Bauhaus-Universität Weimar

Fakultät Medien

Studiengang Mediensysteme

# DataMiningCup 2008: Paarweise Klassifikationsverfahren für Mehrklassenprobleme

Bachelorarbeit

David Wiesner Matrikelnummer 41178
geb. am 10. April 1983

1. Gutachter: Prof. Dr. Benno Maria Stein

Datum der Abgabe: 16. Dezember 2009

## Erklärung der Selbstständigkeit

Ich versichere, dass ich die Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen worden ist. Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

Weimar, 16. Dezember 2008 David Wiesner

## Abstract

Diese Arbeit beschäftigt sich mit dem Problem paarweiser Klassifikationsverfahren für Mehrklassenprobleme. Anlaß war der DataMiningCup 2008 - ein Wettbewerb mit Umfragedaten der Süddeutschen Klassenlotterie. Die Arbeit gibt einen allgemeinen Überblick über die wichtigsten Grundlagen zum Datamining und zeigt anhand der Wettbewerbsteilnahme verschiedenen Methoden, ihre Vorteile und Einschränkungen auf. Es wurde versucht durch Experimente die optimalen Methoden zur Dimensionsreduktion und für die Erstellung der Klassifizierer zu finden. Ein besonderer Schwerpunkt lag auf unterschiedlichen paarweisen Verfahren für die Erstellung geeigneter Klassifizierer. In Experimenten wurde versucht, eine geeignete Kombination von Dimensionsreduktionsverfahren und Klassifizierern zu finden. Nach der Vorstellung und Analyse der Wettbewerbsergebnisse werden mögliche Optimierungsoptionen aufgezeigt.

# Inhaltsverzeichnis


1 Einleitung 1
- 1.1 Datamining . . . 1
- 1.2 Der DataMiningCup 2008 . . . 2

2 Attributauswahl und Dimensionsreduktion 4
- 2.1 Information Gain . . . 5
- 2.2 Hauptkomponentenanalyse . . . 7
- 2.3 Schrittweise Diskriminanzanalyse . . . 8
- 2.4 Lineare SVM . . . 10
- 2.5 Autoencoder . . . 14

3 Mehrklassenklassifikation 17
- 3.1 Diskriminanzanalyse . . . 17
- 3.2 Support Vector Machine . . . 18
- 3.3 Paarweise Klassifizierung . . . 22
  - 3.3.1 Einer gegen den Rest . . . 22
  - 3.3.2 Einer gegen Einen . . . 22
  - 3.3.3 Tournament . . . 23

4 Experimente 26
- 4.1 Datenanalyse . . . 26
- 4.2 Aufbau der Experimente . . . 28
- 4.3 Dimensionsreduktion . . . 31
- 4.4 Mehrklassen-Klassifikation . . . 35

5 Fazit und Ausblick 38
- 5.1 Ergebnisse des DataMiningCup 2008 . . . 39

Literaturverzeichnis 43

A Grafiken 45

# KAPITEL 1

## Einleitung

Daten werden gesammelt, getauscht und für viel Geld verkauft. Um die Masse einzelner Informationen gewinnbringend nutzen zu können, müssen die Besitzer lernen, daraus zu lesen. Die Daten sind nur soviel Wert wie die Analyse, die sie interpretiert. Die Methode, die aus Alter und Internetnutzung potentielle Werbekunden herausfiltern soll, heißt Datamining. Die Süddeutsche Klassenlotterie (SKL) hat eine Umfrage unter ihren Kunden durchgeführt und stellte die Daten 2008 einem Wettbewerb zur Verfügung, der seine Teilnehmer nach der besten Auswertungsmethode suchen ließ. Die Teilnahme am Wettbewerb und die theoretischen Grundlagen bilden den Ausgangspunkt dieser Arbeit. Im ersten Kapitel werde ich zunächst allgemein auf das Thema Datamining eingehen und die Fragen klären, was Datamining ist und welche Schritte für einen effizienten Ablauf beachtet werden müssen. Im Weiteren werde ich die Aufgabenstellung des Wettbewerbes vorstellen und auf einige Problembereiche hinweisen. In den Kapiteln zwei und drei werde ich auf die theoretischen Hintergründe der wichtigsten von mir verwendeten Methoden eingehen; den Dimensionsreduktionsverfahren und einigen Klassifizierungsmethoden. Der Experimentaufbau des Wettbewerbsprojektes und die Experimente mit dem Datensatz bilden das vierte Kapitel. Am Schluss dieser Arbeit werden die Ergebnisse des Wettbewerbes vorgestellt und einige Optimierungsmöglichkeiten aufgezeigt.

### 1.1 Datamining

Datamining bezeichnet ein Verfahren, bei dem mit mathematischen Methoden nützliche und aussagekräftige “Patterns” aus Datensammlungen gewonnen werden. Patterns sind Aussagen oder Modelle, die bisher unbekannte Zusammenhänge und Strukturen beschreiben. Der Datamining-Prozess basiert auf mehreren aufeinander aufbaueneden Schritten, in denen ausgehend von Rohdaten aussagekräftige Patterns gefunden werden sollen. Bevor der iterative Prozess starten kann, muss zuvor das Ziel des Dataminings festgelegt werden. Es muss genau bestimmt werden, welche

Aufgaben das Pattern später lösen oder welche Aussagen es treffen können soll. Erst dann kann das eigentliche Datamining beginnen, indem eine Datenselektion anhand der festgelegten Aufgabe durchgeführt wird. Dies kann zum Beispiel eine Auswahl bestimmter Datensätze oder Attribute sein. Im zweiten Schritt folgt die verlustfreie Datenaufbereitung. Diese umfasst das Transformieren der Attributwerte in bestimmte Skalenniveaus, das Festlegen einer Strategie für fehlende Datenwerte sowie das Entfernen von Rauschen in den Daten. Als Nächstes erfolgt die verlustbehaftete Datenreduktion und -transformation, die für das Herausfinden von nützlichen Attributen und Attributkombinationen zuständig sind. Die Dimensionsreduktion begrenzt die Anzahl der Attribute und ermöglicht das Finden von Invarianzen in den vorliegenden Daten. Die Patterns werden schließlich im vierten Schritt, der Mustererkennung, ermittelt. Dies umfasst zum Beispiel das Erstellen von Klassifizierern, das Clustering und die Regression. Im letzten Schritt werden die gewonnenen Patterns interpretiert und evaluiert. Dabei ist es entscheidend, die Patterns auszuwählen, die die konkrete Aufgabenstellung am besten erfüllen. Es kann beispielsweise wichtig sein, die Strukturen zu verstehen und diese nachvollziehbar darzustellen. Für andere Zielsetzungen werden dagegen die Patterns eher für eine Vorhersage auf neuen Datensätze benötigt.[Fay96]

In meinem Fall sind die Daten aus einer Umfrage mit 113 476 Benutzern entstanden. Im Rahmen meiner Arbeit werde ich diese Patterns verwenden, um neue Umfragedaten in einen Zusammenhang mit den bekannten Datensätzen zu bringen. Ich benötigte Patterns, die Daten in bestimmte Klassen einordnen können. Diese Patterns werden als Klassifizierer bezeichnet. Eine Klasseneinteilung ist in den Daten meist als Attribut vorhanden. Dieses Attribut wird im Folgenden als Zielattribut bezeichnet. Die von mir verwendeten Klassifizierer erlauben nur diskrete Klassen. Besteht das Zielattribut aus kontinuierlichen Werten, müssen diese vorher diskretisiert werden.

## 1.2 Der DataMiningCup 2008

Der DataMiningCup (DMC) ist ein Wettbewerb, der seit neun Jahren jährlich veranstaltet wird. An diesem internationalen Wettbewerb können nur Studierende und Auszubildende teilnehmen.
Bei dieser Art der Lotterie ist der Spielzeitraum in sogenannte Klassen unterteilt. Diese Klassen werden im Folgenden “Lotterieklassen” genannt, um Verwechslungen zu vermeiden. In der SKL beträgt der Spielzeitraum einer Lotterie sechs Monate, wobei jeder Monat einer Lotterieklasse entspricht. Ein neue Lotterie beginnt immer am ersten Juni und ersten Dezember eines Jahres. Für jede gespielte Lotterieklasse muss der Spieler ein Los erwerben. Eine Lotterieklasse kann jedoch nur gespielt

werden, wenn auch die vorherige Lotterieklasse derselben Lotterie gespielt wurde. Ein Einstieg in eine laufende Lotterie ist demzufolge nicht möglich.
Die Spieldauer eines Teilnehmers ist das Zielattribut des DMC2008. Das Zielattribut wird im Kontext des Dataminings ebenfalls als "Klasse" und in dieser Arbeit im Folgenden als "Zielklasse" bezeichnet. Die Zielklasse ist ordinalskaliert und besitzt die folgenden fünf Ausprägungen:

0. Es wurde kein Los für die erste Klasse erworben.
1. Nur das Los für die erste Klasse wurde bezahlt.
2. Nur die ersten beiden Klassen wurden gespielt.
3. Alle Klassen der Lotterie wurden gespielt, aber es wurde kein Los für die folgende Lotterie erworben.
4. Auch die erste Klasse der folgenden Lotterie wurde gespielt.

Die Aufgabe des Wettbewerbes war, die Spieldauer der Lotterieteilnehmer so genau wie möglich vorherzusagen. Die Genauigkeit der Vorhersage wurde über eine vorgegebene Kostenmatrix bestimmt. Auf diese Matrix werde ich im Abschnitt 4.1 eingehen. Neben den Trainingsdaten gab es einen Testdatensatz, bei dem die Zielklasse fehlte. Beide Datensätze enthielten jeweils 113 476 Umfragedaten. Die Aufgaben des Wettbewerbes wurden am 1. April 2008 bekannt gegeben. Bis zum 15. Mai 2008 mussten die Ergebnisse eingereicht werden. Dafür musste die Zielklasse für jeden Datensatz aus der Testmenge bestimmt werden.

Meine Aufgabe bestand nun darin, einen Klassifizierer $c$ mit den Trainingsdaten $D$ zu trainieren. Dabei wurde jeder Datensatz $d_i$ als Tupel dargestellt $d_i = (\mathbf{x_i}, y_i)$, wobei $y_i$ die Zielklasse und der Vektor $\mathbf{x_i}$ die Werte aller anderen Attribute enthält. Dieser Vektor wird als Merkmalsvektor bezeichnet; die Menge aller Vektoren $X$ als Merkmalsraum. Die Menge aller möglichen Ausprägungen der Zielklasse nenne ich $Y$.

$$\begin{aligned} \mathbf{x} &\in X \\ Y &= \{1, 2, 3, 4, 5\} \\ y &\in Y \\ c &: X \to Y \end{aligned}$$

# KAPITEL 2

## Attributauswahl und Dimensionsreduktion

Eine Möglichkeit, die Leistungsfähigkeit eines Klassifizierer zu erhöhen, ist die Dimension des Merkmalsraumes $X$ zu verringern. Es gibt Attribute, deren Ausprägungen keinen Rückschluss auf die Klassenzugehörigkeit zulassen. Diese „nutzlosen" Attribute können jedoch in Kombination mit anderen Attributen separierend wirken. Das Entfernen oder Kombinieren von Attributen wird durch eine Transformation des Merkmalsraumes erreicht. Den so entstandenen Raum werde ich als Featureraum und die transformierten Attribute als Features bezeichnen, um sie von den gegebenen Attributen aus den Umfragedaten abzugrenzen.

Ein Spezialfall der Feature-Erzeugung ist die Attributauswahl. Die Features bestehen in diesem Fall nur aus einer Teilmenge der Attribute. Eine reine Attributauswahl erleichtert besonders die Wissensgenerierung, den letzten Schritt des Datamining-Prozesses. Klassenbezogene Featureausprägungen ermöglichen eine direkte Interpretation in Bezug auf die Ausprägungen des korrespondierenden Attributes. Zum Beispiel wird dadurch eine Verbindung (oder Gegenüberstellung) der Spieldauer aller Teilnehmer mit ihrem Alter einfacher. Ein Ergebnis könnte wie folgt aussehen: „90% der Teilnehmer über 67 spielen alle sechs Lotterieklassen".

$$X' = \iota(X), \iota : \mathbb{R}^p \to \mathbb{R}^q, q < p \qquad \text{Feature-Erzeugung} \qquad (2.1)$$

$$X' \subseteq X \qquad \text{Attributauswahl} \qquad (2.2)$$

Ein Ziel der Dimensionsreduktion ist die Steigerung der Erzeugungsgeschwindigkeit von Klassifizierern [Guy03]. Eine Dimensionsreduktion muss nicht unbedingt zu einem schlechteren Ergebnis führen. Attribute, die allein betrachtet redundant erscheinen, können in Konjunktionen oder Produkten mit anderen Attributen nützliche Features erzeugen. Dies bedingt zunächst jedoch eine Dimensionserhöhung. Eine solche Dimensionserhöhung wird später auch beim Autoencoder verwendet. Die verschiedenen Verfahren zur Attributauswahl können in drei Gruppen unterteilt werden:

Filter benutzt vom Klassifizierer unabhängige Sortier- und Auswahlverfahren für die Attribute.

Wrapper benutzt Klassifizierer, um anhand der Vorhersagekraft von einzelnen Attributen oder Attributkombination eine Auswahl von Attributen zu treffen.

Embedded nimmt die Attributauswahl während des Trainings eines Klassifizierers vor.

Bei der Attributauswahl mit Hilfe von Wrappern gibt es verschiedene Methoden. Die einfachste ist, für jedes Attribut einen Klassifizierer zu trainieren. Anhand eines Gütemaßes, das die Genauigkeit der Trennung eines Klassifizierers beurteilt, können dann die Attribute sortiert werden. Aus dieser sortierten Liste kann eine Auswahl der Attribute erfolgen. Dieses Verfahren geht jedoch davon aus, dass die Attribute unabhängig voneinander sind. Attributkombinationen, die eine bessere Trennbarkeit ermöglichen könnten, werden somit nicht berücksichtigt. Des Weiteren steigert ein solches Verfahren das Problem der Überanpassung (dem *overfitting*) [Reu03]. Es gibt zwei weitere Verfahren, die Attributkombinationen in der Attributauswahl berücksichtigen und robust gegen eine Überanpassung sind. Bei der *Forwardselection* werden nach und nach größere Teilmengen von Attributen ausgewählt und der Klassifizierer mit dieser Auswahl trainiert. Bei dem zweiten Verfahren, der *Backwardelimination*, wird der Klassifizierer am Anfang mit der gesamten Attributmenge trainiert, um dann Schritt für Schritt Attribute aus der Menge zu entfernen.

Es ist dabei zu beachten, dass die einzelnen Methoden zur Dimensionsreduktion unterschiedliche Anforderung an die Attribute stellen. Einige Methoden fordern ein bestimmtes Skalenniveau der Variablen, andere hingegen verlangen sogar eine bestimmte Verteilung der Attributausprägungen. Diese Anforderungen werden bei der Beschreibung der verwendeten Methoden in den folgenden Abschnitten genauer erklärt.

## 2.1 Information Gain

Das in deutschen Publikation meist als Informationsgewinn bezeichnete Information Gain ist ein Kriterium für die Attributauswahl. Ein Attribut $A$ besitzt mehrere Ausprägungen. Diese Ausprägungen werden benutzt, um eine Trainingsmenge $D$ in mehrere disjunkte Teilmengen aufzuteilen. Besitzt ein Attribut $n$ diskrete Ausprägungen, werden durch die Attributauswahl entsprechend $n$ Teilmengen gebildet. Ist der Wertebereich eines Attributes kontinuierlich, muss dieser zuerst in den diskreten Wertebereich überführt werden. In der Abbildung 2.1 ist diese Teilung bei zwei verschiedenen Attribute dargestellt.

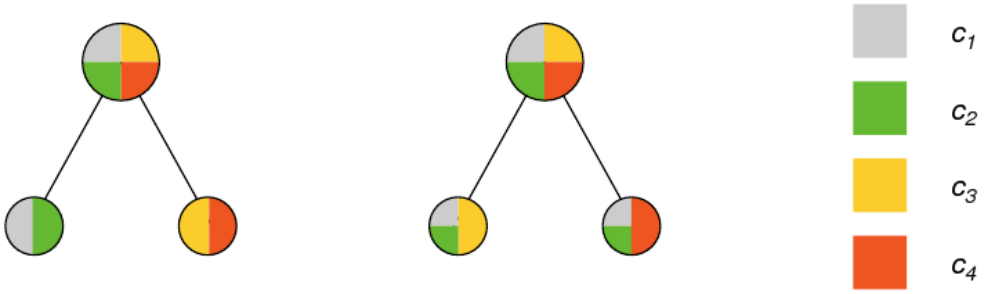



**Abbildung 2.1:** Unreinheit(Impurity) einer Trennung durch Attributauswahl (Quelle: [Ste08, Unit. Splitting])

In diesem Beispiel wird eine Menge, die vier Zielklassen $c_1$, $c_2$, $c_3$ und $c_4$ enthält, in jeweils zwei Teilmengen unterteilt. Um die Güte einer Trennung bestimmen zu können, wird der Begriff der Reinheit (Purity) bzw. Unreinheit (Impurity) benötigt. Eine Möglichkeit, die Impurity einer Menge zu bestimmen, ist der Informationsgehalt (Entropie)[Ste08, Unit. Splitting].

$$D_y = \{(\mathbf{x}, c(\mathbf{x})) \in D : c(\mathbf{x}) = y\} \quad \text{Menge aller Datensätze der Zielklasse y} \tag{2.3}$$

$$P(D_y) = \frac{|D_y|}{|D|} \tag{2.4}$$

$$H(D) = -\sum_{y \in Y} P(D_y) \log_2 (P(D_y)) \quad \text{Entropie} \tag{2.5}$$

Um nun die Güte der Trennung durch ein Attribut $A$ zu messen, das die Trainingsmenge $D$ in $n$ disjunkte Teilmengen unterteilt, muss die gewichtete Impurity aller Teilmengen bestimmt werden. Dazu benutzt man die bedingte Entropie.

$$H(D|A) = \sum_{i=1}^{n} P(A_i) \cdot H(D|A_i) \quad \text{bedingte Entropie} \tag{2.6}$$

$$InfoGain(D, A) = H(D) - H(D|A) \quad \text{Informationsgewinn} \tag{2.7}$$

Der Informationsgewinn ergibt sich aus der Differenz des Informationsgehaltes der Trainingsmenge und des bedingten Informationsgehaltes der Teilmengen. Zur Berechnung der Attributauswahl muss der Informationsgewinn aller Attribute berechnet werden. Ausgewählt werden dann die Attribute mit dem größten Gewinn.

## 2.2 Hauptkomponentenanalyse

Die Hauptkomponentenanalyse (Principal Component Analysis, PCA) ist eine der bekanntesten Methoden zur Dimensionsreduktion. Dabei erzeugt die Hauptkomponentenanalyse einen Featureraum $X' \in \mathbb{R}^q$ durch eine lineare Transformation des Merkmalsraumes $X \in \mathbb{R}^p$. Die PCA nimmt an, dass die relevanten Informationen des Merkmalsraumes in den Richtungen enthalten sind, wo die Daten die größte Varianz besitzen. Diese Richtungen werden als Basisvektoren für den Featureraum genutzt. Dazu werden die Basisvektoren von der PCA in geordneter Form erzeugt. Der erste Basisvektor wird so erstellt, dass er die größte Varianz in den Daten erklärt. Der zweite Basisvektor wird im orthegonalen Unterraum bezüglich des ersten Basisvektors gesucht, steht also senkrecht auf diesem. Auch dieser wird so erzeugt, dass er in die Richtung der verbliebenen maximalen Varianz zeigt. Diese Prozedur wird für alle $q$ Basisvektoren durchgeführt. Diese Basisvektoren spannen dann einen $q$-dimensionalen Unterraum auf, der die größte Varianz des Merkmalsraumes hat [Sch]. Die Basisvektoren des so entstandenen Featureraumes $X'$ werden als Hauptkomponenten bezeichnet. Die Hauptkomponentenanalyse ist eine lineare Transformation des Merkmalsraumes und bewirkt daher keine Dimensionsreduktion bis nicht eine Begrenzung der Basisvektoren $q : q < p$ stattgefunden hat.

Der folgende Abschnitt wird auf die mathematische Funktionsweise der Hauptkomponentenanalyse eingehen. Bei diesem Verfahren wird der Trainingsdatensatz als eine Menge $D = \{\mathbf{a_1}, \ldots, \mathbf{a_p}\}$ von $n$-dimensionalen Attributvektoren dargestellt. Der Attributvektor enthält die Attributausprägungen aller $n$ Datensätze für ein Attribut. Der Trainingsdatensatz $D$ kann als eine $(n \times p)$-Matrix dargestellt werden. Da die Hauptkomponentenanalyse eine lineare Transformation $D' = D\Gamma$ ist, muss eine Transformationsmatrix $\Gamma = \{\gamma_\mathbf{1}, \ldots, \gamma_\mathbf{p}\}$ gefunden werden, so dass die Hauptkomponenten $\mathbf{b_i}$ des Featureraumes eine maximale Varianz aufweisen. Das bedeutet, dass die Varianz von $\mathbf{b_i} = \gamma_\mathbf{1}\mathbf{a_1} + \ldots + \gamma_\mathbf{p}\mathbf{a_p} = {\gamma_\mathbf{i}}^T D$ maximal wird. Für die Suche der Hauptkomponenten wird zunächst eine Kovarianzmatrix $C$ des Trainingsdatensatzes

$D$ erstellt.

$$\bar{a} = \frac{1}{n}\sum_{k=1}^{n} a_k \qquad \text{arithmetisches Mittel von } \mathbf{a} \tag{2.8}$$

$$Var(\mathbf{a}) = \frac{1}{n-1}\sum_{k=1}^{n}(a_k - \bar{a})^2 = \frac{1}{n-1}(\mathbf{a} - \bar{a})^T(\mathbf{a} - \bar{a}) \qquad \text{Varianz von } \mathbf{a} \tag{2.9}$$

$$Cov(\mathbf{a}, \mathbf{b}) = \frac{1}{n-1}(\mathbf{a} - \bar{a})^T\left(\mathbf{b} - \bar{b}\right) \qquad \text{Kovarianz von } \mathbf{a} \text{ und } \mathbf{b} \tag{2.10}$$

$$C(D) = \left(Cov(\mathbf{a_i}, \mathbf{a_j})\right)_{i,j}, i, j = 1, \ldots, p \tag{2.11}$$

Mit Hilfe dieser Kovarianzmatrix lässt sich die Maximierung von $Var\left({\gamma_\mathbf{i}}^T D\right)$ in ein Eigenwertproblem überführen [Jol02]. Durch die Lösung der Gleichung

$$C\gamma_\mathbf{i} - \lambda_i\gamma_\mathbf{i} = 0 \forall i \in \{1, \ldots, p\}$$

lassen sich die Eigenvektoren bestimmen, die den Hauptkomponenten entsprechen. Die Eigenwerte $\lambda_i$ sind die Varianz der Hauptkomponenten. Anhand der Eigenwerte können nun auch die erklärte Varianz $Var_q$ und den Rekonstruktionsfehler bei der Begrenzung des Featureraumes auf die ersten $q : q < p$ Hauptkomponenten berechnet werden.

$$Var_q = \sum_{i=1}^{q}\lambda_i \tag{2.12}$$

$$\frac{Var_q}{Var_p} = \frac{\sum_{i=1}^{q}\lambda_i}{\sum_{i=1}^{p}\lambda_i} \qquad \text{Anteil der erklärten Varianz} \tag{2.13}$$

Wird der Featureraum auf $q$ Hauptkomponenten begrenzt, so ergibt sich folgender Rekonstruktionsfehler

$$Err_q = \frac{\sum_{i=q+1}^{p}\lambda_i}{\sum_{i=1}^{p}\lambda_i}$$

## 2.3 Schrittweise Diskriminanzanalyse

Die Diskriminanzanalyse benutzt wie auch die Hauptkomponentenanalyse die Varianz als Optimierungskriterium. Zudem werden auch die Zielklassen in die Berechnung mit einbezogen. Dabei wird diese durch einen Klassenzentroiden (2.14) beschrieben. Bei der linearen Diskriminanzanalyse wird eine lineare Diskriminanzfunktion $r(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0$ gesucht, die eine maximale Trennung der Zielklassen

ermöglicht. Für eine maximale Trennung müssen zwei Varianzkriterien optimiert werden. Das erste zu maximierende Kriterium ist die Varianz zwischen den Zielklassen (2.15), die *Intervarianz*. Dies entspricht der Maximierung der Abstände zwischen den Klassenzentroiden. Als zweites gilt es die Varianz innerhalb einer Zielklasse, die Intravarianz, zu minimieren (2.16).

$$\bar{\mathbf{z}}_i = \frac{1}{|\{\mathbf{x} \in X : c(\mathbf{x}) = i\}|} \sum_{\mathbf{x_k} \in X : c(\mathbf{x}) = i} \mathbf{x}_k \qquad \text{Klassenzentroid der } i\text{-ten Klasse} \tag{2.14}$$

$$S_b = \sum_{i=1}^{|Y|} |\{\mathbf{x} \in X : c(\mathbf{x}) = i\}| \, (\bar{\mathbf{z}}_i - \bar{\mathbf{z}})^2 \qquad \text{Intervarianz} \tag{2.15}$$

$$S_w = \sum_{i=1}^{|Y|} \sum_{\mathbf{x_k} \in X : c(\mathbf{x}) = i} (\mathbf{x_k} - \bar{\mathbf{z}}_i)^2 \qquad \text{Intravarianz} \tag{2.16}$$

$$\gamma = \frac{S_b}{S_w} \qquad \text{zu minimierendes Diskriminanzkriterium} \tag{2.17}$$

Der Quotient $\gamma$ dieser beiden Kriterien wird als Diskriminanzkriterium zusammengefasst, das es zu minimieren gilt. Dieses Optimierungsproblem lässt sich ebenfalls in ein Eigenwertproblem überführen, wie schon zuvor bei der Hauptkomponentenanalyse (siehe Abschnitt 2.2 dieser Arbeit). Auf die Überführung der Diskriminanzanalyse in ein Eigenwertproblem und die Bestimmung der Diskriminanzfunktion aus Eigenvektoren wird gesondert im Abschnitt 3.1 eingegangen.

Für die Attributauswahl mittels Diskriminanzanalyse wird ein Gütemaß benötigt, welches die Trennschärfe der Diskriminanzfunktionen bewertet. Die von mir verwendete Software SPSS arbeitet unter anderem mit dem Gütemaß Wilks' Lambda $\lambda = \frac{1}{1+\gamma}$. Da es sich hierbei um ein inverses Maß handelt, entspricht ein kleiner Wert einer großen Trennschärfe und ein großer Wert einer geringen Trennung.

Die Attributauswahl beginnt mit der Aufnahme des Attributes in die Diskriminanzfunktion, welches laut Gütemaß die beste Trennung der Zielklassen ermöglicht. Im zweiten Schritt wird aus den verbleibenden Attributen jenes ausgewählt, das die zweitbeste Trennung ermöglicht. Mit einem Kriterium kann nun getestet werden, ob die Hinzunahme eines Attributes eine signifikante Verbesserung in der Trennung der Zielklassen ermöglicht und so der Verbleib in der Diskriminanzfunktion gerechtfertigt ist. Dieses Kriterium wird *Erklärungsmaß* genannt. In SPSS stehen zwei Maße aus der F-Statistik zur Wahl: der F-Wert und die Signifikanz von F [Bro98, SPS05]. Ich verwendete die voreingestellten F-Werte. Es ist jeweils ein Aufnahme- und ein

Ausschlusswert definiert, die den Grad der Verbesserung bzw. Verschlechterung der Diskriminanzfunktion bewerten. Wird der Aufnahmewert unterschritten, wird das Attribut verworfen und die Suche nach einem geeigneten zweiten Attribut für die Diskriminanzfunktion fortgesetzt. Dazu werden nach und nach die laut Gütemaß nächstbesten Attribute getestet, bis eines dem Erklärungsmaß genügt und als zweites Attribut in die Funktion aufgenommen wird. Im nächsten Schritt wird geprüft, ob der Wegfall des bisherigen ersten Attributes eine relevante Verschlechterung bewirkt, die unter dem Niveau des Ausschlusswertes liegt. In diesem Fall ist der Verbleib des Attributes in der Funktion nicht mehr gerechtfertigt und es wird entfernt. Liegt der Wert jedoch über dem Ausschlusswert, verbleibt es in der Funktion. Die nächsten Schritte folgen dem gleichen Prinzip. Es wird weiter nach Attributen gesucht, die dem Auswahlkriterium der F-Statistik genügen. Auch wird weiterhin in jedem Schritt geprüft, ob der Verbleib der bisher ausgewählten Attribute noch gerechtfertigt ist. Diese schrittweise Selektion der Attribute ist beendet, wenn kein Attribut mehr existiert, das den Auswahlkriterien genügt oder alle Attribute in die Diskriminanzfunktion aufgenommen wurden.

## 2.4 Lineare SVM

Die Supportvektormaschine (SVM) ist ein Klassifizierer, der in den letzten zehn Jahren immer mehr an Bedeutung gewonnen hat und als sehr leistungsfähig gilt. Zur Dimensionsreduktion werde ich die SVM als Wrapper einsetzen, um anhand einer Forwardselection bzw. Backwardelimination die Anzahl der Attribute zu begrenzen. Das Problem bei der Verwendung von Filtern, wie dem Information Gain, ist, dass sie meist andere Informationskriterien besitzen als die Klassifizierer. Diese sollen anhand der gewählten Attribute ein leistungsfähiges Pattern erzeugen. Wenn man also eine SVM als Klassifizierer verwendet, liegt es nahe, auch eine SVM zur Dimensionsreduktion zu benutzen. Ein Argument, das gegen den Einsatz der SVM spricht, ist jedoch die Komplexität. Filter sind meist deutlich schneller. Um die Komplexität der Dimensionsreduktion zu senken, sollte man für die Attributauswahl einen linearen Klassifizierer verwenden. Zum Erstellen des Modells kann man einen nicht-linearen Klassifizierer einsetzen[Che06, Bi03].

In diesem Abschnitt werde ich mich auf die Erklärung der linearen SVM als binären Klassifizierer $c(\mathbf{x_i}) = y_i : y_i \in Y, |Y| = 2$ in einem separierbaren Fall beschränken. Im nächsten Kapitel werde ich dann auf die nicht-lineare SVM und das Mehrklassenproblem zu sprechen kommen.

Die Abbildung 2.2 zeigt, dass bei einem linear separierbaren Problem viele Hyperebenen $\mathbf{w}^T\mathbf{x}_i + b = 0$ existieren, die die beiden Klassen voneinander trennen. Die SVM versucht für die Trainingsdaten $(\mathbf{x_i}, y_i) \in D, \mathbf{x}_i \in \mathbb{R}^p, y_i \in Y = \{-1, 1\}$ einen

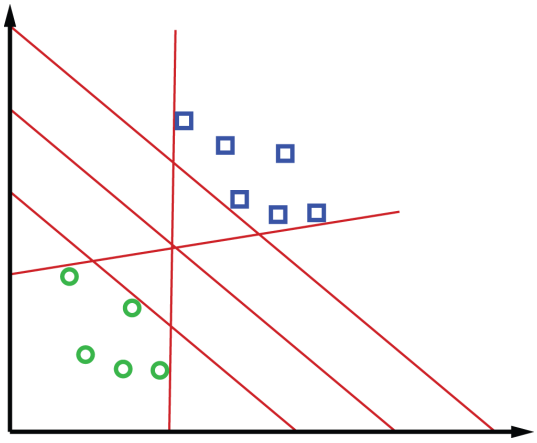


**Abbildung 2.2:** Vielzahl von möglichen Trennebenen

linearen Klassifizierer $\hat{y}_i = c(\mathbf{x}_i) = \mathbf{w}^T\mathbf{x}_i + b, \mathbf{w} \in \mathbb{R}^p, b \in \mathbb{R}$ zu finden, der die größte Separierbarkeit aufweist. Dies stellt sich dabei folgendermaßen dar:

$$y_i = \begin{cases} -1 & \text{if } \hat{y}_i \ll 0 \\ 1 & \text{if } \hat{y}_i \gg 0 \end{cases} \qquad \text{große Separierbarkeit des Klassifizierers} \tag{2.18}$$

$$\hat{\gamma}_i = y_i \left(\mathbf{w}^T\mathbf{x}_i + b\right) \qquad \textit{functional margin} \tag{2.19}$$

Die Separierbarkeit wird durch eine Hyperebene realisiert, die den größten Abstand $|\hat{\gamma}_i| \gg 0$ zu den beiden Klassen besitzt. Die SVM wählt dafür einige Vektoren $\mathbf{x}_i \in \mathbb{R}^p$ aus, die in der Nähe der Hyperebene liegen. Diese Vektoren werden als die *Supportvektoren* bezeichnet. Die Auswahl von nur einigen Merkmalsvektoren verhindert dabei eine Überanpassung (ein Overfitting) des Klassifizierers und reduziert die Komplexität. Die Suche nach dem minimalen rechnerischen Abstand (2.19), dem *functional margin*, wirft ein Problem auf. Um den *functional margin* zu erhöhen, reicht eine Skalierung von $\hat{\gamma}_i = y_i \left(c\mathbf{w}^T\mathbf{x}_i + cb\right)$ mit einer Konstanten $c : c \in \mathbb{R}^p \setminus \{0\}$. Um dies zu verhindern wird $(\mathbf{w}, b) \in \mathbb{R}^p \times \mathbb{R}$ relativ zu den Supportvektoren $\mathbf{x}_1, \ldots, \mathbf{x}_m$ folgendermaßen skaliert:

$$\min_{i=1,\ldots,m} \left|\mathbf{w}^T\mathbf{x}_i + b\right| = 1 \tag{2.20}$$

Diese skalierte Hyperebene wird als kanonische Hyperebene bezeichnet. Ist das Kriterium (2.20) erfüllt, hat der am dichtesten an der Hyperebene liegende Punkt den Abstand $\frac{1}{\|\mathbf{w}\|}$. Um dieses Problem zu lösen betrachten wir die geometrische Definition des Abstandes, dem *gemetric margin*.

Ist $\mathbf{x}_1$ ein Supportvektor mit der Klasse $y_1 = 1$ mit dem Abstand $\gamma_1$ zu der

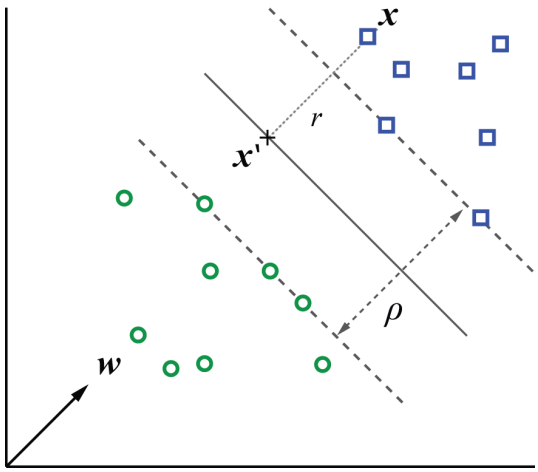



**Abbildung 2.3:** geometrische Abstand $\gamma$ eines Punktes $\mathbf{x}$ zur Hyperebene

Hyperebene $\mathbf{w}^T\mathbf{x}_i + b = 0$, so muss folgendes erfüllt sein [Ng]:

$$\mathbf{w}^T\left(\mathbf{x}_1 - \gamma_1\frac{\mathbf{w}}{||\mathbf{w}||}\right) + b = 0 \tag{2.21}$$

Umgestellt nach dem Abstand $\gamma_1$ ergibt sich:

$$\gamma_1 = \frac{\mathbf{w}^T\mathbf{x}_1 + b}{||\mathbf{w}||} = \left(\frac{\mathbf{w}}{||\mathbf{w}||}\right)^T\mathbf{x}_1 + \frac{b}{||\mathbf{w}||} \tag{2.22}$$

Dies gilt für alle Datensätze $\mathbf{x}_i : y_i = 1$. Für die Datensätze $\mathbf{x}_i : y_i = -1$ ist der Abstand aus (2.22) negativ. Um dies zu verhindern erweitert man die Abstandsformel um den Faktor $y_i$ und erhält eine allgemeine Formel für den geometrischen Abstand:

$$\gamma_i = y_i\left(\frac{\mathbf{w}}{||\mathbf{w}||}\right)^T\mathbf{x}_i + \frac{b}{||\mathbf{w}||} \tag{2.23}$$

$$\gamma_i = \frac{\hat{\gamma}_i}{||\mathbf{w}||} \qquad \text{Vgl. (2.19)} \tag{2.24}$$

Ist $||\mathbf{w}|| = 1$, entspricht der geometrische Abstand $\gamma_i$ genau dem *functional margin* $\hat{\gamma}_i$ (2.19). Durch die Normalisierung ist der geometrische Abstand auch robust gegen eine Skalierung.

Nun kann man versuchen, ein Optimierungsproblem für die SVM zu formalisieren.

$$\underset{\mathbf{w}\in\mathbb{R}^p, b\in\mathbb{R}}{\text{maximiere}} \quad \frac{\hat{\gamma}_i}{||\mathbf{w}||} \overset{(2.20)}{=} \underset{\mathbf{w}\in\mathbb{R}^p}{\text{maximiere}} \quad \frac{1}{||\mathbf{w}||} \Rightarrow \underset{\mathbf{w}\in\mathbb{R}^p}{\text{minimiere}} \quad \frac{1}{2}||\mathbf{w}||^2 \tag{2.25}$$

$$\text{Nebenbedingung:} \quad y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) \geq 1, i = 1, \ldots, m \tag{2.26}$$

Nun gibt es ein konvexes quadratisches Optimierungsproblem mit linearen Nebenbedingungen. Dies wird als primales Problem bezeichnet. Mit Hilfe der Lagrange-

Multiplikatoren kann es in ein duales Problem überführt werden. Die Lösungen des dualen sind dabei auch Lösungen des primalen Problems. Dazu wird das Optimierungsproblem mit Nebenbedingungen in ein Optimierungsproblem ohne Nebenbedingungen überführt. Hierfür wird die Lagrange-Funktion genutzt mit den Lagrange-Multiplikatoren $\alpha_i \geq 0$

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2}||\mathbf{w}||^2 - \sum_{i=1}^{m} \alpha_i \left( y_i \left( \mathbf{w}^T \mathbf{x}_i + b \right) - 1 \right) \tag{2.27}$$

Die Lagrange-Funktion $L(\mathbf{w}, b, \alpha)$ muss nun bezüglich $\mathbf{w}$ und $b$ minimiert und bezüglich $\alpha_i$ maximiert werden:

$$\frac{\partial}{\partial b} L(\mathbf{w}, b, \alpha) = 0, \frac{\partial}{\partial \mathbf{w}} L(\mathbf{w}, b, \alpha) = 0 \tag{2.28}$$

Daraus folgt:

$$\sum_{i=1}^{m} \alpha_i y_i = 0 \tag{2.29}$$

$$\text{und } \mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i \tag{2.30}$$

Setzt man diese Lösung wieder in die Lagrange-Funktion ein, erhält man das duale Optimierungsproblem:

$$\underset{\alpha \in \mathbb{R}^m}{\text{maximiere}} \; W(\alpha) = \sum_{i=1}^{m} \alpha_i - \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \tag{2.31}$$

$$\text{Nebenbedingungen: } \alpha_i \geq 0, i = 1, \ldots, m \tag{2.32}$$

$$\text{und } \sum_{i=1}^{m} \alpha_i y_i = 0 \tag{2.33}$$

Die SVM berechnet nun im ersten Schritt die Lagrange-Multiplikatoren $\alpha_i$ durch das Lösen des dualen Problems. Danach wird die Normale $\mathbf{w}$ der Hyperebene mit (2.30) berechnet. Der Abstand $b$ der Hyperebene zum Koordinatenursprung lässt sich nun mit Hilfe der Supportvektoren bestimmen, die am dichtesten zur Hyperebene liegen.

$$b = \frac{min_{i:y_i=1} \mathbf{w}^T \mathbf{x}_i - max_{i:y_i=-1} \mathbf{w}^T \mathbf{x}_i}{2} \tag{2.34}$$

Mit $\mathbf{w}$ und $b$ kann nun die trennende Hyperebene $\mathbf{w}^T \mathbf{x} + b = 0$ beschrieben werden

und man erhält einen binären Klassifizierer $c(\mathbf{x})$:

$$c(\mathbf{x}) = sgn\left(\mathbf{w}^T\mathbf{x} + b\right) = sgn\left(\sum_{i=1}^{m} \alpha_i y_i \mathbf{x}^T \mathbf{x}_i + b\right) \tag{2.35}$$

Für die Attributauswahl benutzt das in Java geschriebene Tool WEKA einen einfachen rekursiven Backwardelimination-Algorithmus [Wit02, page 423]. Der in [Guy02] beschrieben Algorithmus basiert auf der Tatsache, dass die Normale der Hyperebene $\mathbf{w}$ bei einer linearen SVM die Attribute aus den Trainingsdatensätzen $\mathbf{x}$ direkt gewichtet. Die SVM wird am Anfang mit allen Attributen trainiert. Die Elemente des Vektors $\mathbf{w} = \{w_1, \ldots, w_p\}$ entsprechen bei einer linearen SVM den Koeffizienten für die Attribute $\mathbf{a}_k$ des Trainingsdatensatzes $D = \{\mathbf{a}_1, \ldots, \mathbf{a}_p\}$. Mit Hilfe eines Sortierkriteriums, zum Beispiel $r_k = w_k^2, k = 1, \ldots, p$, können die Attribute anhand ihrer Koeffizienten sortiert werden. Dann wird das Attribut $\mathbf{a}_k$ ausgewählt, welches nach dem Sortierkriterium den kleinsten Wert besitzt $\mathbf{a}_k : argmin_{k=1,\ldots,p} r_k$. Jetzt wird die SVM wieder mit dem Trainingsdatensatz ohne das ausgewählte Attribut $\bar{D} = D \setminus \{\mathbf{a}_k\}$ trainiert. Dies wird solange wiederholt, bis beispielsweise nur noch eine festgelegte Anzahl an Attributen den Datensatz beschreibt. Um das Verfahren zu beschleunigen, kann man auch in jedem Schritt mehrere Attribute aus dem Trainingsdatensatz entfernen.

## 2.5 Autoencoder

Der Autoencoder ist ein spezielles neuronales Netz, das eine Dimensionsreduktion durch eine nichtlineare Transformation des Attributraumes erreicht[Hin06]. Ein neuronales Netz besteht aus mehreren Schichten: der Eingabeschicht (*Inputlayer*), einer oder mehreren Zwischenschichten (*Hiddenlayer*) und einer Ausgabeschicht (*Outputlayer*). Die Schichten bestehen aus mehreren Neuronen. Die Neuronen einer Schicht sind über gewichtete Verbindungen $w_i$ mit allen Neuronen $n_i, i = 1, \ldots, m$ der vorherigen Schicht verknüpft. Jedes Neuron besitzt eine Aktivierungsfunktion $g : \mathbb{R} \to [0, 1]$. Die Aktivierung eines Neurons ergibt sich aus $y = g\left(\mathbf{w}^T\mathbf{x}\right)$. Durch ein Lernverfahren, zum Beispiel dem *Backpropagation*, welches später noch erläutert wird, werden die Gewichte der Verbindungen so angepasst, dass die Eingabedaten mit dem geringsten Fehler auf die Ausgabedaten abgebildet werden. Der Autoencoder ist ein symmetrisches Netzwerk und lernt eine identische Abbildung. Dazu entsprechen die Eingänge des neuronales Netzes den Ausgängen. Der Autonecoder besitzt mehrere Hiddenlayer; der Symmetrie wegen eine ungerade Anzahl. Die Anzahl der Neuronen des mittleren Hiddenlayers entspricht der Größe der Dimension des Featureraumes. Man kann also die Abbildung des neuronalen Netzes in zwei Abbildungen aufteilen. In der ersten Abbildung $\Phi_{ih} : X \to X', X \in \mathbb{R}^p, X' \in \mathbb{R}^q, q < p$

wird der Merkmalsraum auf den Featureraum abgebildet. Die zweite Abbildung $\Phi_{ho} : X' \to X, X \in \mathbb{R}^p, X' \in \mathbb{R}^q, q < p$ versucht aus dem Featureraum wieder den Merkmalsraum zu konstruieren. Besitzt der Autoencoder nur eine Zwischenschicht, so ist $\Phi_{ih}$ eine lineare Abbildung, wie sie auch von der Hauptkomponentenanalyse 2.2 gesucht wird. Deshalb wird der Autoencoder als nichtlineare Hauptkomponentenanalyse bezeichnet.

Der Rekonstruktionsfehler der Abbildung $\hat{\mathbf{x}} = \Phi_{ho}(\Phi_{ih}(\mathbf{x}))$ für die Merkmalsvektoren $\mathbf{x} \in \mathbb{R}^p$ eines Trainingsdatensatzes $D$ ist $Err = \frac{1}{2||D||} \sum_{x \in D} \sum_{i=1}^{p} (x_i - \hat{x}_i)^2$. Ziel ist es nun, diesen Fehler zu minimieren. Dazu wird zunächst ein einzelnes Neuron $y(x) = g\left(\mathbf{w}^T\mathbf{x}\right)$ betrachtet, mit dem Fehler $Err(\mathbf{w}) = \frac{1}{2} \sum_{(\mathbf{x},y) \in D} \left(y - g\left(\mathbf{w}^T\mathbf{x}\right)\right)^2$. Für die Optimierung der Gewichte $\mathbf{w}$ wird nun der Gradient der Fehlerfunktion bestimmt.

$$\nabla Err(\mathbf{w}) = \left(\frac{\partial Err}{\partial w_1}, \ldots, \frac{\partial Err}{\partial w_m}\right)^T \tag{2.36}$$

Mit Hilfe dieses Fehlergradienten werden dann die Gewichte angepasst $\mathbf{w} = \mathbf{w} - \eta \nabla Err(\mathbf{w})$. Für jede Komponente des Gewichts gilt daher $w_i = w_i - \eta \frac{\partial Err}{\partial w_i}$ und kann errechnet werden mit:

$$\frac{\partial Err}{\partial w_i} = \frac{\partial Err}{\partial g} \frac{\partial g}{\partial \mathbf{w}^T\mathbf{x}} \frac{\partial \mathbf{w}^T\mathbf{x}}{\partial w_i} = g'\left(\mathbf{w}^T\mathbf{x}\right)\left(y - g\left(\mathbf{w}^T\mathbf{x}\right)\right) x_i \tag{2.37}$$

Der Backpropagation ist ein Algorithmus, der den Fehler an der Ausgabeschicht auf die vorhergehenden Schichten zurück reicht. Dieser Algorithmus besteht aus mehreren Schritten, die für jeden Datensatz $\mathbf{x} \in D$ wiederholt werden. Bevor der Algorithmus zum ersten Mal angewendet wird, werden die Gewichte meist mit Zufallswerten initialisiert. Jetzt wird der erste Datensatz ausgewählt und durchläuft folgende Schritte: Zunächst werden die Neuronen der Eingabeschicht entsprechend den Werten aus dem Datensatz $\mathbf{x}$ aktiviert. Diese Werte werden nun in Abhängigkeit der Verbindungsgewichte und der Aktivierungsfunktion der Neuronen, die sich in den Zwischenschichten befinden, bis in die Ausgabeschicht propagiert. Nun wird der Fehler in der Ausgangsschicht errechnet und zurückpropagiert, um auch die Gewichte in der Zwischenschicht anpassen zu können. Der Fehler in der Zwischenschicht wird anhand der gewichteten Fehler der nachfolgenden Schicht berechnet.

$$\frac{\partial Err}{\partial w_i} = \frac{\partial Err}{\partial g} \frac{\partial g}{\partial \mathbf{w}^T\mathbf{x}} \frac{\partial \mathbf{w}^T\mathbf{x}}{\partial w_i} = g'\left(\mathbf{w}^T\mathbf{x}\right) x_i \sum_k \frac{\partial Err}{\partial w_k} w_k \tag{2.38}$$

Wurde der Fehler wieder bis in die Eingabeschicht zurückpropagiert, werden die Gewichte entsprechend $w_i = w_i - \eta \frac{\partial Err}{\partial w_i}$ angepasst. Mit der Lernrate $\eta$ kann der Einfluss des Fehlers bei der Gewichtsanpassung beeinflusst werden. Eine häufig benutzte Aktivierungsfunktion $g : \mathbb{R} \to [0, 1]$ ist die Sigmoidfunktion. Diese ist leicht zu diffe-

renzieren, was für die Gradienberechnung (2.37) beim Backpropagation-Algorithmus notwendig ist.

$$g\left(\mathbf{w}^T\mathbf{x}\right) = \frac{1}{1 - e^{-\mathbf{w}^T\mathbf{x}}} \tag{2.39}$$

$$g'\left(\mathbf{w}^T\mathbf{x}\right) = g\left(\mathbf{w}^T\mathbf{x}\right)\left(1 - g\left(\mathbf{w}^T\mathbf{x}\right)\right) \tag{2.40}$$

Der Backpropagation-Algorithmus kann solange für einen Trainingsdatensatz $D$ wiederholt werden, bis der Fehler konvergiert oder eine maximale Anzahl an Iterationen überschritten ist. Die gelernte Abbildung $\Phi_{ih} : X \rightarrow X', X \in \mathbb{R}^p, X' \in \mathbb{R}^q, q < p$ kann dann zur Transformation des Merkmalsraumes in den Featureraum benutzt werden.

# KAPITEL 3

## Mehrklassenklassifikation

## 3.1 Diskriminanzanalyse

Die Diskriminanzanalyse ist ein Klassifizierer. Bei einer linearen Diskriminanzanalyse wird eine Diskriminanzfunktion erstellt. Diese wird später dazu genutzt, um neue Datensätze zu klassifizieren und besteht aus den Linearkombinationen der Attribute $r(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0, k = 1, \ldots, |Y| - 1$.

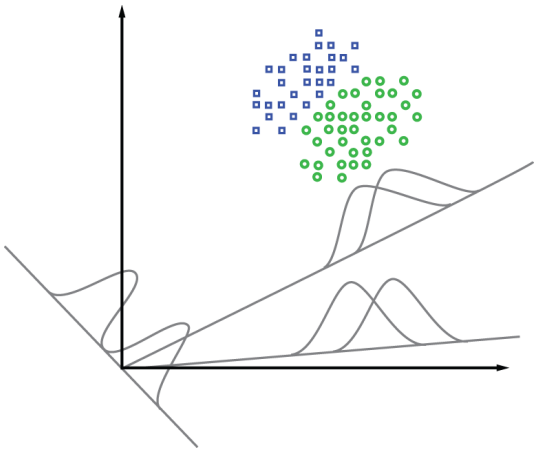


**Abbildung 3.1:** Varianzen in der Diskriminanzfunktion

Die Varianz der Diskriminanzfunktion ergibt sich aus der Summe der Intervarianz(2.15) und der Intravarianz(2.16) $Var(r_k) = S_b + S_w$. Wie auch in der Hauptkomponentenanalyse wird dieses Optimierungsproblem in ein Eigenwertproblem überführt. Dazu wird eine Kovarianzmatrix erstellt. Sie kann zerlegt werden in eine Summe aus der Kovarianzmatrix$C_b$ für die Intervarianz und der Kovarianzmatrix für die Intravarianz$C_w$. Die Intravarianz wird bestimmt für einen Teildatensatz

$D_i : D_i = \{\mathbf{x}_1, \ldots, \mathbf{x}_k\}^T, c(\mathbf{x}_j) = i.$

$$C = C_b + C_w \tag{3.1}$$

$$\bar{\mathbf{z}_\mathbf{i}} = \frac{1}{|D_i|} \sum_{\mathbf{x}_k \in D_i} \mathbf{x}_k \qquad \text{Klassenzentroid der } i\text{-ten Klasse} \tag{3.2}$$

$$C_i = \sum_{\mathbf{x}_k \in D_i} (\mathbf{x}_k - \mathbf{z}_i)(\mathbf{x}_k - \mathbf{z}_i)^T \tag{3.3}$$

$$C_w = \frac{1}{|D|} \sum_{i=1}^{|Y|} |D_i| C_i \tag{3.4}$$

$$C_b = \frac{1}{|D|} \sum_{i=1}^{|Y|} |D_i| (\mathbf{z}_i - \mathbf{z})(\mathbf{z}_i - \mathbf{z})^T \tag{3.5}$$

Die Varianz der Diskriminanzfunktion lässt sich nun folgendermaßen bestimmen: $Var(r) = \mathbf{w}C\mathbf{w}^T = \mathbf{w}C_b\mathbf{w}^T + \mathbf{w}C_w\mathbf{w}^T$. Die Koeffizienten der Diskriminanzfunktion $\mathbf{w}$ müssen nun so gewählt werden, dass $\frac{\mathbf{w}C_b\mathbf{w}^T}{\mathbf{w}C_w\mathbf{w}^T}$ maximal wird. Dies lässt sich als symmetrisches Eigenwertproblem formulieren, das es zu lösen gilt.

$$C_b\mathbf{w} = \lambda C_w\mathbf{w}, \lambda = \frac{\mathbf{w}C_b\mathbf{w}^T}{\mathbf{w}C_w\mathbf{w}^T} \tag{3.6}$$

Die Suche, die das Diskriminanzkriterium maximiert, ist nun die Suche des größten Eigenwertes. Der zum Eigenwert gehörende Eigenvektor ist dann der Gewichtsvektor der Diskriminanzfunktion $r(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0$. Um einen neuen Datensatz zu klassifizieren, wird dieser zunächst mit Hilfe der Diskriminanzfunktion in den Diskriminanzraum transformiert. Um die Klassenzugehörigkeit des neuen Datensatzes zu berechnen, wird nun der Abstand zu allen Klassenzentroiden berechnet. Der Datensatz wird schließlich der Klasse zugeordnet, bei der der Abstand am geringsten ist.

$$d_{L_2}(\mathbf{a}, \mathbf{b}) = \sqrt{\sum_{j=1}^{p} (a_j - b_j)^2} \qquad \text{Euklidischer Abstand} \tag{3.7}$$

$$c(\mathbf{x}) = \underset{i=1,\ldots,|Y|}{\arg\min} d(r(\mathbf{x}), \bar{\mathbf{z}}_i), d : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R} \tag{3.8}$$

Als Abstandsmaß kann man zum Beispiel den Euklidischen Abstand (3.7) wählen.

## 3.2 Support Vector Machine

Im Abschnitt 2.4 wurde die linearen SVM mit einem separierbaren Fall beschrieben. In diesem Abschnitt soll nun zunächst auf die nichtlineare SVM eingegangen werden. Anschließend wird der Umgang mit einem nichtseparierbaren Problem gezeigt. Zum Schluss werde ich mich mit dem Mehrklassenproblem beschäftigen und aufzeigen,

wie dieses in der von mir verwendeten Bibliothek *LibSVM*[1] gelöst wird. Im nächsten Kapitel werden dann weitere Lösungsmöglichkeiten für ein Mehrklassenproblem aufgezeigt.

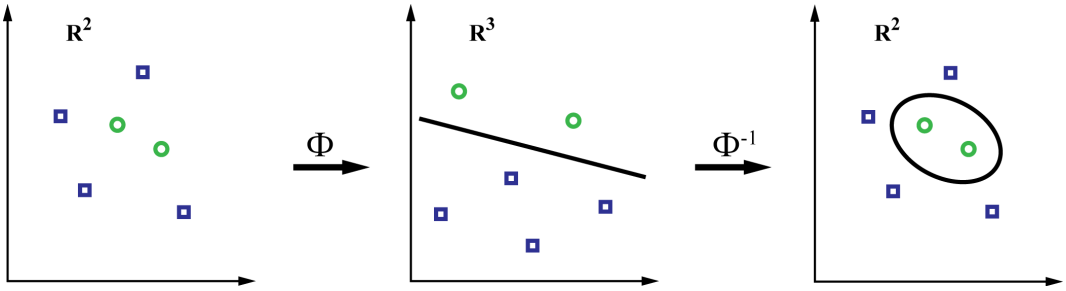



**Abbildung 3.2:** Nichtlineare SVM mit dem Kerneltrick (Quelle: [Sch01])

Um eine bessere Trennung zu erreichen, kann man versuchen, eine nichtlineare Trennebene zu finden. Dazu wird der Merkmalsraum $X \in \mathbb{R}^p$ in einen höherdimensionalen Raum $\mathcal{H} \in \mathbb{R}^s$ überführt. Die dafür verwendete Funktion lautet $\Phi : \mathbb{R}^p \rightarrow \mathbb{R}^s$. In diesem höherdimensionalen Raum wird nun eine Hyperebene gesucht, welche die maximale Trennung (siehe Kapitel 2.4 dieser Arbeit) der Klassen ermöglicht. Diese Hyperebene ist dann im eigentlichen Merkmalsraum eine nichtlineare Trennfläche. Schauen wir uns nun einmal die Veränderung der Entscheidungsfunktion der SVM aus dem Kapitel 2.4 an. Dazu werde ich das Skalarprodukt $\langle ., . \rangle$ nochmal hervorheben.

$$c\left(\mathbf{x}\right) = sgn\left(\sum_{i=1}^{m} \alpha_i y_i \langle \mathbf{x}, \mathbf{x}_i \rangle + b\right) \tag{3.9}$$

Diese wird nun für eine Trennung im höherdimensionalen Raum erweitert.

$$c\left(\mathbf{x}\right) = sgn\left(\sum_{i=1}^{m} \alpha_i y_i \langle \Phi\left(\mathbf{x}_i\right), \Phi\left(\mathbf{x}\right) \rangle + b\right) \tag{3.10}$$

An dieser Formel erkennt man, dass der höherdimensionale Raum nur eine einzige Anforderung erfüllen muss: in $\mathcal{H}$ muss das Skalarprodukt definiert sein. Ist die Dimension von $\mathcal{H}$ sehr groß, ist die Transformation aller Trainingsdatensätze in den höherdimensionalen Raum sehr rechenaufwendig. Zudem wird in diesem höherdimensionalen Raum „nur" ein Skalarprodukt ausgeführt. Um den Rechenaufwand zu verringern, greift man auf einen Trick zurück, den nicht nur die SVM benutzt. Die Idee ist, nicht die Daten in den höherdimensionalen, sondern das Skalarprodukt in den niedrigdimensionalen Raum zu überführen. Die Funktionen, die die Berechnung des Skalarproduktes implizit in einem höherdimensionalen Raum berechnen, heißen

1 http://www.csie.ntu.edu.tw/~cjlin/libsvm/

*Kernels*. Diese Kernel $k(\mathbf{x}_i, \mathbf{x}_j)$ müssen positiv definit und symmetrisch sein.

$$k(\mathbf{x}_i, \mathbf{x}_j) > 0, \quad \forall \mathbf{x}_i, \mathbf{x}_j \in \mathbb{R}^p, \mathbf{x}_i, \mathbf{x}_j \neq \vec{0} \qquad \text{positiv definit} \tag{3.11}$$

$$k(\mathbf{x}_i, \mathbf{x}_j) = k(\mathbf{x}_j, \mathbf{x}_i), \quad \forall \mathbf{x}_i, \mathbf{x}_j \in \mathbb{R}^p \qquad \text{symmetrisch} \tag{3.12}$$

Durch die einfache Ersetzung des Skalarproduktes durch eine Kernelfunktion, können nun auch nichtlineare Trennflächen mit der SVM bestimmt werden. Diese Ersetzung wird Kerneltrick genannt.

$$c(\mathbf{x}) = sgn\left(\sum_{i=1}^{m} \alpha_i y_i k(\mathbf{x}, \mathbf{x}_i) + b\right) \tag{3.13}$$

$$k(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{x}_i, \mathbf{x}_j \rangle \qquad \text{Linear} \tag{3.14}$$

$$k(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{x}_i, \mathbf{x}_j \rangle^d \qquad \text{Polynomiell vom Grad } d \tag{3.15}$$

$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{c}\right) \qquad \textit{radial basis} \tag{3.16}$$

$$k(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\kappa \langle \mathbf{x}_i, \mathbf{x}_j \rangle + c) \quad k > 0, c \in \mathbb{R} \qquad \text{Neuronales Netzwerk} \tag{3.17}$$

Bisher wurde von der Annahme ausgegangen, dass die Daten separierbar sind. In der Praxis wird es jedoch kaum Fälle geben, in denen man eine Hyperebene vorfindet, die alle Datensätze voneinander trennt. Selbst wenn man mit einer nichtlinearen SVM eine solche Hyperebene findet, ist diese meist überangepasst(*overfitted*) und generalisiert schlecht. Schon einzelne Ausreißer können die Hyperebene entscheidend beeinflussen. Um dies zu verhindern, müssen beim Trainieren der SVM einzelne Ausreißer zugelassen werden. Diese Idee wird als *Soft Margin* bezeichnet. Um die Bedingung (2.20) trotzdem zu erfüllen, wird eine sogenannte Schlupfvariable $\xi_i$ eingeführt.

$$\xi_i \geq 0 \quad \forall i = 1, \ldots, m \tag{3.18}$$

$$y_i\left(\mathbf{w}^T \mathbf{x}_i + b\right) \geq 1 - \xi_i, \quad i = 1, \ldots, m \tag{3.19}$$

Die Schlupfvariable kann nun die folgenden Ausprägungen besitzen: Ist die Schlupfvariable $0 < \xi_i < 1$ so liegt der $i$-te Datensatz innerhalb des zu maximierenden Abstandes und der trennenden Hyperebene, jedoch auf der richtigen Seite $c(\mathbf{x}_i) = y_i$ der Hyperebene. Ist $\xi_i = 1$ so liegt $\mathbf{x}_i$ direkt auf der Hyperebene. Für ein $\xi_i > 1$ ist der Datensatz falsch klassifiziert. Die Einführung einer Schlupfvariablen führt zu einem neuen Optimierungsproblem. Neben der Maximierung des Abstandes, also

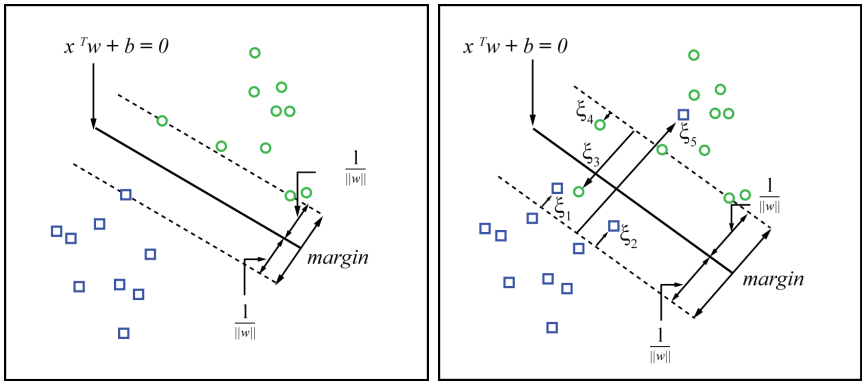



**Abbildung 3.3:** Links: separierbarer Fall, Rechts: Datensatz mit Ausreißern

der Minimierung von $\frac{1}{2}||\mathbf{w}||^2$, muss nun auch die Schlupfvariable minimiert werden.

$$\underset{\mathbf{w}\in\mathbb{R}^p}{\text{minimiere}} \quad \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{m}\xi_i \tag{3.20}$$

$$\text{Nebenbedingung:} \quad y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) \geq 1 - \xi_i, \quad i = 1, \ldots, m \tag{3.21}$$

Die neue Variable $C$ gibt uns die Möglichkeit, den Einfluss der Ausreißer auf das Optimierungsproblem zu bestimmen. Dieser Parameter gestattet es, die Optimierungsprobleme "maximaler Abstand" und "korrekte Klassifizierung aller Datensätze" relativ zueinander zu gewichten. Mit Hilfe der Lagrange-Funktion kann dieses Optimierungsproblem wieder in ein duales Problem überführt werden[Ng].

$$\underset{\alpha\in\mathbb{R}^m}{\text{maximiere}}\ W(\alpha) = \sum_{i=1}^{m}\alpha_i - \sum_{i,j=1}^{m}\alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j \tag{3.22}$$

$$\text{Nebenbedingungen: } 0 \leq \alpha_i \leq C, i = 1, \ldots, m \tag{3.23}$$

$$\text{und} \ \sum_{i=1}^{m}\alpha_i y_i = 0 \tag{3.24}$$

Die einzige Änderung zum dualen Problem ohne Schlupfvariable(2.32) ist die obere Schranke für $\alpha_i$. $C$ beschränkt damit den Einfluss der Datensätze $\mathbf{x}_i$ auf den Gewichtsvektor $\mathbf{w}$. Es gibt verschiedene Ansätze, das duale Problem zu lösen, wie beispielsweise die *Sequential Minimal Optimization*[Ng], auf die ich hier jedoch nicht weiter eingehen werde.

Die SVM ist ein binärer Klassifizierer $c(\mathbf{x}) \in \{-1, 1\}$. Für die Klassifizierung bei einem Mehrklassenproblem gibt es verschiedene Strategien. Auf einige davon werde ich nun gesondert im folgenden Kapitel eingehen.

## 3.3 Paarweise Klassifizierung

Ein Nachteil der SVM ist, dass der Wertebereich der Entscheidungsfunktion nur zwei Zustände besitzt. Bei vielen Problemen ist die Anzahl der Zielklassen jedoch größer. Ich werde in diesem Kapitel auf drei Strategien eingehen, die binäre Klassifizierer um die Möglichkeit erweitern, auch Mehrklassenprobleme zu lösen.

### 3.3.1 Einer gegen den Rest

Die „Einer gegen den Rest"-Strategie (im Englischen auch *one-versus-rest* oder *one-against-rest* genannt) ist die am häufigsten verwendete Methode, um Mehrklassenproblem mit der SVM zu lösen[Sch01]. Dabei werden $k = |Y|$ binäre Klassifizierer $c^{(1)}, \ldots, c^{(k)}$ trainiert.

$$c(\mathbf{x}) = sgn\left(g^{(j)}(\mathbf{x})\right), \quad g^{(j)}(\mathbf{x}) = \sum_{i=1}^{m} \alpha_i^{(j)} y_i k(\mathbf{x}, \mathbf{x}_i) + b^{(j)} \tag{3.25}$$

Für jeden Klassifizierer wird eine eigene Trainingsmenge erstellt, für die gilt:

$$D^{(j)} = \{(\mathbf{x_1}, \hat{y}_1), \ldots, (\mathbf{x_n}, \hat{y}_n)\}, \quad \forall(\mathbf{x_i}, y_i) \in D, \hat{y}_i = \begin{cases} -1 & \text{if } y_i \neq j \\ 1 & \text{if } y_i = j \end{cases} \tag{3.26}$$

Jeder Klassifizierer $c^{(j)}$ trennt dabei eine Zielklasse von den restlichen Zielklassen. Für einen neuen Datensatz $\mathbf{x}$ wird die Zielklasse ausgewählt, bei der der Wert der Entscheidungsfunktion $g^{(j)}$ am größten ist. Diese Entscheidungsstrategie wird *winner-takes-all* genannt.

$$c(\mathbf{x}) = \arg\max_{j \in Y} g^{(j)}(\mathbf{x}) \tag{3.27}$$

Im linken Bild 3.4 sind die Hyperebenen (gestrichelte Linie) der binären Klassifizierer zu sehen. Die entsprechende Entscheidungsstrategie *winner-takes-all* ist mit der roten Linie gekennzeichnet. Auf dieser Abbildung lässt sich erkennen, dass die resultierende Trennfunktion zwischen der zweiten und dritten Klasse den Abstand nicht optimal maximiert.

### 3.3.2 Einer gegen Einen

Die „Einer gegen Einen"-Strategie (auch *one-against-one* bzw. *one-versus-one* genannt) wird in der von mir verwendeten LibSVM-Bibliothek benutzt[Cha01]. Dafür werden für alle möglichen paarweisen Kombinationen der Zielklassen $Y$ binäre Klassifizierer erstellt. Dafür wird der Trainingsdatensatz in $k = \frac{|Y|(|Y|-1)}{2}$ Teildatensätze unterteilt. Die Teilmenge für die Kombination der Zielklassen $r$ und $s$ wird

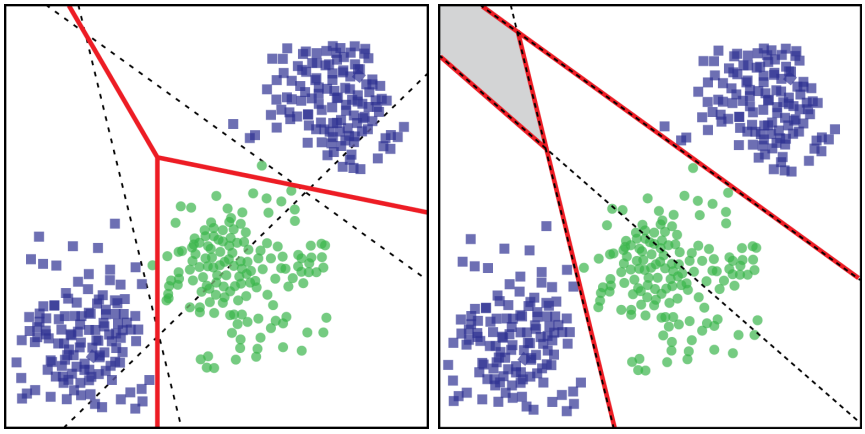


**Abbildung 3.4:** Links: „Einer gegen den Rest", Rechts: „Einer gegen Einen"-Strategie, Gestrichtelten linien sind die binären Klassifizierer, die roten Linien ist (Quelle: [Kre99])

gebildet mit:

$$D^{(r,s)} = \{(\mathbf{x_i}, \hat{y}_i) | (\mathbf{x_i}, y_i) \in D, y_i = r \vee y_i = s\}, \quad \hat{y}_i = \begin{cases} -1 & \text{if } y_i \neq j \\ 1 & \text{if } y_i = j \end{cases} \tag{3.28}$$

Für die Entscheidungsfunktion der LibSVM wird die *maxwins*-Strategie (auch *voting*-Strategie) verwendet. Welcher Zielklasse ein neuer Datensatz $\mathbf{x}$ zugeordnet wird, entscheidet die Mehrheitswahl. Jeder binäre Klassifizierer $c^{(r,s)}(\mathbf{x})$ gibt dabei eine Stimme ab. Ist $c^{(r,s)}(\mathbf{x}) > 0$ erhält die Zielklasse $r$ eine Stimme, andernfalls die Klasse $s$.

$$c^{(r,s)}(\mathbf{x}) = \sum_{(\mathbf{x}_i, y_i) \in D^{(r,s)}} \alpha_i^{(r,s)} y_i k(\mathbf{x}, \mathbf{x}_i) + b^{(i,j)} \tag{3.29}$$

$$c(\mathbf{x}) = \arg\max_{r \in Y} \sum_{s \in Y} \frac{1}{2} \left| c^{(r,s)}(\mathbf{x}) + 1 \right|, \quad c^{(r,s)}(\mathbf{x}) : \mathbb{R}^p \to \{-1, 1\} \tag{3.30}$$

Es kann jedoch geschehen, das mehrere Zielklassen die größte Stimmanzahl bekommen haben. Die LibSVM löst das Problem damit, dass sie dem Datensatz die Zielklasse zuweist, die numerisch den kleineren Wert besitzt.

### 3.3.3 Tournament

Für dieses Verfahren wird ein Binärbaum als Entscheidungsfunktion genutzt. Ähnlich wie beim Entscheidungsbaum (siehe [Ste08, Part. Entscheidungsbäume]) wird der Datensatz von der Wurzel bis in die Blätter weitergereicht. Die Knoten dieses gerichteten azyklischen Graphen sind, im Gegensatz zum Entscheidungsbaum, die binären Klassifizierer. Jeder dieser Klassifizierer $c^{(R,S)}$ unterteilt die Menge Zielklas-

sen in zwei Teilmengen.

$$Y \qquad \text{Menge aller Zielklassen} \tag{3.31}$$

$$G \subseteq Y \tag{3.32}$$

$$R \subset G \tag{3.33}$$

$$S = G \setminus R \tag{3.34}$$

$$c^{(R,S)}(\mathbf{x}) = \begin{cases} -1 & \text{if } c(\mathbf{x}) \in S \\ 1 & \text{if } c(\mathbf{x}) \in R \end{cases} \tag{3.35}$$

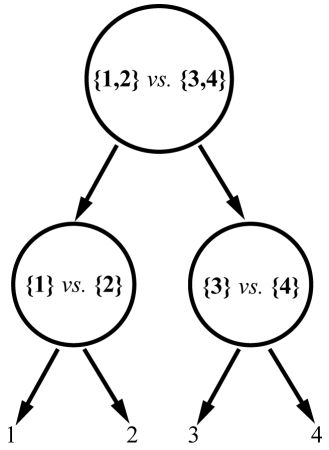



**Abbildung 3.5:** Das Tournament benutzt einen Baum mit binären Klassifizierern

Im Wurzelknoten entspricht $G = Y$. Entscheidet dieser, dass die Zielklasse des Datensatzes in der Menge $R$ enthalten ist, wird $R$ im nächsten linken Kindknoten wieder in zwei Teilmengen zerlegt. Ist die Zielklasse in $S$, wird sie an den rechten Kindknoten weitergeleitet. Der Datensatz wird solange im Baum weitergereicht, bis in den Teilmengen nur noch eine Zielklasse enthalten ist. Dann wird dem Datensatz diese Zielklasse zugewiesen. Bei der Suche nach dem optimalen Binärbaum kann man die Knoten eines Baumes anhand eines Gütemaßes auswählen. Das Gütemaß gibt dabei die Trennfähigkeit des Klassifikators an. Mögliche Gütemaße für binäre

Klassifizierer:

$$\text{accuracy}^{(R,S)} = \frac{|\{(\mathbf{x}, y) \in D | y \in R \wedge c(\mathbf{x}) \in R \vee y \in S \wedge c(\mathbf{x}) \in S\}|}{|\{(\mathbf{x}, y) \in D\}|} \tag{3.36}$$

$$\text{precision}^{(R,S)} = \frac{|\{(\mathbf{x}, y) \in D | y \in R \wedge c(\mathbf{x}) \in R\}|}{|\{(\mathbf{x}, y) \in D | y \in R \wedge c(\mathbf{x}) \in R \vee y \in S \wedge c(\mathbf{x}) \in R\}|} \tag{3.37}$$

$$\text{recall}^{(R,S)} = \frac{|\{(\mathbf{x}, y) \in D | y \in R \wedge c(\mathbf{x}) \in R\}|}{|\{(\mathbf{x}, y) \in D | y \in R \wedge c(\mathbf{x}) \in R \vee y \in R \wedge c(\mathbf{x}) \in S\}|} \tag{3.38}$$

$$\text{F}^{(R,S)} = 2\frac{\text{precision}^{(R,S)} \cdot \text{recall}^{(R,S)}}{\text{precision}^{(R,S)} + \text{recall}^{(R,S)}} \quad \text{F-Maß} \tag{3.39}$$

# KAPITEL 4

## Experimente

Am 1. April 2008 wurden die Aufgaben für den DataMiningCup 2008 freigeschaltet. Nach einer kurzen Einarbeitung in das umfangreiche Gebiet des Dataminings meldete ich mich am 3. April 2008 bei dem Wettbewerb an. Kurz darauf bekam ich die Zugangsdaten und konnte die Trainigs- und Testdaten sowie eine detaillierte Aufgabenbeschreibung vom Portal des Wettbewerbes[1] herunterladen. Bis zur Abgabe der Ergebnisse am 15. Mai 2008 arbeitete ich mich tiefer in verschiedene Teilgebiete des Dataminings ein und versuchte, geeignete Methoden für die gestellte Aufgabe zu finden. In den letzten beiden Kapiteln habe ich die wichtigsten Methoden erklärt, die ich im Rahmen des DataMiningCup 2008 verwendet habe. In diesem Kapitel werde ich nun auf die Anforderungen des Wettbewerbes und die konkrete Vorgehensweise meiner Lösung eingehen. Zunächst werde ich dazu die Umfragedaten analysieren und deren Eigenheiten herausstellen. In Abschnitt 4.2 wird der Aufbau der Experimente beschrieben und konkret dargelegt, welche Schritte zur Anpassung an die Eigenheiten der Umfragedaten notwendig waren. Danach werden die Ergebnisse der Dimensionsreduktion vorgestellt. Der nächste Abschnitt widmet sich dem Mehrklassenproblem, bevor im letzten Teil die Ergebnisse der verwendeten Verfahren verglichen werden.

### 4.1 Datenanalyse

Insgesamt enthält der Trainingsdatensatz 113 476 Datensätze, in denen sich dabei jeweils wiederum bis zu 71 Attribute befinden. Eines dieser Attribute enthält die ID des Datensatzes, ein weiteres die Zielklasse. Die Zielklasse besitzt dabei fünf Ausprägungen $Y = \{0, 1, 2, 3, 4\}$. Diese beiden Attribute können aus dem Merkmalsraum entfernt werden. Somit hat der Merkmalsraum die Dimension $p = 69$. Jeder Datensatz wird als Tupel dargestellt $d_i\left(\mathbf{x}_i, y_i\right), \mathbf{x}_i \in \mathbb{R}^p, y_i \in Y$.

1 http://www.data-mining-cup.de/

Für die ersten, einfachen Datenanalysen begann ich mit der Entwicklung einer Testumgebung. Die in Java programmierte Umgebung sollte mir zunächst ein Histogramm für die Verteilung der Datensätze über die Zielklassen liefern. Wie im

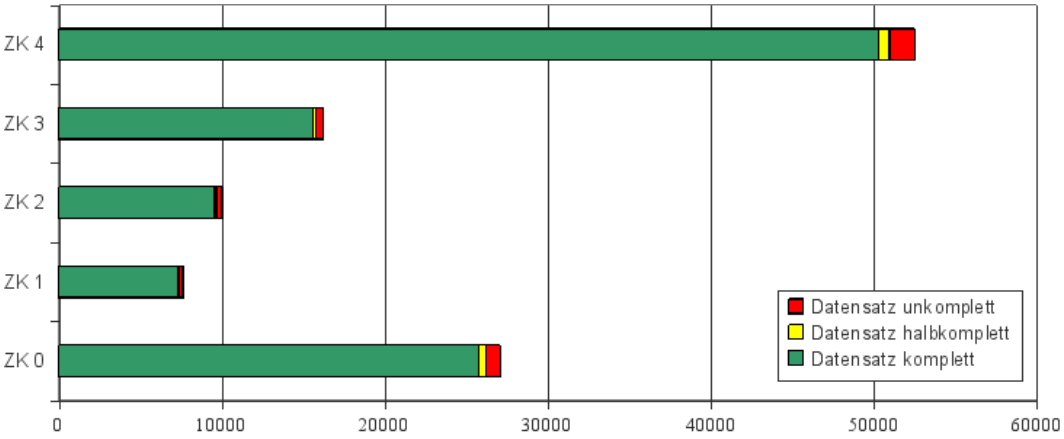



**Abbildung 4.1:** Histogramm der Zielklassenverteilung

Diagramm 4.1 ersichtlich, sind die Datensätze sehr unterschiedlich über die Zielklassen verteilt. Diese unbalancierten Zielklassen können zu Problemen beim Erstellen eines Klassifizierers führen. Ein Klassifizierer versucht beim Trainieren, die Fehler beim Klassifizieren zu minimieren. Dazu nehmen viele Klassifizierer eine gleiche Zielklassenverteilung im Trainingsdatensatz und in den später zu klassifizierenden Daten an. Trainiert man einen Klassifizierer mit einem Daten, bei dem 99% der Datensätze der ersten Zielklasse angehören, und die restlichen der zweiten Zielklasse, so erreicht man mit einer vollständigen Zuordnung aller Datensätze in die erste Zielklasse eine Missklassifikationsrate von nur einem Prozent. Wird dieser Klassifizierer dann jedoch bei Daten benutzt, deren Zielklassenverteilung gleich ist, wird seine Missklassifikationsrate auf 50% steigen.

Im nächsten Schritt habe ich den Datensatz auf fehlende Datenwerte(*missing values*) hin untersucht. Dabei gab es genau drei Gruppen von Datensätzen. Bei 95,7% der Datensätze waren alle 69 Attribute belegt, 1,3% der Datensätze enthielten lediglich 18 Attribute und 3% nur 16 Attribute. Im Bild 4.1 ist die Zielklassenverteilung der drei Gruppen dargestellt. Diese zeigten jedoch keinen signifikanten Unterschied. Da die meisten Datensätze vollständig waren, entschied ich mich, alle Tests nur mit diesem Datensatz durchzuführen und vernachlässigte zunächst die restlichen beiden Datensätze.

Nach der Frage der Vollständigkeit richtete ich mein Interesse auf die Skalenniveaus der Attribute. Für einige der verwendeten Dimensionsreduktions- und Klassifizierungsverfahren werden bestimmte Skalenniveaus vorausgesetzt. Nach der Anmeldung zum DataMiningCup 2008 erhielt man neben der Aufgabenbeschreibung noch ein weiteres Dokument (siehe Anhang A). Dort waren alle Attribute mit einer kur-

zen Beschreibung und den Attributausprägungen enthalten. Zehn dieser Attribute waren nominalskaliert; die restlichen 59 mindestens ordinalskaliert, enthielten also eine Ordnung. Drei sind sogar verhältnisskaliert: das Alter, die Anzahl der bisher gespielten Lotterien und die z-standardisierte Anzahl von Lotterielosen, die in der ersten Lotterieklasse gekauft wurden.

Die Aufgabenstellung des DataMiningCup enthielt noch eine Besonderheit: eine Kostenmatrix. Sie ist in der Tabelle 4.1 dargestellt. Wie bereits im ersten Kapitel be-

| | Vorhersage | | | | |
|---|---|---|---|---|---|
| Zielklasse | 0 | 1 | 2 | 3 | 4 |
| 0 | 20 | 5 | 0 | -5 | -10 |
| 1 | 0 | 20 | 5 | 0 | -5 |
| 2 | -10 | 0 | 20 | 5 | 0 |
| 3 | -20 | -10 | 0 | 20 | 5 |
| 4 | -40 | -20 | -10 | 0 | 20 |

**Tabelle 4.1:** Kostenmatrix des DataMiningCup 2008

schrieben, beinhalten die Datensätze fünf verschiedene Zielklassen. Diese orientieren sich an der Spieldauer eines Lotterieteilnehmers, sind also ordinalskaliert. Diese Tatsache spiegelt sich auch in der Kostenmatrix wieder. Ist ein Datensatz der Zielklasse 4 zugeordnet (letzte Spalte der Tabelle 4.1), so ist die Bestrafung für Vorhersage in die Zielklasse 0 höher als die Vorhersage in die Klasse 3. Diese Kostenmatrix wird für die Berechnung der Punkte und die Auswertung des Wettbewerbes verwendet. Jeder Datensatz wird auf seine korrekte Einordnung in die Zielklasse hin überprüft. Je näher die Zuordnung an der tatsächlichen Zielklasse liegt, desto mehr Punkte gibt es.

## 4.2 Aufbau der Experimente

Für die Experimente benutzte ich mehrere Programme: SPSS[1] (für die Diskriminanzanalyse als einer der ersten Klassifzierungsversuche), Weka[2] mit der Erweiterung LibSVM[3] (für die Datenanalyse, Dimensionsreduktion und zum Erstellen der Klassifizierer), und das Programm JavaNNS[4] (zum Erstellen und Trainieren eines Autoencoders zur Dimensionsreduktion). Da jedes dieser Programme unterschiedliche Dateiformate besitzt, begann ich, meine eigene Testumgebung weiterzuentwickeln, die die unterschiedlichen Dateiformate umwandeln konnte. Nach den ersten

1 http://www.spss.com/software/
2 http://www.cs.waikato.ac.nz/ml/weka/
3 http://www.csie.ntu.edu.tw/~cjlin/libsvm/
4 http://www.ra.cs.uni-tuebingen.de/software/JavaNNS/

Datenanalysen verwendete ich die Diskriminanzanalyse für einen ersten Klassifikationsversuch.

| | Vorhersage | | | | |
|---|---|---|---|---|---|
| Ziel | 0 | 1 | 2 | 3 | 4 |
| 0 | 44,7% | 0,0% | 0,0% | 12,2% | 43,1% |
| 1 | 37,3% | 0,0% | 0,0% | 23,4% | 39,2% |
| 2 | 26,4% | 0,0% | 0,0% | 32,4% | 41,1% |
| 3 | 18,4% | 0,0% | 0,0% | 46,8% | 34,8% |
| 4 | 9,3% | 0,0% | 0,0% | 14,1% | 76,6% |
| accuracy: 52,8 % | | | | | |

| | gewichtete Vorhersage | | | | |
|---|---|---|---|---|---|
| Ziel | 0 | 1 | 2 | 3 | 4 |
| 0 | 230480 | 0 | 0 | -15790 | -111110 |
| 1 | 0 | 0 | 0 | 0 | -14310 |
| 2 | -25370 | 0 | 60 | 15560 | 0 |
| 3 | -57560 | 0 | 0 | 146020 | 27140 |
| 4 | -186520 | 0 | -10 | 0 | 770680 |
| Punkte gesamt: 779 270 (35,9%) | | | | | |

**Tabelle 4.2:** Die ersten Ergebnisse mit Der SPSS-Diskriminanzanalyse. Links

In der linken Tabelle 4.2 sind die Ergebnisse der Vorhersage zu sehen. Die Ergebnisse sind über eine 10-fache Kreuzvalidierung entstanden. Bei einer $k$-fachen Kreuzvalidierung wird der gegebene Datensatz $D$§ in $k$ Teilmengen $D_1, \ldots, D_k$ unterteilt. Der Klassifizierer, in diesem Fall die Diskriminanzanalyse, wird nun $k$-mal trainiert. Bei jedem $i$-ten Training wird eine Teilmenge $D_i$ als Testmenge benutzt. Als Trainingsmenge $D_t$ werden die restlichen Datensätze verwendet $D_t = D \setminus D_i$. Die Klassifizierungsergebnisse werden dann als Durchschnitt aus allen $k$ Trainingsdurchläufen berechnet.

Die errechnete *accuracy* ist die Summe der $\text{accuracy}_i$ der einzelnen Zielklassen, gewichtet mit der a-priori-Wahrscheinlichkeit der Zielklasse $\sum_{y_i \in Y} p(y_i) \cdot \text{accuracy}_i$. Die rechte Tabelle 4.2 gibt die Anzahl der Punkte an, die sich aus der Vorhersage und der Kostenmatrix aus Tabelle 4.1 ergibt. Die prozentuale Punkteangabe bezieht sich auf die maximal erreichbare Punktzahl und wurde von mir hauptsächlich wegen der besseren Lesbarkeit genutzt. Die Punkte wurden unter der Annahme berechnet, dass der Testdatensatz die gleiche Klassenverteilung wie der Trainingsdatensatz besitzt.

Für die nächsten Experimente setzte ich die umfangreiche Datamining-Software Weka bzw. deren API ein. Ziel war es, automatisierte Tests zu erstellen, die die gleichzeitige Arbeit an mehreren Rechnern und eine einfache Evaluierung der Ergebnisse ermöglichen sollte. Als Gütemaß für die Klassifizierer nutzte ich die Punktezahl, die es zu maximieren galt. Für einen ersten Klassifizierungsversuch mit Weka verwendete ich die Support Vector Machine aus der Bibliothek LibSVM mit den Voreinstellungen. Vorausgewählt war der nicht-lineare Kernel *radial basis* (3.16). Diesen trainierte ich lediglich mit den Datensätzen, die keine fehlenden Messwerte enthielten.

Die *accuracy* in der Tabelle 4.3 zeigt nur einen geringen Unterschied im Vergleich zu dem Ergebnis der Diskriminanzanalyse. Werden jedoch die Punkte verglichen, schneidet die SVM sehr viel schlechter ab. Ein Hauptgrund hierfür ist in der schlechten Klassifikationsrate für die Zielklasse 3 zu sehen, die in der Zielklassenverteilung immerhin einen Anteil von 14,4% besitzt. In anderen Zielklassen, wie

| | Vorhersage | | | | |
|---|---|---|---|---|---|
| Ziel | 0 | 1 | 2 | 3 | 4 |
| 0 | 45,3% | 0,0% | 0,0% | 3,7% | 51,0% |
| 1 | 40,3% | 0,2% | 0,0% | 6,9% | 52,5% |
| 2 | 34,0% | 0,0% | 0,0% | 7,5% | 58,4% |
| 3 | 28,3% | 0,0% | 0,0% | 11,8% | 59,9% |
| 4 | 12,2% | 0,0% | 0,0% | 3,1% | 84,6% |
| accuracy: 51,6 % | | | | | |

| | gewichtete Vorhersage | | | | |
|---|---|---|---|---|---|
| Ziel | 0 | 1 | 2 | 3 | 4 |
| 0 | 233480 | 35 | 0 | -4785 | -131440 |
| 1 | 0 | 300 | 20 | 0 | -19160 |
| 2 | -32610 | 0 | 100 | 3620 | 0 |
| 3 | -88360 | -60 | 0 | 36720 | 46700 |
| 4 | -246480 | -60 | -50 | 0 | 851040 |
| Punkte gesamt: 649 010 (29,9%) | | | | | |

**Tabelle 4.3:** Die SVM mit allen Attributen

zum Beispiel den Zielklasse 4 und 0, ist die Trennung sehr viel besser als bei der Diskriminanzanalyse. Die Konfusionsmatrix der SVM wurden ebenfalls mit einer 10-fachen Kreuzvalidierung berechnet. Mit dem Datensatz, der 108 600 Umfragedatensätze enthielt, dauerte die Berechnung auf einem aktuellen Zweikernprozessor (à 2,1 GHz) ca. 32 Stunden. Schon die Berechnung der Diskriminanzanalyse nahm ca. 2 Stunden auf einem wesentlich älteren Rechner in Anspruch. Deshalb entschied ich mich, die Dimensionen des Merkmalsraumes zu reduzieren. Auf die Experimente und die Ergebnisse verschiedener Dimensionsreduktionsverfahren möchte ich separat im Kapitel 4.3 eingehen.

Eine weitere Möglichkeit, schneller zu den Klassifikationsergebnissen zu kommen, ist die Verwendung eines anderen Validierungsmaßes für den Klassifizierer. Bisher setzte ich die Kreuzvalidierung mit 10 Unterteilungen ein. Der Klassifizierer wurde also zehnmal trainiert, bevor die Konfusionsmatrix vorlag. Bei Support Vektor Maschinen kommt hinzu, dass diese von Natur aus nur binäre Klassifizierer sind. Daher werden für ein Mehrklassenproblem mehrere binäre Klassifizierer erstellt. Die LibSVM benutzte dazu die „Einer gegen Einen"-Strategie (siehe Kapitel 3.3.2). Es wurden daher $10 = k = \frac{|Y|\,(|Y|-1)}{2}, |Y| = 5$ Klassifizierer erstellt. Mit der verwendeten Kreuzvalidierung sind das schon 100 Klassifizierer, die erzeugt werden müssen. Um alle folgenden Experimente zu beschleunigen, entschied ich mich für eine sehr einfache Validierung der Klassifizierer. Dazu nutzte ich 90% der Datensätze zum Trainieren und testete den Klassifizierer auf den restlichen 10% der Daten.

Die SVM konnte in den beiden Zielklassen 0 und 4, die immerhin 70% der Datensätze ausmachten, bessere Ergebnisse liefern als die Diskriminanzanalyse. Da die SVM ein binärer Klassifizierer ist, hoffte ich, durch eine andere Strategie zur Lösung des Mehrklassenproblems eine noch bessere Trennung aller Zielklassen zu erreichen. Auf die Experimente für die Lösung des Mehrklassenproblems werde ich in Kapitel4.4 detailliert eingehen.

Für die Experimente konnte ich ein neues Labor verwenden, in dem mir acht Rechner zur Verfügung standen. Ziel war es, alle Rechner optimal zu nutzen. In der Datamining-Software Weka gibt es eine Umgebung, mit der man die Arbeit auf mehreren vernetzten Rechnern ausführen kann. Leider waren die Rechner zum damaligen

Zeitpunkt noch nicht miteinander vernetzt. Zudem war es mit Weka (Version 3.5.7) nicht möglich, mehrere Experimente gleichzeitig auf einem Rechner zu starten. Da jeder dieser Rechner jedoch vier Prozessorkerne besaß, sollten diese auch voll ausgenutzt werden. Deshalb erweiterte ich meine Testumgebung. Diese arbeitete hauptsächlich mit der API von Weka und der eingebundenen LibSVM und sollte es mir ermöglichen, mehrere SVM gleichzeitig zu starten. Weiterhin kann die Testumgebung eine neue SVM starten, wenn nicht mehr alle Prozessorkerne ausgelastet sind. Dafür wurde eine Dateiliste angegeben, die das Programm abarbeiten sollte. Als weiterer Parameter wurde ein Ausgabeordner angegeben, in dem die Klassifizierungsergebnisse der SVM gespeichert werden sollten. Für jedes erfolgreiche Experiment wurde eine Datei erstellt, in der zwei Klassifizierungsmatrizen gespeichert wurden. In einer Matrix waren die reinen Klassifizierungsergebnisse der SVM, die andere enthielt die mit der Kostenmatrix 4.1 gewichteten Ergebnisse. Der Dateiname setzte sich aus den prozentualen und den tatsächlichen Punkten zusammen. Dies ermöglichte mir, schnell den besten Klassifizierer zu finden. Auch die Modelle der trainierten Klassifizierer wurden für einen späteren Gebrauch in diesem Ordner gespeichert.

## 4.3 Dimensionsreduktion

Die Dimensionsreduktion sollte besonders die Erstellung der Klassifizierer beschleunigen. Als erstes Verfahren hierfür setzte ich die schrittweise Diskriminanzanalyse 2.3 ein. Als Gütemaß benutzte ich Wilk's Lambda, als Erklärungsmaß die F-Werte mit den eingestellten Werten. Wichtig ist, dass durch die Dimensionsreduktion keine wesentliche Verschlechterung der Klassifizierung zustande kommt. Die schrittweise Diskriminanzanalyse ist ein Attributauswahlverfahren. Den ausgewählten Attributen muss es daher trotzdem möglich sein, eine größtmögliche Intervarianzen zu erzeugen, um die Zielklassen von einander trennen zu können. Die schrittweise Diskriminanzanalyse wählte 19 Attribute aus den 69 aus. In der rechten Tabelle 4.4 sind die Ergebnisse der Diskriminanzanalyse mit den 19 Attributen dargestellt. Trotz der Reduzierung der Attribute auf weniger als $\frac{1}{3}$ ist bei den Konfusionsmatrizen kein signifikanter Unterschied zu erkennen. Diese Attributauswahl wurde dann auch für das Trainieren der SVM benutzt. Die besten Attribute hatte ich als Liste gespeichert, wobei das erste Attribut die Nummer eins besitzt. Die Weka Software ist jedoch an einigen Stellen inkonsistent bei der Zählweise der Attribute. Verwendet man beispielsweise einen Attributfilter, so werden die Attribute beginnend mit eins nummeriert. Will man die Attribute jedoch direkt über den Datensatz ansprechen, so wird das erste Attribut mit dem Index 0 angesprochen. Durch einen Rechenfehler passierte es, dass ich für die neu erstellten Datensätze die falschen Attribute auswählte. Wie sich später herausstellen sollte, lieferte die SVM mit diesen Attributen

| | Vorhersage | | | | |
|---|---|---|---|---|---|
| Ziel | 0 | 1 | 2 | 3 | 4 |
| 0 | 44,7% | 0,0% | 0,0% | 12,2% | 43,1% |
| 1 | 37,3% | 0,0% | 0,0% | 23,4% | 39,2% |
| 2 | 26,4% | 0,0% | 0,0% | 32,4% | 41,1% |
| 3 | 18,4% | 0,0% | 0,0% | 46,8% | 34,8% |
| 4 | 9,3% | 0,0% | 0,0% | 14,1% | 76,6% |

accuracy: 52,8 %
Punkte gesamt: 779 270 (35,9%)

| | Vorhersage Attributauswahl | | | | |
|---|---|---|---|---|---|
| Ziel | 0 | 1 | 2 | 3 | 4 |
| 0 | 42,7% | 0,0% | 0,0% | 13,2% | 44,1% |
| 1 | 35,7% | 0,0% | 0,0% | 24,6% | 39,7% |
| 2 | 25,0% | 0,0% | 0,0% | 34,5% | 40,6% |
| 3 | 17,3% | 0,0% | 0,0% | 48,6% | 34,1% |
| 4 | 9,1% | 0,0% | 0,0% | 14,8% | 76,1% |

accuracy: 52,4 %
Punkte gesamt: 775 015 (35,7%)

**Tabelle 4.4:** SPSS-Diskriminanzanalyse links: ohne Attributauswahl, rechts: mit Attributauswahl

aber bessere Ergebnisse als mit den eigentlich durch die schrittweise Diskriminanzanalyse ausgewählten Attributen. Die Tabelle 4.5 zeigt eine Gegenüberstellung der SVM mit allen Attributen und mit der Attributauswahl. Die Attributauswahl hat nur einen geringen Einfluss auf die Konfusionsmatrix. Die Punktebewertung hat sich sogar leicht verbessert.

| | Vorhersage | | | | |
|---|---|---|---|---|---|
| Ziel | 0 | 1 | 2 | 3 | 4 |
| 0 | 45,3% | 0,0% | 0,0% | 3,7% | 51,0% |
| 1 | 40,3% | 0,2% | 0,0% | 6,9% | 52,5% |
| 2 | 34,0% | 0,0% | 0,0% | 7,5% | 58,4% |
| 3 | 28,3% | 0,0% | 0,0% | 11,8% | 59,9% |
| 4 | 12,2% | 0,0% | 0,0% | 3,1% | 84,6% |

accuracy: 51,6 %
Punkte gesamt: 649 010 (29,9%)

| | Vorhersage Attributauswahl | | | | |
|---|---|---|---|---|---|
| Ziel | 0 | 1 | 2 | 3 | 4 |
| 0 | 43,3% | 0,0% | 0,0% | 4,2% | 52,5% |
| 1 | 37,5% | 0,2% | 0,1% | 8,3% | 53,9% |
| 2 | 30,9% | 0,1% | 0,1% | 9,8% | 59,2% |
| 3 | 27,0% | 0,0% | 0,0% | 12,6% | 60,3% |
| 4 | 11,9% | 0,0% | 0,0% | 3,8% | 84,3% |

accuracy: 51,1 %
Punkte gesamt: 649 975 (29,9%)

**Tabelle 4.5:** SVM Links: ohne Attributauswahl, Rechts: mit Attributauswahl der schrittweisen Diskriminanzanalyse

Als nächstes testete ich ein anderes Dimensionsreduktionsverfahren. Die im Kapi-

tel 2.2 beschriebene Hauptkomponentenanalyse sollte mir mehrere Features liefern, die aus den Linearkombinationen der Attribute bestehen. Der Featureraum wurde so erstellt, dass dieser mindestens 95% der Varianz erklären sollte. Die Hauptkomponentenanalyse geht von einer Normalverteilung der Attribute aus. Im Anhang A habe ich die Histogramme der Attribute aufgelistet, die die schrittweise Diskriminanzfunktion ausgewählt hat. Tatsächlich besitzen die meisten Attribute eine annähernde Normalverteilung. Die Hauptkomponentenanalyse erzeugte 37 Features, also einen größeren Featureraum als die schrittweise Diskriminanzanalyse. Mit diesen Features trainierte ich erneut die SVM.

| | Vorhersage | | | | |
|---|---|---|---|---|---|
| Ziel | 0 | 1 | 2 | 3 | 4 |
| 0 | 27,6% | 0,0% | 0,0% | 0,0% | 72,4% |
| 1 | 27,0% | 0,0% | 0,0% | 0,0% | 73,0% |
| 2 | 18,2% | 0,0% | 0,0% | 0,0% | 81,8% |
| 3 | 13,9% | 0,0% | 0,0% | 0,0% | 86,1% |
| 4 | 6,2% | 0,0% | 0,0% | 0,0% | 93,8% |
| accuracy: 50,02 % | | | | | |

| | gewichtete Vorhersage | | | | |
|---|---|---|---|---|---|
| Ziel | 0 | 1 | 2 | 3 | 4 |
| 0 | 142620 | 0 | 0 | 0 | -186630 |
| 1 | 0 | 0 | 0 | 0 | -26625 |
| 2 | -17440 | 0 | 0 | 0 | 0 |
| 3 | -43440 | 0 | 0 | 0 | 67175 |
| 4 | -124200 | 0 | 0 | 0 | 944000 |
| Punkte gesamt: 755 460 (34.8%) | | | | | |

**Tabelle 4.6:** Die SVM mit den Features der Hauptkomponentenanalyse

Die Dimensionsreduktion der Hauptkomponentenanalyse zeigt das Problem der unbalancierten Daten sehr deutlich. Die *accuracy* und die Punkte liegen zwar in einem guten Bereich (vgl. 4.4), jedoch zeigt die Konfusionsmatrix in der linken Tabelle 4.6, dass die guten Werte durch die Präferenz der Zielklasse 4 erzeugt wurden. Ein großes Problem bei der Hauptkomponentenanalyse ist der Verzicht auf Zielklassenunterschiede bei der Komponentenerstellung. Aufgrund der relativ schlechten Klassifizierungsergebnisse mit den Hauptkomponenten verwarf ich diese Methode der Dimensionsreduktion für die späteren Experimente.

Die schrittweise Diskriminanzanalyse lieferte den Beweis, dass eine starke Reduzierung der Attribute die Klassifizierungsergebnisse nicht so stark beeinflusst. Die falsche Auswahl von Attributen für das Trainieren der SVM zeigte die Optimierungsmöglichkeiten in diesem Bereich. Da die Diskriminanzanalyse andere Optimierungskriterien besitzt als die SVM, wählte ich ein an die SVM angepasstes Attributauswahlverfahren. Ich verwendete daher eine lineare SVM mit der *backward-elimination*-Strategie 2.4. Da die Berechnung der SVM mit *radial basis*-Kernel bereits 32 Stunden dauerte, setzte ich die weniger komplexe lineare Kernelfunktion für die Attributaus-

wahl ein. Ich habe auch das Validierungsmaß angepasst und 90% der Datensätze zum Trainieren - die übrigen zum Testen verwendet. Zudem wurden bei jeder Iteration der *backward-elimination*-Strategie fünf Attribute aus der Auswahl entfernt. Trotz dieser Optimierung dauerte es mehrere Tage, bis die Attributauswahl abgeschlossen war und lediglich 20 Attribute übrig waren. Die Klassifizierungsergebnisse zeigten jedoch eine leichte Verschlechterung im Vergleich zur Attributauswahl der Diskriminanzanalyse. Die Punktezahl sank um 29 715 Punkte auf 28,5% ; die *accuracy* auf 49,8%.

Der Autoencoder versucht, wie auch die Hauptkomponentenanalyse, eine Dimensionsreduktion durch eine Transformation des Merkmalsraumes zu erreichen. Das im Kapitel 2.5 beschriebene neuronale Netzwerk trainierte ich mit der Software JavaNNS. Diese Software besaß jedoch keine dokumentierte Programmierschnittstelle, sondern nur ein graphisches Benutzerinterface. Automatisierte Experimente waren somit nicht möglich. Trainiert wurden zwei verschiedene neuronale Netzwerke. Beide Autoencoder besaßen jeweils 69 Neuronen in der Eingabe- und Ausgabeschicht und hatten das Ziel, die Dimensionen des Merkmalsraumes auf 20 zu reduzieren. Ein Autoencoder war ein 69-50-20-50-69 Netzwerk und sollte die Dimensionen schrittweise verringern. Der andere Autoencoder sollte die Dimensionen vor deren Reduzierung noch einmal erhöhen. Ich erhoffte mir durch das 69-100-20-100-69 Netzwerk, dass auch wichtige Attributkombinationen besser „erkannt" werden. Die Abbildung 4.2 zeigt die Fehlerrate über 200 Lernschritte à 100 Iterationen der beiden Autoencoder. Die Autoencoder wurden mit zufälligen Gewichten initialisiert und mit dem *Backpropagation*-Algorithmus gelernt.

Um die Klassifizierungsergebnisse mit der SVM zu erhalten, implementierte ich ein neuronales Netzwerk in meiner Testumgebung. Die gelernten Gewichte und den Aufbau der Autoencoder konnte ich in der Software JavaNNS speichern. Diese Datei wurde von meiner Testumgebung eingelesen um ein neuronales Netz zu erstellen. Für eine Dimensionsreduktion benötigte ich nur die Neuronen von der Eingabeschicht bis zur mittleren Zwischenschicht. Aus diesen wurde das neuronale Netz erstellt und mit den gelernten Gewichten initialisiert. Beim trainieren der SVM wurden die Attributwerte der Datensätze an die Eingabeschicht angelegt und dann durch das neuronale Netz in den Featureraum transformiert, den der Autoencoder trainiert hatte. Mit 90% der Datensätze aus dem Featureraum wurde dann die SVM trainiert. Mit den restlichen 10% wurde der Klassifizierer getestet. Die *accuracy* Betrug lediglich 46,2%; die Punkte dezimierten sich sogar auf 351 495 (16,8%).

Im folgenden Kapitel 4.4 werde ich für die Erstellung einer Mehrklassen-SVM mehrere binäre Klassifizierer erzeugen. Jeder dieser binären Klassifizierer sollte zwei Mengen von Zielklassen voneinander trennen. Um auch diese zu optimieren, erstellte ich für jeden binären Klassifizierer eine eigene Attributauswahl. Dazu verwendete

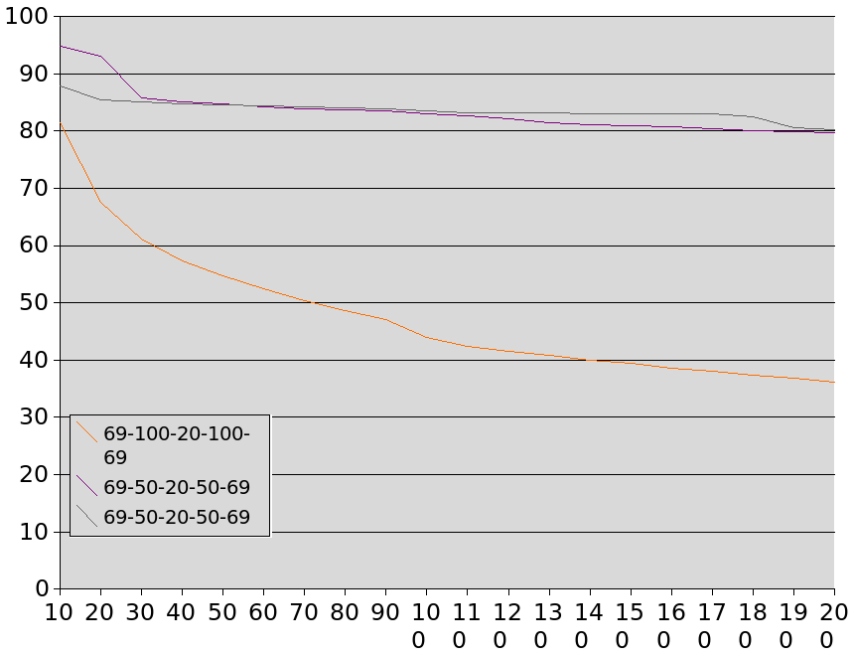



**Abbildung 4.2:** Lernraten der beiden Autoencoder.

ich wieder die schrittweise Diskriminanzanalyse, die lineare SVM und den einfachen Filter Information Gain 2.1.

## 4.4 Mehrklassen-Klassifikation

Die von LibSVM bereitgestellte SVM löste das Mehrklassenproblem mit einer „Einer gegen Einen"-Strategie. Da die Ergebnisse jedoch noch nicht zufriedenstellend waren, wollte ich die Leistungsfähigkeit der SVM als binären Klassifizierer testen. Hierfür erstellte ich einen, der die Zielklassen 0 und 4 trennen sollte. Diesen beiden Zielklassen sind immerhin 70% der Trainingsdatensätze zugeordnet und verursachen nach der gegebenen Kostenmatrix 4.1 die höchsten Kosten bei einer Fehlklassifikation. Die Tabelle 4.7 zeigt, dass der binäre Klassifizierer die beiden Zielklassen deutlich

| | Vorhersage Attributauswahl | | | | |
|---|---|---|---|---|---|
| Ziel | 0 | 1 | 2 | 3 | 4 |
| 0 | 43,3% | 0,0% | 0,0% | 4,2% | 52,5% |
| 1 | 37,5% | 0,2% | 0,1% | 8,3% | 53,9% |
| 2 | 30,9% | 0,1% | 0,1% | 9,8% | 59,2% |
| 3 | 27,0% | 0,0% | 0,0% | 12,6% | 60,3% |
| 4 | 11,9% | 0,0% | 0,0% | 3,8% | 84,3% |

| | Vorhersage | |
|---|---|---|
| Ziel | 0 | 4 |
| 0 | 56,7% | 43,3% |
| 4 | 15,0% | 85,0% |

**Tabelle 4.7:** Links: Mehrklassen SVM der LibSVM, Rechts: binäre SVM

besser voneinander trennen kann. Deshalb begann ich mit der Implementierung der Mehrklassen-SVM als Tournament, wie bereits in Kapitel 3.3.3 beschrieben. Dazu

unterteilte ich den Trainingsdatensatz in alle möglichen 70 binären Attributkombinationen. Bei dem ersten Versuch nutzte ich die vorhandene Attributauswahl der schrittweisen Diskriminanzanalyse. Diese Datensätze verteilte ich auf alle acht mir zur Verfügung stehenden Rechner und trainierte mit jedem Datensatz eine binäre SVM. Danach erstellte ich alle 120 möglichen Bäume des Tournaments mit Hilfe der trainierten Klassifizierer. Alle Bäume wurden anschließend mit dem gesamten Datensatz getestet. Die Ergebnisse dieser Test wurden dann, wie bereits beschrieben, für jeden Baum in einer extra Datei gespeichert. Da die prozentuale Punktzahl ein Teil des Dateinamens war, konnte ich schnell den besten Baum finden. Die entsprechende Datei enthielt die Klassifizierungsmatrix und die mit der Kostenmatrix gewichtete Matrix. Der beste Baum mit der Attributauswahl der schrittweisen Diskriminanzanalyse ist in der Abbildung 4.3 dargestellt; die zugehörigen Konfusionsmatrizen in der Tabelle 4.8.

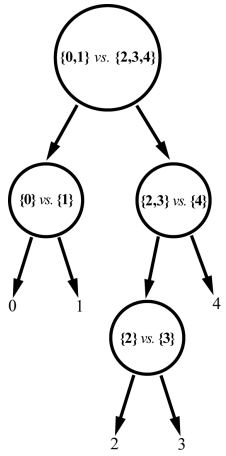



**Abbildung 4.3:** der nach den Punkten beste Baum für das Tournament

Diese Ergebnisse stellten einen großen Fortschritt dar. Jedoch können sie nicht direkt mit den Resultaten der Merklassen-SVM aus der LibSVM verglichen werden, da bei der Erstellung der Mehrklassen-SVM der LibSVM ein anderes Validierungsmaß benutzt wurde. So sind die Ergebnisse aus der Tabelle 4.3 durch eine Kreuzvalidierung entstanden. Jeder binäre Klassifizierer aus dem Tournament wurde jedoch mit den jeweiligen kompletten Teildatensätzen trainiert. Alle Datensätze, die die Zielklassen 2 und 3 besitzen, wurden also zum Trainieren der binären SVM $\{2\}\, vs.\, \{3\}$ benutzt und der gesamte Baum mit allen Datensätzen getestet.

Als nächstes versuchte ich, die Attributauswahl an die binären Klassifizierer anzupassen; also für jede mögliche binäre Attributkombination die besten Attribute zu ermitteln. Dazu verwendete ich drei unterschiedliche Verfahren: den Information Gain 2.1, die schrittweise Diskriminanzanalyse und die lineare SVM. Keines der Verfahren konnte jedoch das Ergebnis aus der Tabelle 4.8 erreichen, bei dem alle

| | Vorhersage | | | | |
|---|---|---|---|---|---|
| Ziel | 0 | 1 | 2 | 3 | 4 |
| 0 | 55,0% | 0,0% | 1,8% | 13,6% | 29,6% |
| 1 | 43,6% | 3,9% | 2,4% | 21,2% | 28,9% |
| 2 | 9,6% | 0,0% | 19,6% | 37,2% | 33,5% |
| 3 | 6,2% | 0,0% | 3,0% | 59,3% | 31,5% |
| 4 | 2,7% | 0,0% | 1,0% | 9,9% | 86,4% |

accuracy: 63,6 %

| | gewichtete Vorhersage | | | | |
|---|---|---|---|---|---|
| Ziel | 0 | 1 | 2 | 3 | 4 |
| 0 | 283680 | 55 | 0 | -17550 | -76350 |
| 1 | 0 | 5720 | 860 | 0 | -10535 |
| 2 | -9260 | 0 | 37660 | 17860 | 0 |
| 3 | -19260 | -20 | 0 | 185100 | 24615 |
| 4 | -54120 | -20 | -4950 | 0 | 869440 |

Punkte gesamt: 1 232 925 (56,8%)

**Tabelle 4.8:** bester Baum des Tournament mit der Attributauswahl der schrittweisen Diskriminanzanalyse

binären Klassifizierer mit der gleichen Attributauswahl erstellt wurden. Ein Vergleich aller verwendeten Attributauswahlverfahren mit dem Tournament ist in der Abbildung 4.4 zu sehen.

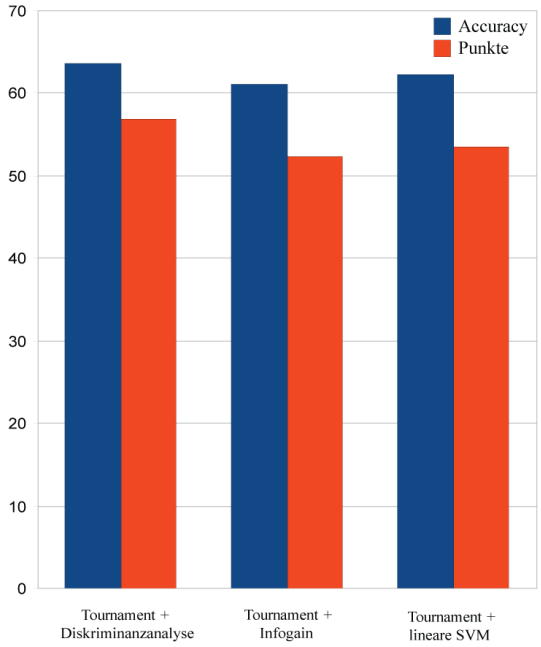



**Abbildung 4.4:** der nach den Punkten beste Baum für das Tournament

# KAPITEL 5

## Fazit und Ausblick

Datamining ist ein vielschichtiger Prozess, der aus mehreren Schritten besteht. Für jeden Schritt stehen verschiedenen Methoden zur Verfügung, die je nach spezieller Anforderung der Aufgabenstellung ausgewählt werden müssen. Ich habe versucht, durch Experimente die optimalen Methoden zur Dimensionsreduktion und für die Erstellung der Klassifizierer zu finden. Die Experimente zeigten, dass man die Geschwindigkeit der Klassifizierer durch geeignete Dimensionsreduktionsverfahren verbesser kann und lediglich eine geringe Verschlechterung der Trennung in Kauf nehmen muss. Die Verfahren Hauptkomponentenanalyse und Autoencoder berücksichtigen die Zielklassen nicht. Sie haben sich für das vorliegende Mehrklassenproblem als ungeeignet herausgestellt. Dimensionsreduktionsverfahren, die Klassifizierer für die Attributauswahl benutzen, erzielten in meinen Experimenten deutlich bessere Ergebnisse, da sie auch die Zielklasen berücksichtigen. Die beste Dimensionsreduktion mit der geringsten Fehlerrate wurde durch die Attributauswahl der schrittweisen Diskriminanzanalyse erreicht. In den Experimenten zum Tournament erstellte ich für jeden binären Klassifizierer eine angepasste Attributauswahl. Damit konnte ich jedoch keine Verbesserung erreichen.

Der zweite wichtige Aspekt bei den Experimenten zielte auf die Verbesserung der Klassifizierungsergebnisse. Für die ersten Versuche verwendete ich die Diskriminanzanalyse, die sich als gutes Verfahren für ein Mehrklassenproblem herausstellte. Jedoch bot diese Methode kaum Möglichkeiten, die Klassifizierungsergebnisse über einen gewissen Punkt hinaus zu beeinflussen. Die Support Vector Machine (SVM) ist ein sehr leistungsfähiger Klassifizierer, der jedoch nur zwei Zielklassen voneinander trennen kann. Die von mir verwendete Bibliothek LibSVM versucht, das Mehrklassenproblem mit Hilfe der „Einer genen Einen"-Strategie zu lösen. Die Ergebnisse des Verfahrens zeigten jedoch, dass es für meine konkretes Problem nich geeignet war. Mit Hilfe des Tournament konnte ich deutlich bessere Ergbenisse erziehlen. Jedoch stieg der Aufwand beim Trainieren der Klassifizierer mit allen binären Kombinationen der Zielklassen exponetiel in Abhängigkeit zur Anzahl der Zielklassen. Bei fünf

Zielklassen mussten schon 70 binäre SVM trainiert werden. Im Vergleich dazu mussten bei der „Einer gegen Einen"-Strategie, der LibSVM, nur 10 binäre Klassifizierer erzeugt werden. Das Problem der Suche des optimalen Binärbaumes für das Tournament ist NP-vollständig. Das bedeuted, dass alle Binärbäume (in meinem Fall 120 Stück) erstellt werden müssen, um den optimalen Baum zu finden. Um den Aufwand der Suche zu reduzieren, könnte man die Auswahl der Knoten eines Baumes durch ein geeignetes Kriterium erleichtern. Ein solches Kriterium wäre in diesem Fall das mit der Kostenmatrix gewichtete Klassifzierungsergebnis, das anschließend aufsummiert wird. Beim Erzeugen eines guten Baumes wird zuerst der Wurzelnknoten ausgewählt, der alle fünf Zielklassen enthält und das Kriterium maximiert. Bei der Auswahl der Kindknoten werden ebenfalls die ausgewählt, die die verbleibenden Zielklassen trennen und das Kriterium maximieren. Dieses Verfahren wrd rekursiv solange durchgeführt, bis alle Zielklassen voneinander getrennt wurden. Den kostensensitven Aspekt der Aufgabenstellung habe ich mit der Suche nach dem besten Baum berücksichtigt. So wurde der Baum ausgewählt, der die Punkte maximiert, also die Kosten minimiert hat. Eine Optimierungsmöglichkeit läge zusätzlich darin, auch die Entscheidung der einzelnen Knoten kostensensitiv zu gestalten [Esm07].

Bei den Klassifizierern benutzte ich hauptsächlich die Diskriminanzanalyse und die Support Vector Machine. Bei der SVM testete ich zwei verschiedene Verfahren zum lösen des Mehrklassenproblemes, die „Einer gegen Einen"-Strategie und das Tournament. Die SVM wurde mit unterschiedlichen Dimensionsredunktionsverfahren kombiniert. Die Abbildungen 5.1, 5.2 und 5.3 zeigen die verwendeten Verfahren. Ein direkter Vergleich aller Verfahren ist nicht ohne weiteres möglich, da sie mit unterschiedlichen Validierungsmaßen entstanden sind. Bei den ersten Experimenten verwendete ich eine 10-fache Kreuzvalidierung, bei der Validierung der verschiedenen Dimensionsreduktionsverfahren nutzte ich 90% der Datensätze zum Trainieren und die restlichen Datensätze zum Testen des Klassifizierers. Beim Erstellen der binären SVM verwendete ich die kompletten Teildatensätze zum Trainieren und testete die Bäume mit dem kompletten Datensatz.

## 5.1 Ergebnisse des DataMiningCup 2008

Drei Wochen nach Einsendung der Ergebnisse wurden die Ergebnisse des DataMiningCup 2008 veröffentlicht.[1] Für die Klassifizierung der 113 476 Datensätze aus der Testmenge habe ich 865 565 Punkte erhalten. Damit belegte ich bei 212 Teilnehmern den 100-sten Platz. Meine Ergebnisse lagen über dem Median von 844 160 Punkten. Der beste Teilnehmer erreichte 1 030 240 Punkte, der schlechteste −1 494 315.

1 http://www.prudsys.de/Service/Downloads/files/Rankingliste_Studenten_dt.pdf

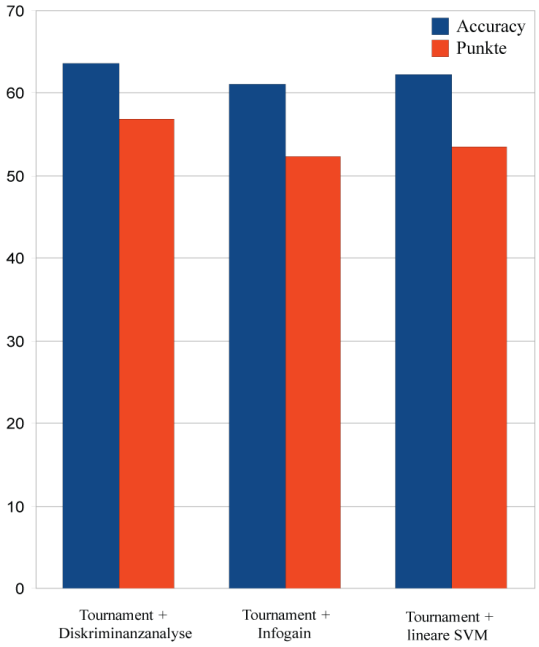



**Abbildung 5.1:** der nach den Punkten beste Baum für das Tournament

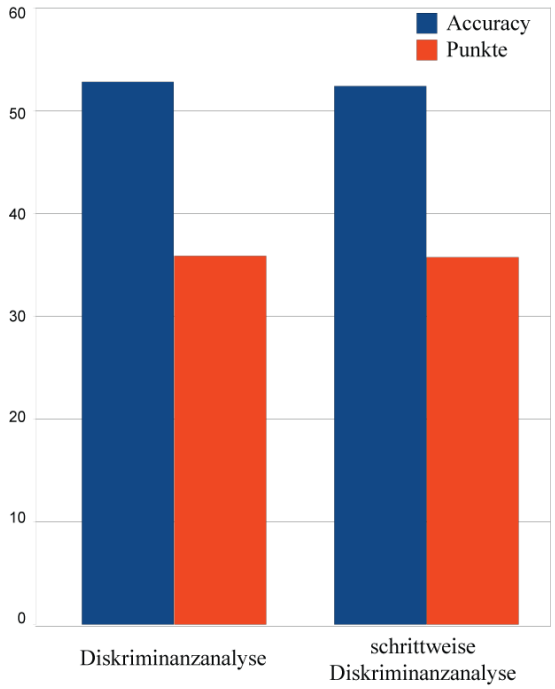



**Abbildung 5.2:** 10-fach Kreuzvalidierte Klassifizierer

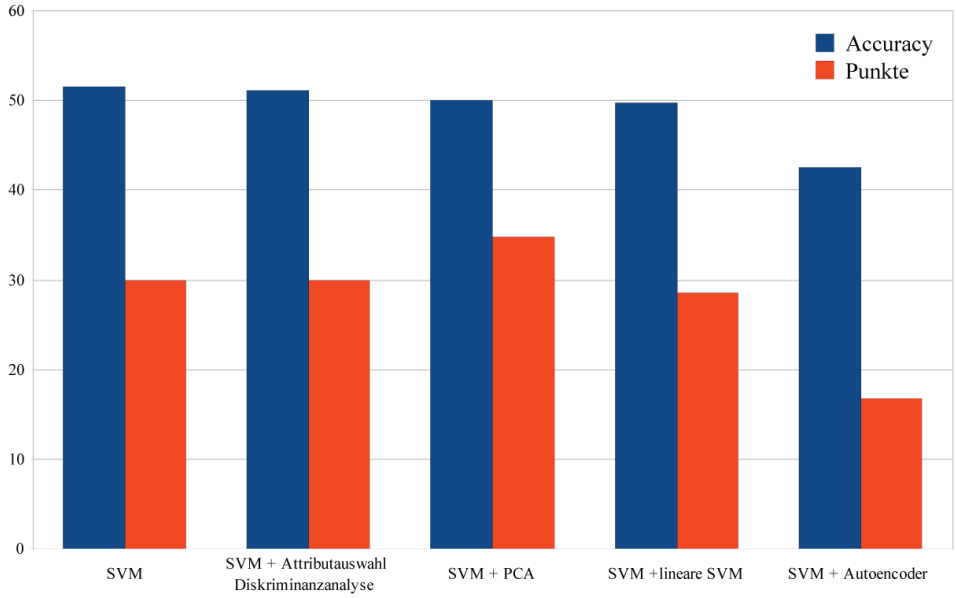



**Abbildung 5.3:** Klassifizierer, die mit 90% der Datensätze trainiert und mit den restlichen 10% getestet wurden

| | Vorhersage | | | | |
|---|---|---|---|---|---|
| Ziel | 0 | 1 | 2 | 3 | 4 |
| 0 | 55,0% | 0,0% | 1,8% | 13,6% | 29,6% |
| 1 | 43,6% | 3,9% | 2,4% | 21,2% | 28,9% |
| 2 | 9,6% | 0,0% | 19,6% | 37,2% | 33,5% |
| 3 | 6,2% | 0,0% | 3,0% | 59,3% | 31,5% |
| 4 | 2,7% | 0,0% | 1,0% | 9,9% | 86,4% |
| accuracy: 63,6 % | | | | | |

| | gewichtete Vorhersage | | | | |
|---|---|---|---|---|---|
| Ziel | 0 | 1 | 2 | 3 | 4 |
| 0 | 283680 | 55 | 0 | -17550 | -76350 |
| 1 | 0 | 5720 | 860 | 0 | -10535 |
| 2 | -9260 | 0 | 37660 | 17860 | 0 |
| 3 | -19260 | -20 | 0 | 185100 | 24615 |
| 4 | -54120 | -20 | -4950 | 0 | 869440 |
| Punkte gesammt: 1 232 925 (56,8%) | | | | | |

**Tabelle 5.1:** SVM mit der Attributauswahl der schrittweisen Diskriminanzanalyse auf den Trainigsdaten

In den Tabellen 5.1 ist das Ergebnis des Tournaments zu sehen, das ich für die Klassifizierung der am Ende eingereichten Ergebnisse verwendet habe. Die linke Tabelle zeigt die Konfusionsmatrix, die beim Testen mit allen Trainingsdaten entstanden ist. Der Baum für dieses Tournament ist in Abbildung 4.3 zu sehen und enthält die Attributauswahl der schrittweisen Diskriminanzanalyse. Auf Nachfrage beim Veranstallter des DataminingCup erhielt ich die richtigen Zielklassen des Testdatensatzes. Mit diesen evaluierte ich die von mir abgegebenen Ergebnisse. Die Tabelle 5.2 zeigt nun die Konfusionsmatrix der Testdaten.

Es ist zu erkennen, dass zwischen den Klassifiaktionsergebnissen der Trainings- und Testdaten deutliche Unterschiede liegen. So sank die Accuracy von 63,6% auf 51,5%. Wenn ein Klassifizierer mit den Trainingsdaten bessere Ergebnisse erzielt als

| | Vorhersage | | | | |
|---|---|---|---|---|---|
| Ziel | 0 | 1 | 2 | 3 | 4 |
| 0 | 37,2% | 0,1% | 3,5% | 18,7% | 40,5% |
| 1 | 29,0% | 0,2% | 3,8% | 26,3% | 40,7% |
| 2 | 19,4% | 0,1% | 4,6% | 31,1% | 44,8% |
| 3 | 13,5% | 0,1% | 3,3% | 40,2% | 43,0% |
| 4 | 6,2% | 0,0% | 1,5% | 13,7% | 78,5% |
| accuracy: 51,5 % | | | | | |

| | gewichtete Vorhersage | | | | |
|---|---|---|---|---|---|
| Ziel | 0 | 1 | 2 | 3 | 4 |
| 0 | 203080 | 80 | 0 | -25590 | -110700 |
| 1 | 0 | 320 | 1430 | 0 | -15250 |
| 2 | -19610 | 0 | 9200 | 15695 | 0 |
| 3 | -42860 | -110 | 0 | 128160 | 34220 |
| 4 | -131360 | -120 | -8000 | 0 | 826980 |
| Punkte gesammt: 865 565 (38,1%) | | | | | |

**Tabelle 5.2:** SVM mit der Attributauswahl der schrittweisen Diskriminanzanalyse auf den Testdaten

mit den Testdaten, ist davon auszugehen, dass er überangepasste Hypothesen erstellt und schlecht generalisiert hat. Neben bereits genannten kostensensitiven Verfahren, liegen weitere Optimierungsmöglichkeiten in der Verwendung von anderen Kernelfunktionen für die SVM oder anderen Methoden für die paarweise Klassifizierung. Es gibt neben den beschriebenen noch viele andere Methoden im Bereich des Dataminig, die im begrenzten zeitlichen Rahmen des Wettbewerbes nicht behandelt werden konnten. Ich habe jedoch versucht mit der Diskriminanzanalyse und der Support Vector Machine zwei der vermutlich wichtigsten Klassifizierer auszuwählen und mit zahlreichen Dimensionsreduktionsverfahren zu kombinieren.

# Literaturverzeichnis


[Bi03] Bi, Jinbo; Bennett, Kristin; Embrechts, Mark; Breneman, Curt und Song, Minghu: Dimensionality reduction via sparse support vector machines. *Journal of Machine Learning Research* (2003), Bd. 3:S. 1229–1243

[Bro98] Brosius, Felix: *SPSS 8: professionelle Statistik unter Windows*, MIT Press (1998), URL http://www.wiwi.uni-tuebingen.de/cms/fileadmin/Uploads/Lehrstuehle/Prof._Grammig/Lehre_Alte_Homepage_S-Z/SPSS_Diskriminanzanalyse.pdf, [Online: Stand 12.12.2009]

[Cha01] Chang, Chih-Chung und Lin, Chih-Jen: *LIBSVM: a library for support vector machines* (2001), URL http://www.csie.ntu.edu.tw/~cjlin/papers/libsvm.pdf, [Online: Stand 12.12.2009]

[Che06] Chen, Y. W. und Lin, C. J.: *Combining SVMs with various feature selection strategies*, Springer (2006)

[Esm07] Esmeir, Saher und Markovitch, Shaul: Anytime Induction of Cost-sensitive Trees (2007), URL http://www.cs.technion.ac.il/~shaulm/papers/pdf/Esmeir-Markovitch-NIPS2007.pdf

[Fay96] Fayyad, Usama; Piatetsky-Shapiro, Gregory und Smyth, Padhraic: From data mining to knowledge discovery in databases. *AI Magazine* (1996), Bd. 17:S. 37–54, URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.42.1071, [Online: Stand 12.12.2009]

[Guy02] Guyon, Isabelle; Weston, Jason; Barnhill, Stephen und Vapnik, Vladimir: Gene Selection for Cancer Classification using Support Vector Machines. *Mach. Learn.* (2002), Bd. 46(1-3):S. 389–422

[Guy03] Guyon, Isabelle und Elisseeff, André: An Introduction to Variable and Feature Selection. *Journal of Machine Learning Research* (2003), Bd. 3:S. 1157–1182, URL http://www.jmlr.org/papers/volume3/guyon03a/guyon03a.pdf, [Online: Stand 12.12.2009]

[Hin06] Hinton, G. E. und Salakhutdinov, R. R.: Reducing the Dimensionality of Data with Neural Networks. *Science* (2006), Bd. 313(5786):S. 504–507

[Jol02] Jolliffe, I. T.: *Principal Component Analysis*, Springer, second Aufl. (2002)

[Kre99] Kressel, H.-G. Ulrich: *Pairwise classification and support vector machines*, Springer (1999), S. 255–352

[Ng] Ng, Andrew: Part V: Support Vector Machine, URL http://www.stanford.edu/class/cs229/notes/cs229-notes3.pdf, [Online: Stand 12.12.2009]

[Reu03] Reunanen, Juha: Overfitting in making comparisons between variable selection methods. *J. Mach. Learn. Res.* (2003), Bd. 3:S. 1371–1382

[Sch] Scholz, Matthias: Nichtlineare Hauptkomponentenanalyse auf Basis neuronaler Netze, URL http://edoc.hu-berlin.de/docviews/abstract.php?id=28770, [Online: Stand 12.12.2009]

[Sch01] Schölkopf, Bernhard und Smola, Alexander J.: *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*, MIT Press, Cambridge, MA, USA (2001)

[SPS05] SPSS Inc.: *SPSS 14.0 Base Benutzerhandbuch*, SPSS Inc. (2005), URL http://support.spss.com/productsext/spss/documentation/spssforwindows/German/SPSS%20Base%20Users%20Guide%2014.0.pdf, [Online: Stand 12.12.2009]

[Ste08] Stein, Benno und Lettmann, Theodor: Maschinelles Lernen und Data Mining (2008), URL http://www.uni-weimar.de/cms/medien/webis/teaching/lecture-notes.html#machine-learning, [Online: Stand 12.12.2009]

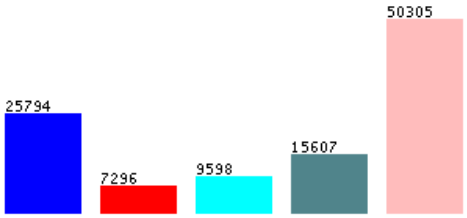
[Wit02] Witten, Ian H. und Frank, Eibe: Data mining: practical machine learning tools and techniques with Java implementations. *SIGMOD Rec.* (2002), Bd. 31(1):S. 76–77

# ANHANG A

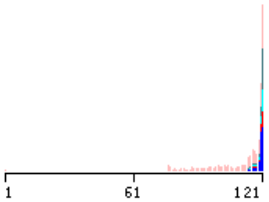
## Grafiken

### Histogramme der besten 20 Attribute

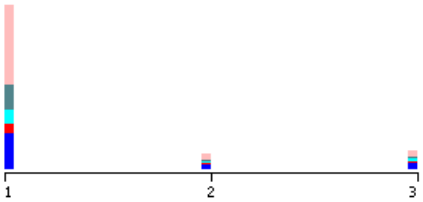
Im folgenden sind die Histogramme der Attribute aufgelistet, die die schrittweise Diskriminanzanalyse für alle Zielklassen ausgewählt hat.



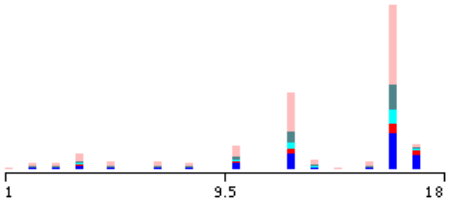
Target



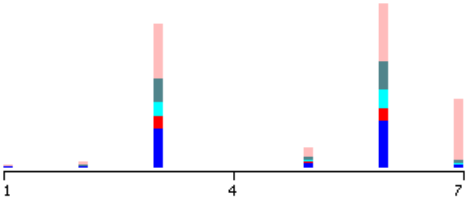
Aufnahme Lotterie



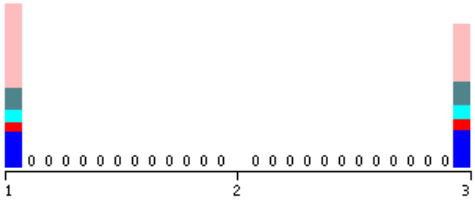
tel art



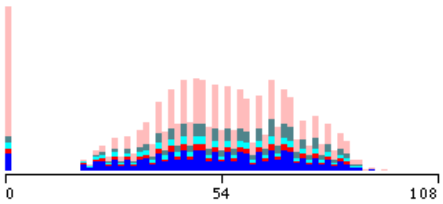
bankart



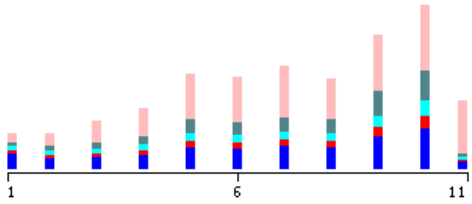
werbeweg



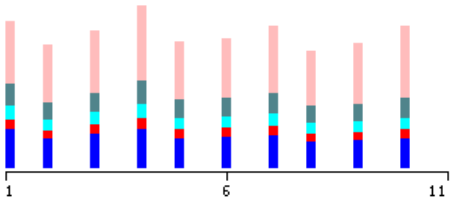
geschlecht



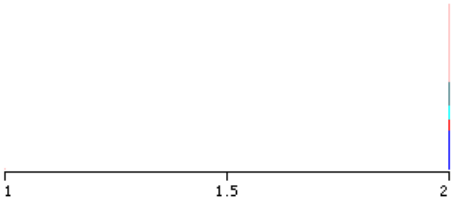

kunden alter



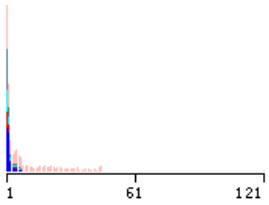

altersdeziel



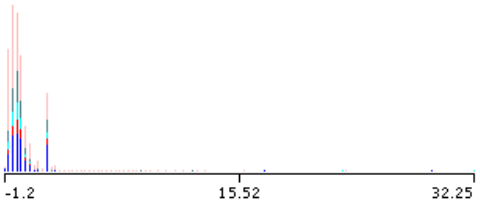

kaufkraftdeziel



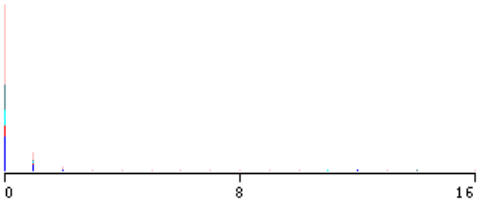

serviceweg auf



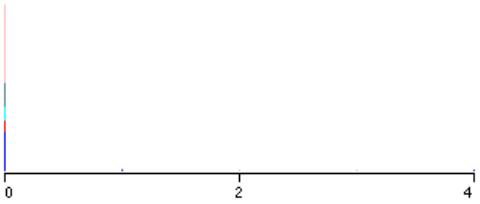

anz gespielte lotterien



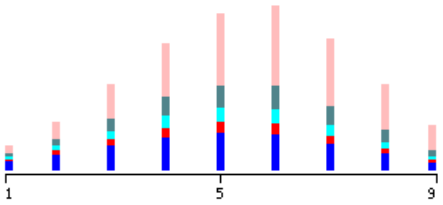

zbest vol t1



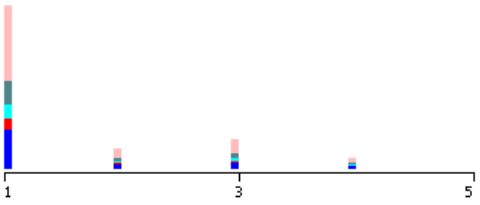

kki info lotterie vorlot



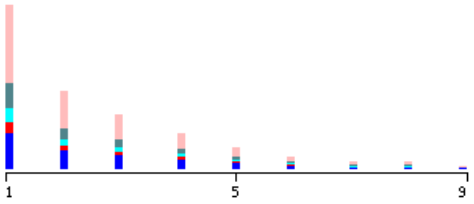

kki reklamation lott vorlot



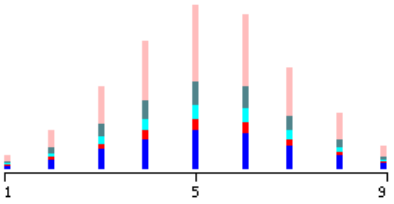

mtstag



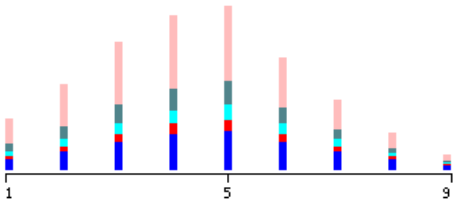

mtstrg



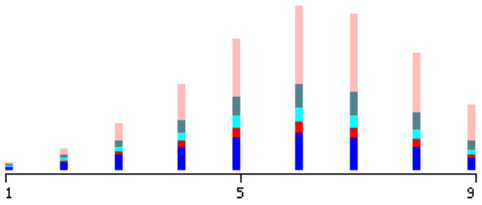

mtausg



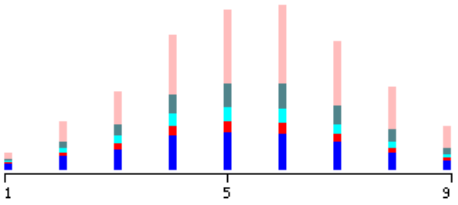

mtpozg




msbrkg




mtpzvg




mtkdig

## Dokumente zum DataminingCup 2008

### Erklärung zu den Attributen

# DATA MINING CUP 2008 - description of features

| Feature | Type | Description | Attributes |
|---|---|---|---|
| id | integer | Unique customer ID | |
| target | integer | Target feature | 0 The first ticket has not been paid for.<br>1 Only the ticket for the first class has been paid for.<br>2 Only the first two classes were played.<br>3 The lottery was played until the end but no ticket was purchased for the following lottery 122.<br>4 At least the first ticket for lottery 122 was played as well. |
| Neukunde | integer | New customer | 0 regular customer / 1 new customer |
| AUFNAHME_LOTTERIE | integer | Lottery, the customer first entered the game | range 1-121 |
| TEL_ART | integer | Category of telephone | 1 fixed line<br>2 n/a<br>3 cell phone |
| BANKART | integer | Category of bank | 1 BBBank<br>2 Citibank<br>3 Commerzbank<br>4 Deutsche Bank<br>5 Dresdner Bank<br>6 Dt. Apotheker- und Ärztebank<br>7 Hypovereinsbank<br>8 Landesbank<br>9 Norisbank<br>10 Postbank<br>11 PSD Bank<br>12 Raiffeisen- und VB<br>13 Other<br>14 SEB<br>15 Spardabank<br>16 Sparkasse<br>17 not defined/unknown (user defined)<br>18 not defined/unknown (system defined) |
| WERBEWEG | integer | Means by which customer was acquired | 1 Supplements<br>2 Internet<br>3 Direct Mailing<br>4 Bulk mail<br>5 Other<br>6 Telephone marketing<br>7 not defined/unknown (system defined) |
| GESCHLECHT | integer | Gender of customer | 1 male<br>2 female |
| KUNDEN_ALTER | integer | Age of customer | in years; range 18 to over 100 |
| ALTERSDEZIL | integer | Decile of customer age | range 1 (low) to 10 (high)<br>1 18 to 28 years<br>2 28 to 33 years<br>3 33 to 38 years<br>4 38 to 42 years<br>5 42 to 47 years<br>6 47 to 52 years<br>7 52 to 58 years<br>8 58 to 64 years<br>9 64 to 72 years<br>10 72 to 130 years<br>unknown |
| KAUFKRAFTDEZIL | integer | Decile of purchasing power | range 1 (low) to 10 (high)<br>1 29 to 82<br>2 82 to 89<br>3 89 to 93<br>4 93 to 97<br>5 97 to 100<br>6 100 to 103<br>7 103 to 107<br>8 107 to 111<br>9 111 to 118<br>10 118 to 267<br>unknown |
| ERS_LOS_AUFN | integer | Automatic replacement of tickets dropped out of the lottery | 1 always<br>2 unknown<br>3 never |
| SERVICEWEG_AUF | integer | Preferred communication channel | 1 e-mail<br>2 regular mail |
| ZAHLUNGSWEG_AUF | integer | Method of payment | 1 automatic debit transfer<br>2 bank transfer by customer |
| ANZ_GESPIELTE_LOTTERIEN | integer | Number of lotteries played | range 1 to 121 |
| ZBEST_VOL_t1 | float | Z-standardized amount of tickets ordered in the first class | |

# DATA MINING CUP 2008 - description of features

| Feature | Type | Description | Attributes |
|---|---|---|---|
| KKI_INFO_LOTTERIE_VORLOT | integer | Number of customer-initiated contacts before lottery.<br>Purpose: demanding information | greater or equal 0 |
| KKI_REKLAMATION_LOTT_VORLOT | integer | Number of customer-initiated contacts before lottery.<br>Purpose: complaint | greater or equal 0 |
| HGEW | integer | Number of businesses in the building | greater or equal 0 |
| MTKAUG | integer | Purchasing power - group value | -9 very low to 9 very high |
| MTSTAG | integer | Status - group value | 1 low to 9 high |
| MTSTRG | integer | Type of street | 1 Residential street<br>2 Shopping street<br>3 Business present<br>4 Business street<br>5 Street used an extreme amount for business |
| MTBEBG | integer | Type of building | 0 no building type assigned<br>1 Single and two-family dwellings<br>2 Three to five-family dwellings<br>3 Buildings containing 6 to 9 households<br>4 Buildings containing 10 to 19 households<br>5 Buildings containing 20 or more households |
| MTAUSG | integer | Immigrant index - group value | 1 low to 9 high |
| MTALTG | integer | Age distribution - group value | 1 younger to 8 older |
| MTAJUG | integer | Age distribution < 30 years of age - group value | 1 low to 9 high |
| MTAALG | integer | Age distribution >= 60 years of age - group value | 1 low to 9 high |
| MTFAMG | integer | Family structure - group value | 1 single/unmarried couples to<br>9 families with children |
| MTMMOG | integer | Mobility - group value | 1 low to 9 high |
| MTBONG | integer | Degree of inspection credit-worthiness - group value | 1 good to 9 bad |
| MTPOZG | integer | Type of media politics - group value | 1 low to 9 high |
| MTWIZG | integer | Type of media business - group value | 1 low to 9 high |
| MTTING | integer | Internet use - group value | 1 low to 9 high |
| MSBKRG | integer | Affinity credit cards - group value | 1 low to 9 high |
| MSBRKG | integer | Affinity to deferred payment credit | 1 low to 9 high |
| klv_scr | integer | Affinity to permanent insurance (KLV) | 1 maximum aff. to 6 minimum affinity |
| tec_scr | integer | Affinity to communication technology | 1 maximum aff. to 6 minimum affinity |
| ifi_scr | integer | Affinity mortgaging | 1 maximum aff. to 6 minimum affinity |
| eih_scr | integer | Affinity home ownership (own house) | 1 maximum aff. to 6 minimum affinity |
| eiw_scr | integer | Affinity home ownership (own flat) | 1 maximum aff. to 6 minimum affinity |
| mtpvvg | integer | Private health insurance | 1 low to 9 high |
| mtpzvg | integer | Private supplemental health insurance | 1 low to 9 high |
| MTKDIG | integer | Car density - group value | 1 low to 9 high |
| MTKLEG | integer | Car-performance - group value | 1 low to 9 high |
| MTKKLG | integer | Car-estate - group value | 1 low to 9 high |
| MTKGBG | integer | Car-used - group value | 1 low to 9 high |
| MTKGLG | integer | Car-SUV - group value | 0 none<br>1 not many present<br>to<br>5 more frequently present |
| MTKPRG | integer | Premium class - group value | 0 none<br>1 individually present<br>to<br>5 considerably increased share |
| MTKSPG | integer | Premium sport car - group value | 0 none<br>1 individually present<br>to<br>5 considerably increased share |
| MTKJAG | integer | "Asian" - group value | 0 none<br>1 not many present<br>to<br>5 more frequently present |
| MTKDSG | integer | Diesel - group value | 1 low to 9 high |
| MTKGEG | integer | Company cars - group value | 0 none<br>1 not many present<br>to<br>5 more frequently present |
| MTKEMG | integer | Low emission value - group value | 1 low to 9 high |
| MTKOEG | integer | Alternative drive - group value | 0 none<br>1 not many present<br>to<br>5 more frequently present |

# DATA MINING CUP 2008 - description of features

| Feature | Type | Description | Attributes |
|---|---|---|---|
| MTKK1K | integer | Brand priority most frequent group | 0 "No cars in the cell"<br>1 "Audi"<br>2 "BMW"<br>3 "Ford"<br>4 "Mercedes Benz"<br>5 "Opel/General Motors"<br>6 "VW/Seat"<br>7 "Fiat/Alfa/Lancia"<br>8 "Peugeot/Citroen"<br>9 "Renault"<br>10 "Saab/Volvo"<br>11 "Nissan"<br>12 "Toyota"<br>13 "Daewoo, Daihatsu, Honda, Hyundai, KIA, Mazda, Mitsubishi, Proton, Subaru, Suzuki, Ssangyong"<br>14 "Porsche/Ferrari/Jaguar"<br>15 "other" |
| MTK01G | integer | Share Audi - group value | 1 low to 9 high |
| MTK02G | integer | Share BMW - group value | 1 low to 9 high |
| MTK03G | integer | Share Ford - group value | 1 low to 9 high |
| MTK04G | integer | Share Mercedes-Benz - group value | 1 low to 9 high |
| MTK05G | integer | Share Opel/General Motors - group value | 1 low to 9 high |
| MTK06G | integer | Share VW/Seat - group value | 1 low to 9 high |
| MTK07G | integer | Share Fiat/Alfa Romeo/Lancia - group value | 1 low to 9 high |
| MTK08G | integer | Share Peugeot/Citroen - group value | 1 low to 9 high |
| MTK09G | integer | Share Renault - group value | 1 low to 9 high |
| MTK10G | integer | Share Saab/Volvo - group value | 1 low to 9 high |
| MTK11G | integer | Share Nissan - group value | 1 low to 9 high |
| MTK12G | integer | Share Toyota - group value | 1 low to 9 high |
| MTK13G | integer | Share Daewoo, Daihatsu, Honda, Hyundai, KIA, Mazda, Mitsubishi, Proton, Subaru, Suzuki, Ssangyong - group value | 1 low to 9 high |
| MTK14G | integer | Share Porsche/Ferrari/Jaguar - group value | 1 low to 9 high |
| ablnbl | integer | Old / New Federal States | 98 Old Federal States<br>99 New Federal States |
| regio | integer | Type of region - 9 classifications | 12 Statistical area - High order centres<br>13 Statistical area - Middle order centres<br>14 Statistical area - Catchment area<br>22 Urbanised area - High order centres<br>23 Urbanised area - Middle order centres<br>24 Urbanised area - Catchment area<br>32 Rural area - High order centres<br>33 Rural area - Middle order centres<br>34 Rural area - Catchment area |

# Abbildungsverzeichnis


2.1 Unreinheit(Impurity) einer Trennung durch Attributauswahl (Quelle: [Ste08, Unit. Splitting]) . . . . . 6
2.2 Vielzahl von möglichen Trennebenen . . . . . 11
2.3 geometrische Abstand $\gamma$ eines Punktes $\mathbf{x}$ zur Hyperebene . . . . . 12

3.1 Varianzen in der Diskriminanzfunktion . . . . . 17
3.2 Nichtlineare SVM mit dem Kerneltrick (Quelle: [Sch01]) . . . . . 19
3.3 Links: separierbarer Fall, Rechts: Datensatz mit Ausreißern . . . . . 21
3.4 Links: „Einer gegen den Rest", Rechts: „Einer gegen Einen"-Strategie, Gestrichtelten linien sind die binären Klassifizierer, die roten Linien ist (Quelle: [Kre99]) . . . . . 23
3.5 Das Tournament benutzt einen Baum mit binären Klassifizierern . . 24

4.1 Histogramm der Zielklassenverteilung . . . . . 27
4.2 Lernraten der beiden Autoencoder. . . . . 35
4.3 der nach den Punkten beste Baum für das Tournament . . . . . 36
4.4 der nach den Punkten beste Baum für das Tournament . . . . . 37

5.1 der nach den Punkten beste Baum für das Tournament . . . . . 40
5.2 10-fach Kreuzvalidierte Klassifizierer . . . . . 40
5.3 Klassifizierer, die mit 90% der Datensätze trainiert und mit den restlichen 10% getestet wurden . . . . . 41

# Tabellenverzeichnis


4.1 Kostenmatrix des DataMiningCup 2008 . . . . . 28
4.2 Die ersten Ergebnisse mit Der SPSS-Diskriminanzanalyse. Links . . 29
4.3 Die SVM mit allen Attributen . . . . . 30
4.4 SPSS-Diskriminanzanalyse links: ohne Attributauswahl, rechts: mit Attributauswahl . . . . . 32
4.5 SVM Links: ohne Attributauswahl, Rechts: mit Attributauswahl der schrittweisen Diskriminanzanalyse . . . . . 32
4.6 Die SVM mit den Features der Hauptkomponentenanalyse . . . . . 33
4.7 Links: Mehrklassen SVM der LibSVM, Rechts: binäre SVM . . . . . 35
4.8 bester Baum des Tournament mit der Attributauswahl der schrittweisen Diskriminanzanalyse . . . . . 37

5.1 SVM mit der Attributauswahl der schrittweisen Diskriminanzanalyse auf den Trainigsdaten . . . . . 41
5.2 SVM mit der Attributauswahl der schrittweisen Diskriminanzanalyse auf den Testdaten . . . . . 42