nach $L(B_1/B_2|\{A_i, A_j, A_k\})$ weitergeschaltet, mit $i \neq j \neq k$. Von allen Resultaten wird am Ende dasjenige als Beziehung zwischen B_1 und B_2 vorgeschlagen, das die höchste positive Stelle einnimmt.

7.3.3 Beispiel für Integration

Es seien je ein Satz mit Tupeln für zwei Datenbanken einer kanadischen und einer deutschen Universität gegeben:

[1] entsprechend alle H-Anfragen

Name	Grad	Hauptfach	Geburtsort	∅
Anderson	M.A.	Geschichte	Vancouver	3,5
Bach	junior	Mathematik	Calgary	3,7
Carey	junior	Literatur	Edmonton	2,6
Fraser	M.S.	Physik	Ottawa	3,9
Gupta	Ph.D.	Mathematik	Bombay	3,3
Hart	sophomore	Chemie	Richmond	2,7
Jackson	senior	Informatik	Victoria	3,5
Liu	Ph.D.	Biologie	Shanghai	3,4
Meyer	sophomore	Musik	Burnaby	2,9
Monk	Ph.D.	Informatik	Victoria	3,8
Wang	M.S.	Statistik	Nanjing	3,2
Wise	freshman	Literatur	Toronto	3,9

Tabelle 7.6: Datesatz einer kanadischen Universität

Name	Rang	Hauptfach	Geburtsort	∅
Böhm	VD	Informatik	Hannover	1,3
Bundis	VD	Informatik	Rastatt	2,0
Ehlert	Prof.	Informatik	Ulm	1,5
Folkert	Dipl.	Informatik	Göttingen	2,7
Freihausen	Dipl.	Informatik	München	2,2
Gentz	VD	Informatik	Pforzheim	1,3
Luree	Prof.	Informatik	Nancy	1,7
Murnau	Dr.	Informatik	Warschau	1,2
Pitzel	Dipl.	Informatik	Stuttgart	3,7
Sauer	VD	Informatik	Salzburg	2,1
Tiedel	Dipl.	Informatik	Rostock	1,5
Gilbert	Dr.	Informatik	Marseille	2,5
Zertig	VD	Informatik	Wien	3,1

Tabelle 7.7: Datesatz einer deutschen Universität

Dazu müssen einige Vereinbarungen getroffen werden:

- Weil [Tab. 7.7] sich auf eine einzige Fakultät bezieht, muß ein der *Informatik* übergeordnetes Konzept, nämlich *Naturwissenschaft* der Konzepthierarchie hinzugefügt werden.
- Die Benotung in [Tab. 7.7] muß der in [Tab. 7.6] angepasst werden, um ein gemeinsames Konzept herzustellen. Hier bietet es sich natürlich an, für *Punktedurchschnitt* vorher einen Lauf von KBSIT zu starten, um eine Integration beider Konzepte vorzunehmen. Da es sich dabei jedoch um einige wenige diskrete Stufen handelt, ist eine a-priori Angleichung angebrachter.

- Die Konzepte für *Ausland* in den bias-Hierarchien muß um das jeweils andere Land (*Deutschland* und *Kanada*) erweitert werden, um Widersprüche zu vermeiden.

Die Konzepthierarchie für [Tab. 7.7] sieht wie folgt aus:

{ Informatik } $\subset$ Naturwissenschaft
{ VD, Dipl. } $\subset$ Studiengrad
{ Prof., Dr. } $\subset$ wissenschaftlicher Titel
{ Hannover, Göttingen } $\subset$ Niedersachsen
{ Rostock } $\subset$ Mecklenburg-Vorpommern
{ Rastatt, Ulm, Pforzheim, Stuttgart } $\subset$ Baden-Württemberg
{ Nancy, Marseille } $\subset$ Frankreich
{ Warschau } $\subset$ Polen
{ Salzburg, Wien } $\subset$ Österreich
{ Niedersachsen, Baden-Württemberg, Mecklenburg-Vorpommern } $\subset$ Deutschland
{ Frankreich, Österreich, Polen, Kanada } $\subset$ Ausland
{ 2,51 - 4,0 } $\subset$ durchschnittlich
{ 1,51 - 2,50 } $\subset$ gut
{ 1,0 - 1,50 } $\subset$ sehr gut

Ziel des Integrationsbeispiels ist es, zwischen den Attributen *Grad* und *Rang* eine Beziehung herzustellen. Die Anfragen an **DBMiner** lauten also $h(B_1 = graduiert|\{Geburtsort, Hauptfach, Punktedurchschnitt\})$ und entsprechendes für *nicht-graduiert* sowie an DB_2 für *Studiengrad* und *wissenschaftlicher Titel*. Der Schwellwert der **DBminer**-Suche sei wieder $S = 3$:

$CHR_{11} : \forall(x) : graduiert(x) \Longrightarrow (GO(x) = Kanada \wedge DU(x) = sehrgut) \bigvee$
$(HF(x) = Naturwissenschaft \wedge GO(x) = Ausland \wedge DU(x) = gut)$
$CHR_{12} : \forall(x) : nicht \Leftrightarrow graduiert(x) \Longrightarrow (GO(x) = Kanada)$

$CHR_{21} : \forall(x) : wissenschaftlicherTitel(x) \Longrightarrow (HF(x) = Informatik \wedge GO(x) = Baden \Leftrightarrow Württemberg \wedge DU(x) = sehrgut) \bigvee$
$(HF(x) = Informatik \wedge GO(x) = Frankreich \wedge DU(x) = gut) \bigvee$
$(HF(x) = Informatik \wedge GO(x) = Polen \wedge DU(x) = sehrgut)$
$CHR_{22} : \forall(x) : Studiengrad(x) \Longrightarrow (HF(x) = Informatik \wedge GO(x) = Deutschland) \bigvee$
$(HF(x) = Informatik \wedge GO(x) = Ausland \wedge DU(x) = gut) \bigvee$
$(HF(x) = Informatik \wedge GO(x) = Ausland \wedge DU(x) = durchschnittlich)$

Die Vergleiche der Regelmengen ergeben:

$CHR_{11} \cap CHR_{21} \neq \emptyset$ (wird durch *Gilbert* erfüllt)
$CHR_{11} \cap CHR_{22} \neq \emptyset$ (z.B. *Sauer*)
$CHR_{12} \cap CHR_{21} = \emptyset$
$CHR_{12} \cap CHR_{22} \neq \emptyset$ (z.B. *Bach*)

Mithin lautet die Ergebnismatix also:

$$\begin{pmatrix} \neq \emptyset & \neq \emptyset \\ \emptyset & \neq \emptyset \end{pmatrix}$$

Nach den Bearbeitungsregeln entsteht daraus das Ergebnis *NULL*, d.h. es gibt keine konsistente Reduktion der Vergleichergebnisse. Die Attribute lassen sich also nicht alle mit Sicherheit voneinander Trennen.

Für die Klassifikationsregeln ergibt sich:

$KLR_{11} : \forall(x) : graduiert(x) \Longleftarrow (HF(x) = Naturwissenschaft \wedge GO(x) = Ausland \wedge DU(x) = gut)$
$KLR_{12} : nicht \Leftrightarrow graduiert(x) \Longleftarrow (GO(x) = Kanada \wedge DU(x) = durchschnittlich)$

$KLR_{21} : \forall(x) : wissenschaftlicherTitel \Longleftarrow (GO(x) = Ausland \wedge DU(x) = gut)$
$KLR_{22} : \forall(x) : Studiengrad(x) \Longleftarrow (GO(x) = Deutschland \bigvee (GO(x) = Ausland \wedge DU(x) = durchschnittlich)$

Mit der Ergebnismatrix:

$$\begin{pmatrix} * & sibl. \\ \emptyset & < \end{pmatrix}$$

Wobei $*$ aus **klr-4** gewonnen wird, in der *Rang* und *Grad* nur dann äquivalent sind, wenn es eine Abbildung der Ausprägungen beider Attribute in eine kompatible Darstellung gibt. Der Ausdruck *sibl.* steht für die Anwendung von **klr-3**, und $<$ für **klr-2**. Die Regeln KLR_{12} und KLR_{21} haben keine gemeinsamen Tupel und sind daher vollständig disjunkt.

Als Endergebnis wird mit den Reduktionsregeln der Zusammenhang *SIBL*, also Ableger eines noch unbekannten Konzeptes, vorgeschlagen. Die beiden Ergebnisse *NULL* und *SIBL* können nun in die Wissensbasis durch **KSBIT** aufgenommen werden, oder es wird eine weitere Anfrage mit leicht veränderten Werten für den Schwellwert, oder erweiterten Datensätzen abgeschickt.

7.4 Ergebnisse

Es ist gezeigt worden, wie sich das Verfahren des attribut-orientierten Lernens von Klassifikations- und Charakterisierungsregeln für das Ermitteln von Verwandtschaftsbeziehungen zwischen Attributen heterogener Datenbanken genutzt werden kann. Dazu werden Mining-Anfragen auf den separaten Datenbanken gestartet, die von einer bereits vorhandenen, gemeinsamen Menge an Attributen abhängen. Alle Ergebnisse erhalten den Rang eines „Vorschlages“ zum Aufstellen der gesuchten Beziehung, d.h. sie kann akzeptiert und in eine Wissensbasis eingegliedert, oder einfach verworfen werden.

Die hier vorgestellte Kombination des attribut-orientierten Lernverfahrens nach [6] und der Integrationsmethode für heterogene Datenbankschemata nach [10] wird von den Autoren bewußt in zwei verschiedenen Modulen realisiert. Der DBMiner, der die gewünschten Regeln lernt, liefert seine Ergebnisse dem KBSIT, das den Integrationsprozeß übernimmt.

Trotz seiner zeitlichen Überlegenheit vor anderen Versionsraum-Verfahren ($O(N \log(N))$, bei N = Anzahl der Tupel in der Datenbank), wollen sich die Autoren Optionen für andere Mining-Methoden offenhalten.

Kapitel 8

KEFIR: Wissensgewinnung im Gesundheitswesen

Renate Ziegler

Kurzfassung *Health-KEFIR ist eine von GTE Laboratories entwickelte Anwendung des KDD-Systems Key Findings Reporter, kurz KEFIR. Ausgehend von einer Datenbank, die Daten zu Kosten, Krankenhausaufenthalten etc. enthält, sowie einer Tabelle mit Normwerten und einer Ansammlung von Hintergrundwissen über das Gebiet werden die in Zukunft möglichen Einsparungen im Gesundheitswesen berechnet und Vorschläge gemacht, wie diese Einsparungen erreicht werden können. Am Ende der Analyse wird ein Bericht, der aus Fließtext, Tabellen und Geschäftsdiagrammen besteht, im HTML-Format erstellt und mit NCSA-Mosaic ausgegeben.*

8.1 Einführung

Auch im Gesundheitswesen können KDD-Systeme interessante Ergebnisse liefern. Dabei ist vor allem der finanzielle Aspekt wichtig, also Möglichkeiten, Kosten einzusparen. Allerdings interessieren nicht so sehr die Einsparungen, die gegenwärtig und in der Vergangenheit möglich gewesen wären, sondern die zukünftig möglichen Einsparungen sowie die Maßnahmen, die dazu ergriffen werden müssen. Mit diesem Ziel wurde bei GTE Laboratories eine Anwendung des Key Findings Reporter (KEFIR) erstellt, Health-KEFIR. Dabei ist es notwendig, eine Vorausschau zu machen, um die zukünftigen Kosten herauszufinden, und es muß berücksichtigt werden, daß der Datenbestand sich ständig ändert. Diese Änderung des Datenbestandes ist sogar sehr wichtig, da die Ergebnisse, die Health-KEFIR liefert, um so besser sind, je größer der zu untersuchende Datenbestand ist, da dann eine genauere Vorhersage möglich ist. Im folgenden soll diese Anwendung genauer vorgestellt werden.

8.2 Überblick

KEFIR hat als Ausgangspunkt die Datenbank mit den Daten über Patienten, Behandlungen, Operationen, Dauer von Krankenhausaufenthalten, etc. Desweiteren benötigt KEFIR eine Ansammlung von Hintergrundwissen über das Gebiet, genannt *domain knowledge*. Um die möglichen Einsparungen herauszufinden, verwendet KEFIR außerdem Tabellen, die Normen zu den Kosten in den verschiedenen Bereichen enthalten. Aus diesen Tabellen lassen sich die Erwartungswerte für die Kosten herausfinden.

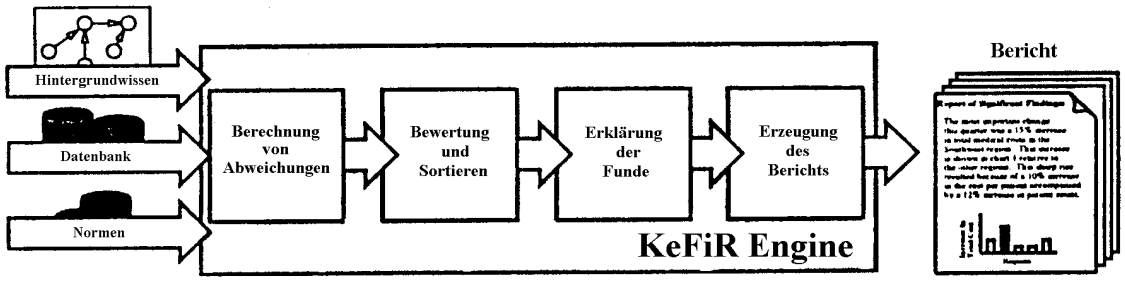


Abbildung 8.1: Überblick über KEFIR

Mit diesen Mitteln berechnet KEFIR zunächst die in Zukunft erwartete Abweichung der Kosten von der Norm. Dabei läßt sich der Suchraum nach mehreren Aspekten unterteilen. Anschließend bewertet KEFIR die gefundenen Abweichungen, d. h. das System berechnet die Auswirkung, d. h. den Impact, den die Mehrkosten in diesem Bereich auf die Gesamtkosten haben werden. Anhand eines Maßes für den Impact wird durch die Gewichtung verschiedener Faktoren die Interessantheit der Abweichung berechnet. Nach dieser Interessantheit werden die gefundenen Abweichungen sortiert und die mit der größten Interessantheit werden in den abschließenden Bericht aufgenommen. Bevor jedoch der Bericht erstellt wird, sucht KEFIR anhand der verwendeten Formeln und Suchraumeinschränkungen nach den Ursachen der gefundenen Abweichungen, so daß das System nicht nur die gefundenen Abweichungen ausgeben kann, sondern auch eine Erklärung, wodurch diese Abweichungen zustande kamen. Mit Hilfe des Hintergrundwissens wählt KEFIR noch einige vordefinierte Verbesserungsvorschläge aus und fügt Abweichungen, Erklärungen und Vorschläge in einer WWW-Seite als Bericht zusammen. Diese WWW-Seite wird dann mit NCSA-Mosaic angezeigt.

8.3 Der Suchraum

Der Suchraum, mit dem KEFIR arbeitet, besteht aus drei verschiedenen Komponenten: der Bevölkerung, dem medizinischen Bereich, und den zu untersuchenden Aspekten, die auch als Maße bezeichnet werden. Die Bevölkerung kann in verschiedene Gruppen unterteilt werden, z. B. nach Beruf, Alter, Wohnort, Bildungsstand. Für jede dieser Bevölkerungsgruppen gibt es eine weitere Unterteilung nach medizinischen Gesichtspunkten, z. B. ob Patienten ambulant behandelt wurden oder nicht, wie lange ihr Krankenhausaufenthalt dauerte, welche Krankheiten sie hatten, ob Operationen durchgeführt wurden.

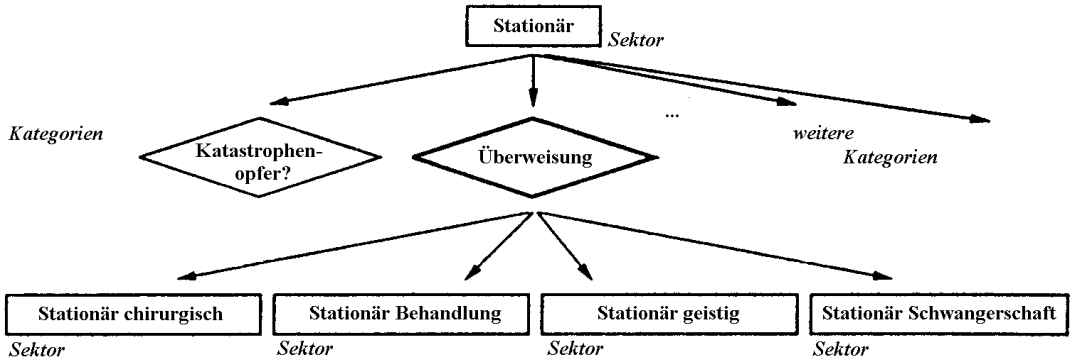


Abbildung 8.2: Aufteilung des Suchraums

Diese Unterteilung ist in Abb. 8.2 dargestellt. Der Durchschnitt der Bevölkerungsgruppe und des medizinischen Bereichs wird als *Sektor* bezeichnet. Die gröbste Einteilung der Bevölkerung wird mit P_0 bezeichnet, die gröbste Einteilung des medizinischen Bereichs mit A_0. Der allgemeinste Sektor ist demnach der Durchschnitt von P_0 und A_0 und wird S_0 genannt. Der beobachtete Sektor umfasst alle Behandlungen, die einen Krankenhausaufenthalt notwendig machten. In der ersten Stufe werden die Daten nach Katastrophenfällen, Einweisungen usw. eingeteilt. Jeder dieser Untersektoren wird weiter unterteilt, z. B. die Einweisungen in chirurgische Fälle, Geisteskrankheiten, Schwangerschaften etc. Dadurch wird eine sehr detaillierte Aufschlüsselung des medizinischen Bereichs erreicht, was für die Erklärung der Abweichungen später wichtig sein wird.

Die relevanten Maße sind wiederum für jedes Gebiet in der medizinischen Einteilung festgelegt. Allerdings hängen die Maße von der medizinischen Einteilung ab, da etwa die Dauer eines Krankenhausaufenthaltes für ambulante Patienten unwichtig ist bzw. nicht existiert (vgl. auch Kapitel 8.6).

8.4 Die Entdeckung und Bewertung von Abweichungen

Wie in der Einführung bereits angedeutet, ist es im Gesundheitswesen nicht sinnvoll, nur zu untersuchen, wie weit die tatsächlichen Kosten über den erwarteten Kosten liegen, da dies nur eine Aussage darüber ermöglicht, welche Einsparungen man verpaßt hat. Man muß also versuchen, eine Aussage über zukünftig mögliche Einsparungen zu machen. Daher benötigt man neben den Normen auch einen Erwartungswert für die zukünftigen tatsächlichen Kosten.

Der erste Schritt bei der Entdeckung von Abweichungen ist deshalb die Vorhersage der Kosten. Dazu führt KEFIR eine Interpolation mit den bisherigen Kosten durch. Diese Interpolation ist allerdings sehr einfach, da KEFIR lediglich eine Gerade durch die bisherigen Werte sucht. Eine Verbesserungsmöglichkeit hier wäre, die Interpolation exakter zu gestalten, also statt einer Geraden auch andere Kurven durch die Stützstellen zu suchen.

Der zweite Schritt ist die Berechnung von Impact und Interessantheit. Dabei geht man folgendermaßen vor: Man wählt ein Maß M_0 als Basis und sucht Funktionen f_i, mittels derer die anderen Maße M_i darauf zurückgeführt werden können, d. h. es muß gelten: $M_0 = f_i(M_i, D)$, wobei D die zu untersuchende Datenbank ist. Bei Health-KEFIR ist M_0 die Summe aller Kosten, die im Gesundheitswesen entstanden. Außerdem muß man die gewählte Bevölkerungsgruppe P_0 und den medizinischen Bereich A_0 berücksichtigen. Es gilt ferner, daß S_0 gleich der Vereinigung aller S_j ist $(1 \leq j \leq k)$ und die S_j paarweise disjunkt sind. Neben der zu untersuchenden Datenbank D_C muß man auch die Referenzdatenbank D_R berücksichtigen, die die alten Werte der Maße enthält. Der Impact bezüglich des Basis-Maßes wird folgendermaßen berechnet: Man berechnet die Abweichung, die dadurch entsteht, daß M_i im Sektor S_j in der Referenzdatenbank (also $M_i(S_j, D_R)$) auf den Wert in der aktuellen Datenbank, also $M_i(S_j, D_C)$ gesetzt wird. Der Unterschied zum Referenzwert $M_0(S_0, D_R)$ ist dann der Impact. Kurz gesagt handelt es sich dabei einfach um die Änderung des Maßes. Wenn z. B. in einem Sektor die Kosten 1992 zwei Millionen Dollar betrugen und 1993 drei Millionen Dollar, dann ist der Impact bezüglich des Basis-Maßes eine Million Dollar. Die Formel für den Impact ist also:

$$impact(M_i, S_j, D_C, D_R \| M_0, S_0) = f_i(M_i(s_j, D_C), D_R) \Leftrightarrow M_0(S_0, D_R)$$

In einem Spezialfall vereinfacht sich die Berechnung des Impacts jedoch erheblich: Wenn die Formeln, die die M_j auf M_0 zurückführen, nur Additionen und Multiplikationen enthalten (was bei Health-KEFIR meistens der Fall ist). Dann hat f_i die Form $M_0(S_0, D) = A(S, D) \times M_i(S, D) + B(S, D)$. $A(S, D)$ und $B(S, D)$ sind dabei von M_i unabhängig. Analog zur Herleitung der ersten Formel für die Berechnung des Impacts kommt man zu folgender Formel:

$$impact(M_i, S, D_C, D_R) = A(S, D_R) \times (M_i(S, D_C) \Leftrightarrow M_i(S, D_R))$$

Da bei der Subtraktion der additive Anteil $B(S, D)$ völlig wegfällt, genügt es bei Health-KEFIR, zur Berechnung eines Maßes bezüglich M_0 den multiplikativen Faktor zu berücksichtigen.

Die Interessantheit wird wie bei CoverStory, einem anderen KDD-System, als Produkt aus der prozentualen Abweichung, die entdeckt wurde, einem Gewichtsfaktor und der Quadratwurzel der Marktgröße berechnet, wobei hier unter Marktgröße die Ausgaben im beobachteten Bereich zu verstehen ist.

Es ist jedoch problematisch, sich ausschließlich auf Änderungen zu konzentrieren. Wenn in einem Bereich ein voraussichtlicher Anstieg der Kosten entdeckt wird, muss dies nicht zwangsläufig ein interessantes Finding sein. Wenn nämlich der erwartete Wert trotzdem unterhalb des Normwertes liegt, brauchen keine Sparmaßnahmen ergriffen zu werden. Andererseits kann auch ein voraussichtlicher Rückgang der Kosten interessant sein, falls der Wert weiterhin oberhalb des Normwertes liegt.

Anhand der bisher errechneten Größen kann man nun in einem dritten Schritt die möglichen Einsparungen berechnen. Zu den vorhergesagten Überschreitungen der Normwerte

wird der zugehörige Impact berechnet, der wiederum, mit einem bestimmten Faktor multipliziert, die voraussichtlichen Einsparungen ergibt. Dieser Faktor gibt prozentual an, wie groß die möglichen Einsparungen überhaupt sein können. So könnte es in einem Bereich z. B. sein, daß grundsätzlich maximal 5 % der Kosten eingespart werden können. In diesem Fall wären die voraussichtlichen Einsparungen das 0,05-fache des voraussichtlichen Impacts.

Das Ergebnis dieser Berechnungen ist allerdings nicht exakt. Ein Grund dafür ist das geringe Alter der Daten. Die zur Verfügung stehenden Daten sind alle nicht älter als zwei Jahre. Daher stehen für die Interpolation nicht viele Stützstellen zur Verfügung. Ein weiterer Grund für die Ungenauigkeit ist das einfache Interpolationsverfahren, das (wie oben erwähnt) durch ein exakteres ersetzt werden könnte. Deshalb ist es sinnvoll, entweder nur die Grössenordnung der voraussichtlichen Einsparungen anzugeben oder den Wert zusätzlich mit einer Verläßlichkeitsangabe zu versehen.

Die so gefundenen Abweichungen werden von KEFIR in einer Struktur namens Finding gespeichert. Ein Finding enthält zu einem Maß in einem Sektor Informationen wie:

- Abweichung von der Norm
- Trend (steigend oder fallend)
- Impact der Abweichung
- seine Beziehung zu anderen Findings
- verschiedene Verwaltungsinformationen

8.5 Sortieren der Findings

Um zu bestimmen, welches die Findings sind, die den Benutzer interessieren könnten, wird die zuvor berechnete Interessantheit herangezogen, da sie ein objektives Kriterium für dieses Problem darstellt. Die Findings werden deshalb nach ihrer Interessantheit sortiert und die ersten N davon werden für die Weiterbearbeitung ausgewählt. N kann dabei vom Benutzer beliebig festgelegt werden, die Voreinstellung ist $N = 10$. Die so ausgewählten Findings werden auch Key Findings genannt.

8.6 Erklärung

Bei der Suche nach einer Erklärung spielt einerseits der Herkunftssektor der Abweichung und andererseits die zur Berechnung der Interessantheit verwendete Formel eine wichtige Rolle. KEFIR verwendet dabei die in Kapitel 8.3 vorgestellte Aufteilung des Suchraumes in Sektoren. KEFIR überprüft, aus welchem Sektor die Daten stammen, die zu der Abweichung führen. Ferner wird festgestellt, welche Variablen in den verwendeten Formeln die Abweichung verursacht hat. Dadurch gelangt man zu weiteren Findings, von denen das mit der größten Interessantheit weiter untersucht wird: Innerhalb des Sektors wird

weiter überprüft, zu welchem Untersektor die Daten gehören, während bei den Formeln die Ursache für die Abweichung der Variablen gesucht wird.

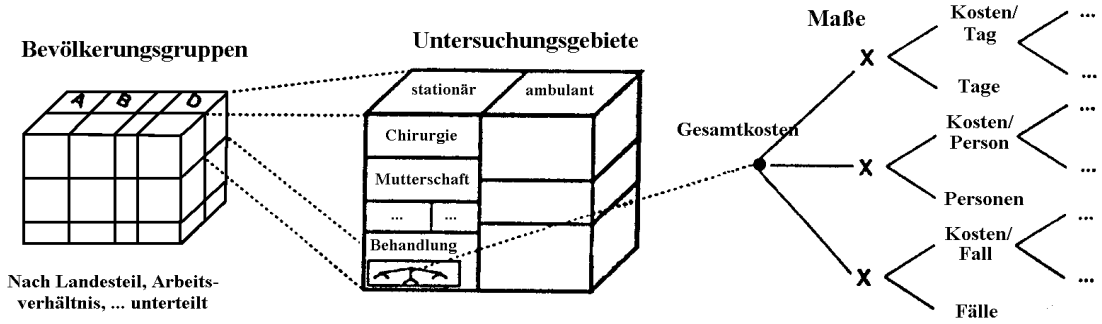


Abbildung 8.3: Abhängigkeit der Formeln vom Sektor

Letzteres ist in Abb. 8.3 dargestellt. Anhand der Aufteilung des Suchraumes in Bevölkerungsgruppen und medizinische Teilgebiete erhält man die Sektoreneinteilung. Für jeden dieser Sektoren gibt es bestimmte Formeln, durch die das zugehörige Maß definiert wird. Je nach Maß sind diese Formeln natürlich verschieden, da z. B. die durchschnittliche Dauer eines Krankenhausaufenthaltes anders berechnet wird als die durchschnittlichen Kosten einer Operation. Bei einem Maß müssen auch nicht notwendigerweise alle zur Verfügung stehenden Daten aus einem Sektor verwendet werden. Wenn man z. B. landesweit die durchschnittlichen Kosten pro Tag des Krankenhausaufenthaltes berechnen will, ist es nicht wichtig, wie lange die Patienten im Krankenhaus blieben, da die Tageskosten in den Krankenhäusern für jede Zimmerkategorie festgelegt werden.

Dies wird immer weiter verfeinert, bis entweder keine nennenswerte Abweichung mehr festgestellt wird oder eine bestimmte Suchtiefe erreicht wird, die vom Benutzer festgelegt werden kann. Die Voreinstellung ist eine Suchtiefe von drei Ebenen. Als Ergebnis dieser Analyse erhält man eine Menge von Abweichungen, die durch die Sektoreneinteilung und die verwendeten Formeln miteinander in Zusammenhang stehen. Anhand der dargestellten Zusammenhänge kann man schnell die Ursachen für mögliche Kostensteigerungen erkennen und geeignete Gegenmaßnahmen vorschlagen.

8.7 Empfehlung

An dieser Stelle wird das anfangs erwähnte Hintergrundwissen wichtig. Zu diesem Hintergrundwissen gehört unter anderem auch eine Art Regelsatz, mit dessen Hilfe die bei den Findings vorhandenen Informationen auf bestimmte Werte überprüft werden. So kann bei einer Regel z. B. nachgeprüft werden, was das Maß ist, wie der Sektor heißt und wie groß die Abweichung ist. Andere Regeln können komplexere Bedingungen enthalten. Je nach Ergebnis wird ein Vorschlag aus der vorhandenen Menge von Vorschlägen ausgewählt. Dies wird für alle Findings gemacht, die bei der Analyse als Ursachen von zu hohen Kosten entdeckt wurden. Die Menge der möglichen Vorschläge wurden von einem Experten fürs Gesundheitswesen formuliert, ebenso die Regeln, nach denen die Vorschläge ausgewählt werden. Zur Zeit sind ca. 35 Empfehlungen verfügbar, weitere sind jedoch in Arbeit. Per

WWW wird es auch Benutzern, die sich auf dem Gebiet auskennen, ermöglicht, weitere Empfehlungen hinzuzufügen oder vorhandene Empfehlungen zu ändern.

8.8 Erzeugung eines Berichts

In der Ausgabe werden die Key Findings, die zugehörigen Erklärungen und die Empfehlungen dargestellt. Dabei wird sowohl Fließtext als auch Tabellen, Balkendiagramme und Tortendiagramme verwendet, um dem Benutzer das Ergebnis so anschaulich wie möglich zu vermitteln. All diese Elemente werden auf einer WWW-Seite zusammengefaßt, die mit NCSA-Mosaic ausgegeben wird. Wenn das Ergebnis ausgedruckt werden soll, wird die Ausgabe in LaTeXvorgenommen und von dort aus nach Postscript konvertiert.

Abb. 8.4 und 8.5 zeigen einen Teil einer Beispiel-Ausgabe. Zunächst werden die Key Findings angegeben, wobei das Maß offensichtlich die durchschnittliche Aufenthaltsdauer war. Anhand eines Balkendiagrammes werden die bisherigen Werte (also für die Jahre 1992 und 1993) mit den zugehörigen Normwerten angegeben. Dazu kommt dann die Schätzung für das folgende Jahr, 1994. Im Fließtext wird erklärt, daß die durchschnittliche Aufenthaltsdauer gestiegen ist, und auch prozentual und absolut angegeben, um wieviel sie gestiegen ist. Dann wird erklärt, wieviel Geld 1993 hätte eingespart werden können, und welche zusätzlichen Kosten entstehen, wenn sich der Trend fortsetzt.

Bei der Suche nach Erklärungen für diesen Anstieg fand das Programm heraus, welcher medizinische Bereich eine besonders hohe Steigerung der Aufenthaltsdauer zu verzeichnen hatte, so daß die Gesamtdauer ebenfalls beträchtlich stieg. Bei der Empfehlung wird genauer erklärt, welche Krankheiten unter diesen medizinischen Bereich fallen. Aufgrund des Hintergrundwissens darüber, welche Sparmaßnahmen bereits durchgeführt werden, kommt Health-KEFIR zu dem Schluß, daß Maßnahmen im Bereich der Langzeitpflege besonders wichtig wären, und daß dadurch Einsparungen von 79000 US-Dollar möglich sind, unter der Annahme, daß ca. 30 % der erwarteten Mehrkosten eingespart werden können.

Im zweiten Teil der Ausgabe wurde das Ergebnis einer Überprüfung aller Zahlungen der Patienten dargestellt. Dabei wurde ein Zahlungsrückgang festgestellt, der hauptsächlich durch eine Verringerung der Zahlungen von Kreislaufpatienten verursacht wurde. Dies wird durch das Tortendiagramm deutlich, in dem die Zahlungen der Patienten nach Behandlungsgebiet aufgeschlüsselt sind. In der Tabelle wird aufgeführt, wie groß die Kosten 1992 und 1993 tatsächlich waren, und für 1993 zum Vergleich der Normwert mit angegeben. Da die tatsächlichen Werte unterhalb der Norm liegen, braucht keine Empfehlung für zukünftige Einsparungen gegeben zu werden.

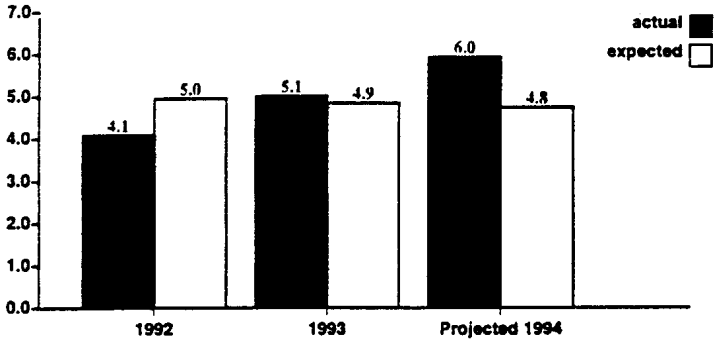
8.9 Implementierung von KEFIR

KEFIR wurde vollständig in tcl und in C geschrieben. Der Datenzugriff erfolgt über eine SQL-Schnittstelle. Für die Darstellung des Ergebnisses werden mehrere HTML- und GIF-Dateien erzeugt, die, wie bereits erwähnt, mit NCSA-Mosaic ausgegeben werden.

Medical Admissions

Significant Findings

Actual and expected values for average length of stay:



Average length of stay in this study area rose 22.6 percent, from 4.1 to 5.1. Around $36,000 could have been saved if average length of stay had been equal to the expected value of 4.9. If current conditions continue into the next period, this trend will result in $263,000 of additional expenses.

Explanations: The increase in average length of stay in this study area is related to the fact that average length of stay in Medical Nervous System (MDC=01) rose 247.9 percent, from 4.4 to 15.2.

Recommendation: These admissions are for medical disorders, conditions like diabetes, heart attack, stroke. Non-medical admissions, e.g. mental health, surgical, maternity, are easier to control because there are specific care management programs. Programs that are disease or condition specific are also starting to develop. Chronic care management is probably the most important to concentrate on for medical admissions.

Projected savings: **$79,000** (based on an expected payoff of 30%).

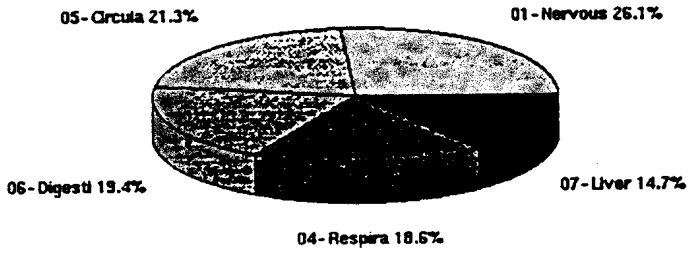
Abbildung 8.4: Beispiel-Ausgabe von KEFIR

Total Payments Summary

Total inpatient payments in this study area fell 22.5 percent, from $1.4 million to $1.1 million. The fact that total inpatient payments was better than the expected value of $1.2 million accounted for a savings of $147,000. If current conditions continue into the next period, this trend will result in $388,000 of savings.

The decrease in total inpatient payments in this study area is related to the fact that total inpatient payments in Medical Circulatory System (MDC=05) fell 58.7 percent, from $370,910 to $153,287. The observed deviation from the expected value is related to the fact that payments per day was 8.1% below the expected value ($910.68 versus $1,479.59).

Percent of Total Inpatient Payments by Top 5 MDC by payments



Measures Table

Measure	1992	1993	Change	1993 Norm	Diff.
Total payment	$1,415,839	$1,097,380	-22.5%	$1,244,343	-11.8%
Pay/capita	$135	$125	-7.8%	$141	-11.8%
Covered lives	10441	8778	-15.9%		
Episodes	299	237	-20.7%		
Average LOS	4.1	5.1	22.6%	4.9	3.4%
Pay/day	$1,141	$910	-20.2%	$1,479	-38.5%
Adms/1000	28.6	27.0	-5.7%	19.5	38.6%
Readmit-30 rt	12.4	16.0	29.6%		
Mjr adv rt	3.3	2.5	-24.3%		

 Medical Admissions

Abbildung 8.5: Beispiel-Ausgabe von KEFIR

Ein Benutzer kann natürlich auch einen anderen WWW-Client benutzen, um sich die Ausgabe anzusehen, aber die Workstation, auf der Health-KEFIR läuft, ruft dazu Mosaic auf. Die erste kommerziell verfügbare Version von Health-KEFIR wurde im Januar 1995 fertiggestellt.

8.10 Bewertung

Im Vergleich zu den Berichten menschlicher Experten fürs Gesundheitswesen schneidet KEFIR sehr gut ab. Alle Problembereiche, die von diesen entdeckt wurden, hat auch KEFIR gefunden, und oft noch einige, die von den Menschen übersehen wurden, für den Benutzer aber dennoch interessant sind.

Für die Leistung von Health-KEFIR wesentlich sind jedoch verschiedene Dinge, die weiter verbessert werden können. Wie bereits erwähnt ist die Interpolation der vorhandenen Stützdaten bei der Vorhersage von Abweichungen keinesfalls ideal. Man muß sich jedoch fragen, ob eine exaktere Interpolation wirklich große Vorteile bringen würde, da das Ziel von Health-KEFIR, Problembereiche im Gesundheitswesen aufzuzeigen, auch mit dieser einfachen Interpolation erreicht wird und bei einem komplexeren Verfahren lediglich die für die Berechnung benötigte Zeit größer würde. Je nachdem, ob der Benutzer nur den Bereich des Problems feststellen will oder an einer genaueren Angabe der möglichen Einsparungen interessiert ist, könnte man verschiedene Verfahren anwenden, die vom Benutzer ausgewählt werden können.

Weitaus wichtiger ist die Vervollständigung der Datenbank durch die Daten weiterer Jahre, so daß für die Berechnung der Vorhersage mehr Stützstellen vorhanden sind. Außerdem scheint ein Ausbau und eine Verfeinerung des Hintergrundwissens notwendig, da mit 35 möglichen Vorschlägen für Sparmaßnahmen sicher nicht alle Möglichkeiten abgedeckt sind. Health-KEFIR mag also zwar bereits beachtliche Leistung bringen, ist aber durchaus noch verbesserungsfähig.

Literaturverzeichnis

[1] Y. Abu-Mostafa. The Vapnik-Chervonenkis dimension: Information versus complexity in learning. *Neural Computation*, 1, 1989.

[2] F. Bernstein, T. Koetzle, G. Williams, E. Meyer, M.Brice, J. Rodgers, O. Kennard, T. Shimanouchi, and M. Tasumi. The protein data bank: a computer-based archival file for macromolecular structures. *Journal of Molecular Biology*, 112:535–542, 1977.

[3] I. Bhandari. Attribut focusing: Machine-assisted knowledge discovery applied to software production process. *Knowledge Acquisition*, 6, 1994.

[4] W. Buntine. Graphical Models for Discovering Knowledge. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*. A Bradford Book, The MIT Press, 1995.

[5] Y. Cai, N. Cercone, and J. Han. An Attribut–Oriented Approach for Learning Classification Rules from Relational Databases. In *Proc. 6th Intl. Conf. on Data Engineering*, pages 281–288, February 1990.

[6] Y. Cai, N. Cercone, and J. Han. Attribute-oriented induction in relational databases. In G. Piatetsky-Shapiro and W. J. Frawley, editors, *Knowledge Discovery in Databases*, pages 213–228. AAAI/MIT Press, 1991.

[7] R.M. Cameron-Jones and J.R. Quinlan. Efficient Top-down Induction of Logic Programs. In *SIGART*, volume 5, pages 33–42, 1994.

[8] E. Charniak. Bayes Networks without Tears. *AI Magazine*, 12(4).

[9] P. Clark and R. Boswell. Rule Induction with CN2: Some Recent Improvements. In Y. Kodratoff, editor, *Machine Learning – EWSL-91*, pages 151–163, Porto, Portugal, March 1991. Springer Verlag.

[10] Son Dao and Brad Perry. Applying a Data Miner to Heterogenous Schema Integration. In U. M. Fayyad and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining. Proc. First Interntl. Conf. on Knowledge Discovery and Data Mining*, pages 213–228. AAAI Press, Menlo Park, CA, 1995.

[11] R.O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis*. Wiley and Son, 1973.

[12] S. Džeroski and I. Bratko. Handling noise in inductive logic programming. In *Proceedings of the Second International Workshop on Inductive Logic Programming*, Tokyo, Japan, 1992. ICOT.

[13] U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors. *Advances in Knowledge Discovery and Data Mining.* A Bradford Book, The MIT Press, 1995.

[14] U. M. Fayyad and R. Uthurusamy, editors. *Proceedings of KDD-94: the AAAI-94 workshop on Knowledge Discovery in Databases* , Menlo Park, 1994. The AAAI Press.

[15] U. M. Fayyad and R. Uthurusamy, editors. *Advances in Knowledge Discovery and Data Mining. Proc. First Interntl. Conf. on Knowledge Discovery and Data Mining.* AAAI Press, Menlo Park, CA, 1995.

[16] E.A. Feigenbaum and Simon H. EPAM-like models of recognition and learning. *Cognitive Science*, 8:305–336, 1984.

[17] D. Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, pages 139–172, February 1987.

[18] D. Fisher. Knowledge acquisition via incremental conceptual clustering. Technical Report 87-22, Department of Information and Computer Science, University of California, Irvine, CA, 1987.

[19] P. Flach. Predicate invention in inductive data engineering. In *Proceedings of the Sixth European Conference on Machine Learning*, pages 83–94, Berlin, Germany, 1993. Springer.

[20] W. Frawley, G. Piatetsky-Shapiro, and C. Matheus. Knowledge discovery in databases: An overview. In G. Piatetsky-Shapiro and W. Frawley, editors, *Knowledge Discovery in Databases.* AAAI Press/The MIT Press, 1991.

[21] W.J. Frawley, G. Piatetsky-Shapiro, and C.J. Matheus. Knowledge Discovery in Databases: An Overview. In G. Piatetsky-Shapiro and W.J. Frawley, editors, *Knowledge Discovery in Databases*, pages 1–29. AAAI/MIT press, July 1991.

[22] W.J. Frawley, G. Piatetsky-Shapiro, and C.J. Matheus. Knowledge Discovery in Databases: An Overview. *AI Magazine*, 13(3):57–70, 1992.

[23] John H. Gennari, Pat Langley, and Doug Fisher. Models of Incremental Concept formation. *Artificial Intelligence*, 1989.

[24] W.R. Gilks, A. Thomas, and D.J Spiegelhalter. A language and program for complex Bayesian modelling. *The Statistician*, 35(2):197–226, 1993.

[25] I. Guyon, B. Boser, and V. Vapnik. Automatic capacity tuning of very large VC-dimension classifiers. In S. Hanson, editor, *Advances in Neural Information Processing Systems 5.* Morgan Kaufmann, 1993.

[26] I. Guyon, N. Matić, and V. Vapnik. Discovering informative patterns and data cleaning. In U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining.* AAAI Press, 1995.

[27] J. Hartung. *Statistik*. Oldenbourg, 1993.

[28] Edmund Klatt. *Langenscheidts Taschenwörterbuch der englischen und deutschen Sprache*. Langenscheidt, 6. neubearbeitung, 79. auflage edition, 1986.

[29] D. Kneller, F. Cohen, and R. Langridge. Improvements in protein secondary structure prediction by an enhanced neural network. *Journal of Molecular Biology*, 214:171–182, 1990.

[30] I. Kononenko, I. Bratko, and E. Roskar. Experiments in automatic learning of medical diagnostic rules. Technical report, Ljubljana, Yugoslavia, E. Krdelj University, Fauclty of Electrical Engineering, 1984.

[31] N. Lavrač and S. Džeroski. *Inductive Logic Programming: Techniques and Applications*. Ellis Horwood, Chichester, UK, 1994.

[32] N. Lavrač, S. Džeroski, and M. Grobelnik. Learning nonrecursive definitions of relations with LINUS. In *Proceedings of the Fifth European Working Session on Learning*, pages 265–281, Berlin, Germany, 1991. Springer.

[33] Michael Lebowitz. UNIMEM, A General Lerning System: An Overview. In *Advances in Artificial Intelligence - II*, 1987.

[34] Lockemann, Krüger, and Krumm. *Telekommunikation und Datenhaltung*. Hanser Verlag, 1993.

[35] Meyers. *Enzyklopädisches Lexikon*, volume 18. Bibliographisches Institut Mannheim, 1976.

[36] K. Morik, S. Wrobel, J.-U. Kietz, and W. Emde. *Knowledge Acquisition and Machine Learning*. Academic Press, New York, 1993.

[37] S. Muggleton. *Inductive Logic Programming*. Academic Press, London, UK, 1992.

[38] S. Muggleton and W. Buntime. Machine invention of first-order predicates by inverting resolution. In *Proceedings of the Fifth International Conference on Machine Learning*, pages 339–352, San Mateo, CA, 1988. Morgan Kaufmann.

[39] S. Muggleton and C. Feng. Efficient induction of logic programs. In *Proceedings of the First Conference on Algorithmic Learning Theory*, pages 368–381, Tokyo, Japan, 1990. Ohmsha.

[40] S. Muggleton, R. King, and M. Sternberg. Protein secondary structure prediction using logic. *Protein Engineering*, 5:647–657, 1992.

[41] S. Muggleton and A. Srinivasan. Mode-directed inverse resolution. In D. Michie, S. Muggleton, and K. Furukawa, editors, *Machine Intelligence*, volume 14. Clarendon Press, Oxford, UK, 1995.

[42] M. Pazzani and D. Kibler. The Utility of Knowledge in Inductive Learning. *Machine Learning*, 6:57–94, 1992.

[43] J. Pearl. *Probabilistic Reasoning in Intelligent Systems.* Morgan Kaufmann, 1988.

[44] G. Piatetsky-Shapiro. Report on AAAI-91 workshop on Knowledge Discovery in Databases. *IEEE Expert*, 6(5):74–76, 1991.

[45] G. Piatetsky-Shapiro, editor. *Proceedings of KDD-93: th AAAI-93 workshop on Knowledge Discovery in Databases* , Menlo Park, 1993. The AAAI Press.

[46] G. Piatetsky-Shapiro and W. Frawley, editors. *Knowledge Discovery in Database* . AAAI/MIT Press, Cambridge, 1991.

[47] G. Plotkin. A note on inductive generalization. In B. Meltzer and D. Michie, editors, *Machine Intelligence*, volume 5, pages 153–163. Edinburgh University Press, Edinburgh, UK, 1969.

[48] John R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, 1993.

[49] L. De Raedt. *Interactive Theory Revision: An Inductive Logic Programming Approach.* Academic Press, London, UK, 1992.

[50] L. De Raedt and M. Bruynooghe. A theory of clausal discovery. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, pages 1058–1063, San Mateo, CA, 1993. Morgan Kaufmann.

[51] L. De Raedt, N. Lavrač, and S. Džeroski. Multiple Predicate Learning. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, pages 1037–1042, San Mateo, CA, 1993. Morgan Kaufmann.

[52] Michalski R.S., J.G. Carbonell, and T.M. Mitchell. *Machine Learning: An Artificial Intelligence Approach.* Tioga Publishing Company, Palo Alto, 1983.

[53] E. Shapiro. *Algorithmic Program Debugging.* MIT Press, Cambridge, MA, 1983.

[54] Ralf Steuernagel. Offenes adaptives Engineering-Werkzeug zur automatischen Erstellung von entscheidungsunterstützenden Informationssystemen. Technical Report Band 60, Universität Karlsruhe, 1994.

[55] J. Whittaker. Graphical Models in Applied Multivariate Statistics. Wiley.

[56] R. Wirth and T.P. Reinartz. Towards a task model for knowledge discovery in databases. Technical report, Daimler Benz Research & Technologie F3S/E, 1995.

[57] W. Ziarko. *Rough Stes, Fuzzy Stes and Knowledge Discovery.* Springer Verlag, Berlin, 1994.