Lehrstuhl für Informatik I
Informationsstrukturen und wissensbasierte Systeme
Universität Würzburg

Prof. Dr. H. Noltemeier
Dr. S. O. Krumke

Würzburg, den 11. Februar 1998

Informationsstrukturen und wissensbasierte Systeme
(WS 97/98)

2. Fallstudie

1 Generelles

Auch in der zweiten Fallstudie sollen wiederum ausgewählte Algorithmen von den einzelnen Arbeitsgruppen implementiert und experimentell untersucht werden. Es gelten die gleichen Anforderungen wie bei der ersten Fallstudie, die hier noch einmal kurz aufgeführt werden.

Jede Arbeitsgruppe besteht aus maximal zwei Studenten. Die von den Gruppen entwickelten Programme werden von den studentischen Hilfskräften abgenommen. Wir stellen wieder konkrete Instanzen des jeweiligen Problems zur Verfügung, die bei der Abnahme – neben anderen Beispielen – vorgeführt werden sollen.

Von jeder Gruppe wird ferner eine schriftliche Ausarbeitung erwartet. In dieser Ausarbeitung sollen kurz die Aufgabenstellung, notwendige Definitionen und die Ergebnisse zusammengefaßt werden.

1.1 Rechnerzeiten im CIP-Pool

Für die Programmierung stehen die CIP-Pools SE05 und SE06 mittwochs von 8 bis 12 Uhr zur Verfügung. Zu diesen Zeiten sind auch die studentischen Hilfskräfte anwesend, um Fragen zu beantworten.

Eine Beratung zu den Fallstudien findet durch die studentischen Hilfskräfte jeweils freitags im S125 in den Übungsgruppen statt.

1.2 Zeitplan

Die Abnahme der Programme erfolgt durch die studentischen Hilfskräfte im CIP-Pool jeweils zu den Rechnerzeiten bis zum 31. März 1998. Die schriftlichen Ausarbeitungen sind bis zum gleichen Termin ebenfalls bei den studentischen Hilfskräften abzugeben.

1.3 Richtlinien für die Programme

1. Implementierungssprache für die Programme ist C++. Für die Programmierung steht der GNU C++-Compiler zur Verfügung.
2. Es wird übersichtlicher und gut dokumentierter Code erwartet.
3. Zum Programm ist ein „Makefile“ zu erzeugen, mit dem das Compilieren/Linken erfolgt.

2 Projekt: Logische Analyse von Daten

In der zweiten Fallstudie ist das Projekt „Logische Analyse von Daten“ zu bearbeiten.

2.1 Übersicht

Ziel der *logischen Analyse von Daten*, kurz *LAD*, ist es, strukturelle Informationen über Datenmengen herauszufinden. Man versucht, aus vorhandenen Observationen logische Schlüsse in Form von aussagenlogischen Formeln herzuleiten und dann neue Observationen so zu klassifizieren, daß sie konsistent mit den bisherigen sind.

Im Rahmen des Projektes „Logische Analyse von Daten“ sollen Algorithmen für LAD implementiert und experimentell evaluiert werden. Dabei sollen sowohl Daten mit binären Attributen als auch mit numerischen Attributen betrachtet werden.

2.2 Kurze Darstellung der Grundbegriffe

Die Eingabe für das LAD besteht aus einem Archiv von bekannten Observationen S. Jedes Element aus S ist dabei ein n-dimensionaler Attributvektor. Darüberhinaus ist für jedes $x \in S$ bekannt, ob es eine positive oder negative Observation ist. Somit ist S in die Menge S^+ der positiven und die Menge S^- der negativen Observationen partitioniert.

Im einfachsten Fall ist $S \subseteq \{0,1\}^d$. Dann kann man jedes Archiv (S^+, S^-) als eine *partiell definierte Boolesche Funktion (PBF)* $\phi = (T, F)$ repräsentieren. Dabei ist $T = S^+$ die Menge der 1-Stellen und $F = S^-$ die Menge der 0-Stellen von ϕ. Jede (totale) boolesche Funktion $f\colon \{0,1\}^d \to \{0,1\}$, die mit ϕ auf S übereinstimmt, nennt man dann eine *Fortsetzung* von ϕ.

Ultimatives Ziel des LAD ist es, eine Forsetzung f_ϕ von ϕ zu finden, die alle unbekannten Observationen richtig klassifiziert. Da dieses Ziel im allgemeinen nicht erreichbar ist, versucht man stattdessen, eine Funktion f zu finden, die f_ϕ „möglichst gut annähert“. Dabei bedeutet „gute Annäherung“, daß f in einer bestimmten Klasse Boolescher Funktionen liegt oder möglichst einfache Struktur besitzt.

Eine gute Einführung in LAD findet sich in [2, 3].

Muster und Klassifizierung durch LAD bei binären Daten

Es sei $S \subseteq \{0,1\}^d$, d.h., jede Observation sei binär. Jeder Punkt $p \in \{0,1\}^d$ entspricht in natürlicher Weise einer Belegung der atomaren Formeln $A_1, \ldots, A_d$. Man nennt A_i und $\neg A_i$ *Literale*. Das Literal A_i ist *positiv*, das Literal $\neg A_i$ *negativ*. Ein *Term* ist eine Konjunktion von Literalen, sein *Grad* ist die Anzahl der Literale im Term.

Ein Term T *überdeckt* einen Punkt $p \in \{0,1\}^d$, wenn $T(p) = 1$ ist, d.h. wenn bei Belegung der atomaren Formeln gemäß p der Term den Wahrheitswert 1 besitzt. Der *charakteristische Term* eines Punktes $p \in \{0,1\}^d$ ist der Term $L_1 \cdots L_d$ mit $L_i = A_i$, falls $p_i = 1$, und $L_i = \neg A_i$ falls $p_i = 0$.

Ein Term T ist ein *Implikant* einer Booleschen Funktion f, wenn aus $T(p) = 1$ folgt, daß $f(p) = 1$ gilt. Ein Implikant T heißt *prim*, wenn der Term minimal ist, d.h., wenn durch Entfernen eines beliebigen Literals aus T die Implikanteneigenschaft von T verlorengeht.

Für eine PBF $\phi = (S^+, S^-)$ kann man in natürlicher Weise zwei extreme Fortsetzungen ϕ^+ und ϕ^- definieren:

$$\phi^+(p) := \begin{cases} 1 & \text{, falls } p \notin S^- \\ 0 & \text{, sonst,} \end{cases}$$

$$\phi^-(p) := \begin{cases} 1 & \text{, falls } p \in S^+ \\ 0 & \text{, sonst.} \end{cases}$$

Die primen Implikanten von ϕ^+, welche mindestens einen Punkt aus S^+ überdecken, heißen *positive Muster von* ϕ. Die *negativen Muster von* ϕ sind die positiven Muster von $\bar{\phi}$, des Komplements von ϕ.

Beim LAD generiert man die positiven und negativen Muster einer PBF ϕ bis zu einem (problemspezifischen) Höchstgrad. Danach versucht man, aus der Kenntnis der Muster eine *Theorie*, d.h. eine Klassifizierungsregel zu erstellen.

Die in [3] vorgeschlagene Methode, um zu einer Theorie zu gelangen, sieht wie folgt aus: Seien P und N Mengen positiver bzw. negativer Muster, die alle Punkte aus S^+ bzw. S^- überdecken. Man wählt geeignete Gewichte $w_x^+ \geq 0$ und $w_y^- \leq 0$ und klassifiziert neue Observationen mit Hilfe folgender *Diskriminanten:*

$$\Delta = \sum_{x \in P} w_x^+ x + \sum_{y \in N} w_y^- y.$$

Falls $|\Delta|$ „groß genug" ist, so ist die neue Observation mit $\Delta < 0$ ein negatives Beispiel, mit $\Delta > 0$ ein positives Beispiel. Falls $|\Delta|$ klein ist, so wird die neue Observation nicht klassifiziert.

Um den Rechenaufwand zur Berechnung der Diskriminanten klein zu halten, sollen möglichst wenige positive und negative Muster in P und N gewählt werden, die positive und negative Observationen trennen. Um solche Mengen P und N zu finden, muß ein ganzzahliges Lineares Programm bzw. ein MinSetCover Problem gelöst werden. Informationen zu Linearen Programmen und zu MinSetCover folgen in den nächsten Abschnitten.

Zur Wahl von geeigneten Gewichten (cf. [3]): Eine einfache Möglichkeit, die Gewichte zu wählen, ist $w_x^+ := 1/|P|$ für alle $x \in P$ und $w_y^- := -1/|N|$ für alle $y \in N$ zu setzen. Ein genauerer Ansatz erfolgt über Lineare Programmierung (siehe [3, Seite 10]).

LAD bei numerischen Daten

Bisher wurde nur LAD bei binären Daten angesprochen. Die Techniken des LAD lassen sich jedoch auch auf allgemeinere Daten anwenden, bei denen jedes Attribut einen numerischen Wertebereich besitzt.

In diesem Fall werden die Daten aus den Observationen durch *Einführung von Trennwerten binarisiert.* In der Fallstudie soll hier der Ansatz aus [2] verfolgt werden. Bei diesem Ansatz werden die numerischen Daten durch Einführung neuer binärer Attribute in binäre Daten gewandelt.

Die binären Attribute sind sogenannte *Level-* oder *Indikator-Variablen.* Da die Menge der Observationen endlich ist, nimmt jedes Attribut nur endlich viele Werte an. Für ein Attribut A seien $u_A^{(0)} > u_A^{(1)} > \cdots > u_A^{(K)}$ die verschiedenen angenommenen Werte aller Observationen aus $S^+ \cup S^-$. Man ordnet nun dem Attribut eine Anzahl von *Trennwerten* α_{Ai} und zugehörige binäre Indikator-Variablen x_{Ai} gemäß

$$x_{A_i} := \begin{cases} 1 & \text{, falls } A \geq \alpha_{Ai}, \\ 0 & \text{, falls } A < \alpha_{Ai} \end{cases}$$

zu. Offenbar liefern alle Trennwerte, die zwischen $u_A^{(i)}$ und $u_A^{(i+1)}$ liegen, die gleichen Werte für die Indikator-Variablen. Außerdem liefern Trennwerte oberhalb von $u_A^{(0)}$ oder unterhalb von $u_A^{(K)}$ keine Information. Folglich genügt es, als Trennwerte die Werte $\alpha_{Ai}(u_A^{(i)} + u_A^{(i+1)})/2$, $i = 0, \ldots, K-1$ zu wählen, für die es $u \in S^+$ und $u' \in S^-$ mit $u_A = u_A^{(i)}$ und $u'_A = u_A^{(i+1)}$ (oder umgekehrt) gibt.

Werden nach der oben skizzierten Methode für ein Attribut A K' Trennpunkte erzeugt, so entspricht das einer Umsetzung der Werte u_A in binäre Vektoren der Dimension K'.

Falls für jedes Attribut die maximal sinnvolle (s.o.) Anzahl von Trennpunkten eingeführt wird, so nennt man die entstehende PBF $\phi = (T, F)$ die *Master-PBF* von (S^+, S^-).[1] Eine Menge von Trennpunkten heißt *konsistent*, falls für die entstehende PBF $\phi = (T, F)$ die Mengen T und F disjunkt sind.

Um die Anzahl der (neu) generierten binären Attribute klein zu halten, soll eine möglichst kleine Menge von Trennpunkten bestimmt werden, die dennoch konsistent ist. Diese Aufgabe führt wieder auf ein ganzzahliges Lineares Programm oder ein MinSetCover Problem.

Lineare Programme

Ein Lineares Programm (LP) besteht darin, eine lineare Zielfunktion über einem Polyeder zu optimieren. Für lineare Programme gibt es mehrere Standardformen, die jedoch einfach ineinander konvertiert werden können. Eine Standardform ist das sogenannte Covering-LP:

$$
\begin{array}{lll}
\text{(CLP)} & \min c^T x & \\
& a_i^T x \geq b_i, & \text{für } i = 1, \ldots, m, \\
& x \geq 0. &
\end{array}
$$

Hierbei ist $c \in \mathbb{Q}^n$, $b = (b_1, \ldots, b_m)^T \in \mathbb{Q}^m$ und $a_i \in \mathbb{Q}^n$ für $i = 1, \ldots, m$. Das Problem besteht darin, ein $x^* \in \mathbb{R}^n$ mit $Ax^* \geq b$ zu finden, so daß

$$c^T x^* = \min\{ c^T x : Ax \geq b \}.$$

Lineare Programme in der oben genannten Form können effizient (insbesondere auch in Polynomialzeit) gelöst werden. Der Name Covering-LP leitet sich aus der Notwendigkeit ab, $x \geq 0$ so „groß“ zu wählen, daß in allen Nebenbedingungen im Skalarprodukt $a_i^T x$ mindestens b_i erreicht, also „abgedeckt“ wird.

Bei den Covering-LPs in der Fallstudie werden zusätzlich Ganzzahligkeitsforderungen an den Lösungsvektor gestellt. Sie besitzen die folgende Form:

$$
\begin{array}{lll}
\text{(CIP)} & \min c^T x & \\
& a_i^T x \geq b_i, & \text{für } i = 1, \ldots, m, \\
& x \in \{0, 1\}^n. &
\end{array}
$$

Im Gegensatz zur kontinuierlichen Variante (CLP) ist es im allgemeinen[2] NP-hart, ein (CIP) zu lösen.

Zum Lösen von (CIP) steht für die Fallstudien die C-Bibliothek LP_SOLVE in der Version 2.2 zur Verfügung. Im Abschnitt 2.3 wird auf diese Bibliothek noch weiter eingegangen.

Ausführliche Informationen über Lineare Programmierung findet man etwa in [6,7].

[1] Man beachte, daß hier nicht wie im binäre Fall $T = S^+$ und $F = S^-$ gilt, da $S = S^+ \cup S^-$ Daten mit allgemeinen numerischen und nicht notwendigerweise binären Attributen enthält.

[2] d.h. ohne zusätzliche Forderungen an die Daten c, b und die a_i

Das MinSetCover Problem

Ein Problem, welches in der Fallstudie beim Finden von Theorien und minimalen Trägermengen auftritt und das als (CIP) formuliert werden kann, ist das sogenannte MinSetCover Problem. Eine Instanz von MinSetCover ist durch eine endliche Grundmenge $U = \{u_1, \dots, u_m\}$, ein System $\mathcal{F} = \{F_1, \dots, F_r\} \subset 2^U$ von Teilmengen von U und Gewichte $c_1, \dots, c_m$ für die Mengen in $\mathcal{F}$ gegeben. Die Aufgabe, besteht darin, eine Teilmenge $\mathcal{C} \subseteq \mathcal{F}$ mit $\bigcup_{F_j \in \mathcal{C}} F_j = U$ und minimalem Gewicht $\sum_{F_j \in \mathcal{C}} c_j$ zu finden. MinSetCover ist ein klassisches NP-hartes Problem.

Man kann MinSetCover als (CIP) formulieren, indem wir r binäre Variablen x_j einführen, wobei uns $x_j = 1$ angibt, daß die Menge F_j zur Überdeckung $\mathcal{C}$ gehören soll, und $x_j = 0$ bedeutet, daß $F_j \notin \mathcal{C}$ gelten soll.

Wir wählen dann in unserer Formulierung als (CIP) a_{ij} (d.h. den j-ten Eintrag von a_i) genau dann als 1, wenn $u_i \in F_j$ gilt. Ferner setzen wir $b = (1, \dots, 1)^T$. Die Nebenbedingung $a_i^T x \geq b_i$, also $\sum_{j: u_i \in F_j} x_j \geq 1$ bedeutet dann, daß für das Element u_i mindestens eine Menge F_j mit $u_i \in F_j$ zum Cover gehören muß.

Übersichtlich schreibt sich dann MinSetCover als (CIP) wie folgt:

$$
\text{(MSC)} \qquad
\begin{aligned}
& \min \sum_{j=1}^{r} c_j x_j \\
& \sum_{j: u_i \in F_j} x_j \geq 1 && \text{für } i = 1, \dots, m \\
& x_j \in \{0, 1\} && \text{für } j = 1, \dots, r.
\end{aligned}
$$

Zur Lösung einer Instanz von MinSetCover kann man also das Problem als (CIP) formulieren und dann einen Algorithmus zur Lösung von ganzzahligen Linearen Programmen benutzen. Wie oben erwähnt, steht für die letztgenannte Aufgabe die Bibliothek `lp_solve` (siehe Abschnitt 2.3 zur Verfügung.

Eine Alternative ist die approximative Lösung des MinSetCover-Problems. Es ist bekannt, daß ein einfacher Greedy-Algorithmus (siehe z.B. [4, Kapitel 37]) benutzt werden kann, um eine Lösung zu finden, deren Gewicht höchstens $1 + \ln f_{\max}$ mal so groß ist wie das des optimalen Covers. Hierbei ist $f_{\max} = \max_{1 \leq j \leq r} |F_j|$ die Kardinalität der größten Menge aus $\mathcal{F}$.

Interessant ist übrigens, daß gezeigt werden konnte [1, 5], daß die Approximationsgüte von $1 + \ln f_{\max}$ des Greedy-Algorithmus im wesentlichen bestmöglich ist. Man beachte, daß $1 + \ln f_{\max} \leq 1 + \ln m$ gilt. Arora und Sudan konnten in [1] beweisen, daß es ein $\alpha > 0$ gibt, so daß eine Approximation in Polynomialzeit mit Güte $\alpha \ln m$ nur dann möglich ist, wenn $\mathsf{P} = \mathsf{NP}$ gilt.

2.3 Algorithmen

Enumerationsalgorithmus für Prim-Muster Dieser Algorithmus aus [3] bestimmt zu einer vorgegeben PBF (T, F) alle positiven und/oder negativen Muster.

Theoriebildung für binäre Daten Ausgehend von den positiven und negativen Mustern wird eine geringe Anzahl von relevanten Mustern zur Bildung einer Theorie herangezogen [3].

Klassifizierung Neuen Fällen werden durch die Bildung der Diskriminanten klassifiziert [3].

Lösen von Linearen Programmen Zur Bestimmung optimaler Auswahlen an Mustern oder optimalen Trennpunkten müssen ganzzahlige lineare Programme gelöst werden. Ein „Solver“ steht mit `LP_SOLVE` zur Verfügung (Details im Abschnitt weiter unten).

Greedy Heuristik für MinSetCover Eine approximative Lösung der Überdeckungsprobleme kann durch die Greedy-Heuristic für MinSetCover [4] erfolgen. Diese Heuristik zeichnet sich durch einfache Implementierung, schnelle Laufzeit und im wesentlichen bestmögliche Gütegarantie aus.

Binarisierung Numerische Daten werden durch Binarisierung [2] in binäre Daten „konvertiert", um dann mit den Booleschen Methoden des LAD verarbeitet zu werden.

Die Bibliothek `LP_SOLVE`

Die C-Bibliothek `LP_SOLVE` enthält Algorithmen zum exakten Lösen von kontinuierlichen oder ganzzahligen LPs. Die Algorithmen besitzen zwar im Worst-Case exponentielle Laufzeit, liefern jedoch in der Praxis dennoch akzeptable Rechenzeiten.

Um mit der Bibliothek arbeiten zu können, sind die Bibliotheksdatei `libplk.a` und die Header-Datei `lp_iws.h` nötig. Die Header-Datei `lp_iws.h` ist dabei eine an C++ angepaßte Version der mit der Bibliothek gelieferten C-Headerdatei `lpkit.h`.

Für die Fallstudien stellen wir bereits fertig übersetzte Versionen der Bibliothek für Solaris 2.5 auf Sparc, OSF1 V4.0 auf DEC Alpha und Linux Kernel 2.0.x auf i386 zur Verfügung. Diese sind entweder über die Web-Seite zur Vorlesung `http://www-info1.informatik.uni-wuerzburg.de/vorlesungen/IWS/iws.html` oder über die studentischen Hilfskräfte erhältlich. Auf gleichem Wege können Sie auch die Header-Datei `lp_iws.h` bekommen. Falls Sie die Quelltexte für die Bibliothek benötigen, wenden Sie sich bitte an die studentischen Hilfskräfte.

Weitere Dokumentation zu `lp_solve` und ein kurzes Beispielprogramm in C++ stellen wir Ihnen sowohl auf der Web-Seite als auch über die studentischen Hilfskräfte zur Verfügung.

2.4 Aufgaben

Die Leistungsfähigkeit von LAD und der Einfluß bestimmter Parameter bei der Theoriegenerierung soll anhand von ausgewählten Datensätzen evaluiert werden. Dabei soll die numerischen Experimente wie folgt ablaufen:

Aus einem Datensatz sollen jeweils 50% bzw. 75% der Observationen als „Trainingsmenge", d.h. als Archiv bekannter Observationen, zufällig gewählt werden. Die Genauigkeit der LAD Techniken ist dann anhand des Komplements der Trainingsmenge zu prüfen.

Insbesondere ist zu untersuchen, wie stark die Genauigkeit des LAD von folgenden Faktoren abhängt:

1. Größe der Trainingsmenge (50% oder 75%)
2. Wahl der Gewichte (einfach als $w_x := 1/|P|$ und $w_y := -1/|N|$ oder über Lineare Programmierung)
3. Genauigkeit beim Lösen des MinSetCover Problems bei der Auswahl von überdeckenden Mustern (exakt durch ganzzahlige Lineare Programmierung oder approximativ durch Greedy-Heuristik)
4. Genauigkeit beim Finden einer minimalen Anzahl von Trennpunkten (exakt durch ganzzahlige Lineare Programmierung oder approximativ durch Greedy-Heuristik)

Bei den approximativen Lösungen der auftretenden MinSetCover Probleme soll auch festgestellt werden, ob sich die damit verbundene Vergrößerung der Muster- bzw. Trennpunktmenge negativ auf Laufzeit- und Speicherplatbedarf auswirkt.

Die oben aufgeführten Fragestellungen sollen anhand der folgenden konkreten Datensätze untersucht werden (die Datensätze sind entweder über die studentischen Hilfskräfte oder über die WWW-Seite zur Vorlesung zu erhalten):

Kongreßmänner (voter.data)

Quelle: Congressional Quarterly Almanac, 98th Congress, 2nd session 1984, Volume XL: Congressional Quarterly Inc. Washington, D.C., 1985.

Beschreibung: Die Datei enthält Stimmen der Kongreßmänner des U. S. House of Representatives für die 16 wichtigsten Abstimmungen.

Die Originalversion der Datei wurde dahingehend geändert, daß Observationen mit partiell unbekannten Attributen entfernt und Inkonsistenzen entfernt wurden.

Anzahl der Attribute: 17

Alle 17 Attribute sind binär. Die Bedeutungen der einzelnen Attribute sind wie folgt:

1. Partei: democrat, republican
2. Stimme für „handicapped-infants“ (y,n)
3. Stimme für „water-project-cost-sharing“ (y,n)
4. Stimme für „adoption-of-the-budget-resolution“ (y,n)
5. Stimme für „physician-fee-freeze“ (y,n)
6. Stimme für „el-salvador-aid“ (y,n)
7. Stimme für „religious-groups-in-schools“ (y,n)
8. Stimme für „anti-satellite-test-ban“ (y,n)
9. Stimme für „aid-to-nicaraguan-contras“ (y,n)
10. Stimme für „mx-missile“ (y/n)
11. Stimme für „immigration“ (y,n)
12. Stimme für „synfuels-corporation-cutback“ (y,n)
13. Stimme für „education-spending“ (y,n)
14. Stimme für „superfund-right-to-sue“ (y,n)
15. Stimme für „crime“ (y,n)
16. Stimme für „duty-free-exports“ (y,n)
17. Stimme für „export-administration-act-south-africa“ (y/n)

Konkrete Aufgabe: Aus den Attributen 2-17, d.h. den Stimmwerten, soll durch LAD die Parteizugehörigkeit geschlossen werden.

Brustkrebs (cancer.data)

Quelle: Dr. William H. Wolberg, General Surgery Dept., University of Wisconsin, Clinical Sciences Center, Madison, WI 53792

Beschreibung: Die Datei enthält die Diagnose von Brustkrebspatienten nach „benign“ (gutartig) and „malignant“ (bösartig).

Aus der Originaldatei wurde die ID (Patientennummer) entfernt.

Anzahl der Attribute: 31 Im einzelnen besitzen die Attribute folgende Bedeutung:

1. Diagnose: B=benign, M=malignant

2.-31 Numerische Werte, die sich aus der Auswertung von Gewebeproben ergeben haben.

Konkrete Aufgabe: Aus den Attributen 2-31 sollen durch LAD die Diagnose geschlossen werden.

Messwerte in der Ionosphäre (ionosphere.data)

Quelle: Space Physics Group, Applied Physics Laboratory Johns Hopkins University, Johns Hopkins Road Laurel, MD 20723.

Beschreibung: Die Datei enthält die Klassifizierung von freien Elektronen in der Ionosphäre mit Hilfe von Radar-Daten nach „good“ (gut) und „bad“ (schlecht). Die Daten wurden von einem System in der Goose Bay, Labrador, aufgenommen. Gute Radar-Werte sind solche, deren Rückgabewerte auf das Vorhandensein einer Struktur in der Ionosphäre schließen lassen. Schlechte Radar-Werte zeigen keine Struktur an, ihre Strahlen gehen dann durch die Ionosphäre durch.

Die Originaldatei wurde dahingehend abgeändert, daß das zu erschließende Attribut (die Klassifizierung) als erstes Attribut aufgeführt wird.

Anzahl der Attribute: 35

1. Die Klassifizierung: g=good, b=bad

2.-35. Numerische Werte aus den Radar-Daten.

Konkrete Aufgabe: Aus den Attributen 2-35 soll durch LAD geschlossen werden, ob es sich um eine gute oder schlechte Radarmessung handelt.

Literatur

[1] S. Arora and M. Sudan, *Improved low-degree testing and its applications*, in Proceedings of the 29th Annual ACM Symposium on the Theory of Computing (STOC'97), 1997, pp. 485–496.

[2] E. Boros, P. L. Hammer, and T. Ibaraki, *Logical analysis of numerical data*, Mathematical Programming, 79 (1997), pp. 163–190.

[3] E. Boros, P. L. Hammer, T. Ibaraki, A. Kogan, E. Mayoraz, and I. Muchnik, *An implementation of logical analysis of data*, Tech. Rep. RRR 22–96, The State University of New Jersey Rutgers, July 1996.

[4] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*, MIT Press, 1990.

[5] U. Feige, *A threshold of* $\ln n$ *for approximating set cover*, in Proceedings of the 28th Annual ACM Symposium on the Theory of Computing (STOC'96), 1996, pp. 314–318.

[6] R. Fletcher, *Practical Methods of Optimization*, John Wiley & Sons, 2 ed., 1987.

[7] D. G. Luenberger, *Linear and Nonlinear Programming*, Addison-Wesley Publishing Company, Reading, Massachusetts, 1984.