Rheinisch Westfälische Technische Hochschule Aachen
Lehrstuhl für Informatik VI
Prof. Dr.-Ing. Hermann Ney

Seminar „Data Mining and Learning from Data“

Predictive Modeling

Thorsten Holz

Matrikelnummer 226 669

Wintersemester 2003/2004

Betreuer: Daniel Keysers

Inhaltsverzeichnis

We are drowning in information
and starving for knowledge.
- Rutherford D. Roger

1 Einleitung

Durch die Fortschritte in der Computertechnologie wird es immer einfacher, große Mengen von Daten zu produzieren und diese in Datenbanken abzuspeichern: Das Einkaufsverhalten von Kunden wird aufgezeichnet, Mobilfunk-Konzerne speichern die Verbindungsdaten, Ärzte die Untersuchungsergebnisse, der Kursverlauf der Aktien an den Börsen wird festgehalten, Teleskope sammeln Daten über entfernte Objekte,... Diese Liste ließe sich beliebig lange fortführen. Auch das World Wide Web wird immer größer und zu einem „Informationsspeicher“ der Welt. Doch das Erfassen und Speichern der Daten nützt wenig, die Daten müssen *verstanden* werden, um daraus Schlüsse ziehen und Entscheidungen treffen zu können.
Genau in diesem Punkt setzt Data-Mining an [HMS01]:

```
Data-Mining is the analysis of (often large) observational data sets to find
unsuspected relationships and to summarize the data in novel ways that are
both understandable and useful to the data owner.
```

Aufgabe des Data-Mining ist es also vor allem, Beziehungen innerhalb einer großen Menge von Daten zu finden sowie eine verständliche und nützliche Zusammenfassung dieser Daten in verschiedene Klassen zu liefern. Dazu gehört auch die nicht-triviale Vorhersage von Klassenzugehörigkeiten bei neuen Datensätzen. Mit diesem Thema – im Englischen *predictive modeling* genannt – beschäftigt sich diese Seminararbeit und stellt einige Verfahren zur Vorhersage sowie Möglichkeiten zur Bewertung verschiedener Modelle vor.

Doch was genau ist *predictive modeling*? Mittels des *predictive modeling* möchte man anhand einer gegebenen Datenmenge Vorhersagen über zukünftige Werte der Daten machen. Dazu versucht man, durch vorgegebene Eigenschaften $\mathbf{x}$ (*predictor variable* oder *Prädiktor/Merkmale* genannt) ein Modell – also eine allgemeine Beschreibung einer Menge von Daten – für eine Zielvariable y (engl.: *response variable*) zu finden. Man möchte also eine Funktion der Form

$$y = f(\mathbf{x}, \Theta)$$

finden, wobei Θ die Parameter des Modells bezeichnet.

Grob gesehen kann man *predictive modeling* in zwei Bereiche aufteilen: Zum einen in Klassifikation (engl.: *classification*) und zum anderen in Regression (engl.: *regression*). Der Unterschied zwischen diesen Bereichen liegt im Wertebereich der Zielvariablen: Bei der Klassifikation ist die Zielvariable diskret oder stammt aus einer bestimmten Kategorie (beispielsweise {gesund, krank} oder {hoch, mittel, niedrig}). In diesem Fall muss auf der Zielvariablen also keine Ordnung definiert sein. Im Gegensatz dazu steht die Regression, bei der die Zielvariable reellwertig ist.

Die weiteren Unterschiede zwischen diesen beiden Arten des *predictive modeling* werden in den nächsten Abschnitten erläutert: Abschnitt 2 erläutert die mathematischen und begrifflichen Grundlagen der Klassifikation und Regression sowie den Kompromiss zwischen Bias und Varianz (*bias-variance trade-off*), ein in der Praxis immer wieder auftretendes Problem. Abschnitt 3 erläutert verschiedene Modelle zum *predictive modeling*, beispielsweise die logistische Diskriminanzanalyse oder die Nächste-Nachbarn-Methode. Verfahren, mit deren Hilfe man die Leistungsfähigkeit von solchen Modellen abschätzen und bewerten kann, werden in Abschnitt 4 vorgestellt. Schließlich fasst Abschnitt 5 die vorgestellten Ergebnisse zusammen.

2 Grundbegriffe des Predictive Modeling

In diesem Abschnitt sollen die Terminologie sowie die mathematischen Grundlagen der Vorhersage von Daten vorgestellt werden. Dies soll zunächst anhand von einigen Beispielen zur Klassifikation beziehungsweise Regression geschehen:

2.1 Klassifikation und Regression

Ein Beispiel für Klassifikation ist die Beurteilung der Kreditwürdigkeit von Bankkunden: Mit Hilfe verschiedener Variablen $\mathbf{x} = (x_1, x_2, \ldots, x_D)^T$ wie beispielsweise „monatliches Einkommen/Verpflichtungen“, „ausgeübter Beruf“, „Familienstand“ oder „Höhe der Gesamtschulden“ wird bei einem Kredit-Antrag darüber entschieden, diesem Kunden einen Kredit zu gewähren oder den Antrag zurückzuweisen. Den Vektor aller dieser Merkmale nennt man *Prädiktor* und bezeichnet ihn mit $\mathbf{x} = (x_1, x_2, \ldots, x_D)^T$. Mittels dieses Prädiktors soll nun der Wert der *Zielvariablen* y vorhergesagt werden, es soll also eine Funktion der Form $y = f(\mathbf{x}, \Theta)$ gefunden werden. Dabei bezeichnet Θ die Parameter des Modells, dessen funktionale Form man im Voraus bestimmt. Die eigentlichen Parameter Θ des Modells *lernt* man während des predictive modeling. Die Zielvariable y hat in diesem Beispiel einen Wert aus der Menge {Kredit gewährt, Kredit abgelehnt}. Diese beiden Klassen können noch verallgemeinert werden, indem mehrere Klassen eingeführt werden: Der Kunde wird dann in eine bestimmte Risikoklasse $c = 1, \ldots, C$ eingeteilt und anhand dieser Klasse wird über die Höhe des gewährten Kredits entschieden. Eine solche Einteilung kann beispielsweise folgendermaßen aussehen: {kein Kredit gewährt, Kredit bis 5.000 Euro gewährt, Kredit zwischen 5.000 Euro und 10.000 Euro gewährt, Kredit über 10.000 Euro gewährt}. Ein einfaches Modell für eine solche Klassifikation stellen *lineare Modelle* dar:

$$\hat{y} = a_0 + \sum_{d=1}^{D} a_d x_d$$

Dabei ist $\hat{y}$ der aufgrund des Modells gewonnene *Schätzer* für y und die $a_d, d = 1, \ldots, D$ sind die Gewichtungen der einzelnen x_d.

Aufgrund des so gewonnenen Schätzers $\hat{y}$ wird nun eine Entscheidung darüber getroffen, zu welcher Klasse c der Prädiktor $\mathbf{x}$ zugeordnet wird. Dazu wird eine sogenannte Entscheidungsgrenze (engl.: *decision boundary*) benutzt, die einen Schwellenwert bei der Zuordnung darstellt. Im Zwei-Klassen-Fall kann dies beispielsweise so aussehen, dass $\mathbf{x}$ genau dann zur Klasse $c = 0$ zugeordnet wird, falls $\hat{y} < 0.5$ ist.

Das oben beschriebene lineare Modell erzeugt eine lineare Entscheidungsgrenze und Abbildung 1 zeigt ein Beispiel für eine solche lineare Entscheidungsgrenze. Die x-Achse stellt die „Höhe der Gesamtschulden“ dar und auf der y-Achse wurde das „monatliche Einkommen“ aufgetragen. Experten haben einige Kunden aufgrund dieser Kritieren zu drei verschiedene Klassen von Risikotypen zugeordnet: Die weißen Kreise stellen jeweils Kunden mit einem niedrigen Einkommen dar; die schwarzen Quadrate haben ein relativ hohes Einkommen und gleichzeitig niedrige Gesamtschulden, während die blauen Kreise zwar auch ein relativ hohes Einkommen, aber auch relativ hohe Gesamtschulden haben. Aus den verschiedenen Klassen ergeben sich für die Bank zwei Entscheidungsgrenzen, die mit schwarzen Linien markiert sind. Die verschiedenen Klassen sind in diesem Beispiel für den unteren Teil {kein Kredit gewährt}, für den rechten oberen Teil {Kredit bis 10.000 Euro gewährt} und für den linken oberen Teil {Kredit über 10.000 Euro gewährt}. Bei der

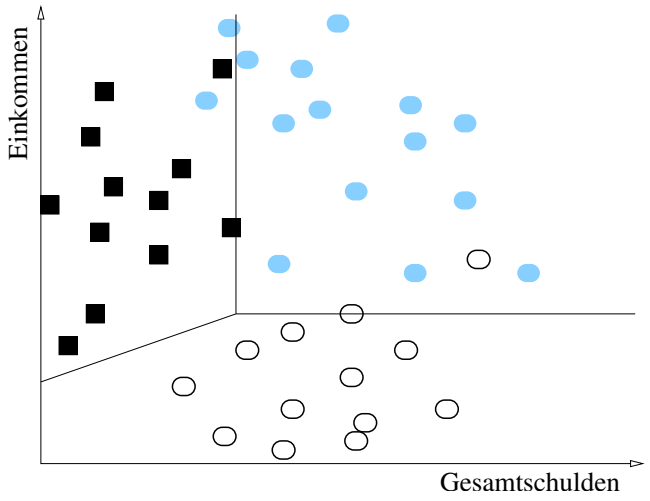

Abbildung 1: Beispiel für lineare Entscheidungsgrenzen

Beurteilung der Kreditwürdigkeit eines Kunden kann nun dessen „Höhe der Gesamtschulden“ sowie „monatliches Einkommen“ in Zusammenhang gestellt und eine Entscheidung getroffen werden. Die Klassifikation liefert immer nur einen Schätzer $\hat{y}$ und dieser ist fehlerbehaftet – in der Abbildung 1 ist dies an den falsch zugeordneten Werten erkennbar.

Im Gegensatz zur Klassifikation steht die Regression, bei der die Zielvariable reellwertig und nicht nur eine Klassen-Kennung ist: Beispielsweise möchte ein Mobilfunk-Konzern anhand der gespeicherten Verbindungsdaten Vorhersagen über das Kundenverhalten zu bestimmten Uhrzeiten machen; oder der Kursverlauf einer Aktie soll mittels verschiedener Kenngrößen und allgemeiner Wirtschaftsdaten vorhergesagt werden; oder ein Energiekonzern möchte anhand der Windgeschwindigkeit, Windrichtung und anderen Merkmalen eine Vorhersage darüber machen, wieviel Energie ein Windrad in einem bestimmten Zeitintervall erzeugt.

Aber auch bei der Regression möchte man eine Funktion der Form $y = f(\mathbf{x}, \Theta)$ erhalten, um Vorhersagen treffen zu können. Doch wie kann man die Parameter Θ einer solchen Funktion bestimmen? Dazu hat man anfangs eine sogenannte *Trainingsmenge* $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)\}$. Bei dieser sind die Prädiktoren $\mathbf{x}_n$, $n = 1, \ldots, N$ sowie die zugehörigen Zielvariablen y_n bekannt – man kennt also für bestimmte Daten die Relation. Anhand dieser Trainingsmenge möchte man nun ein möglichst gutes Modell konstruieren (auf die Bewertung der Leistungsfähigkeit von Modellen wird in Abschnitt 4 noch näher eingegangen), um dann für neue Werte die Zielvariable vorhersagen zu können. Wie bereits angesprochen, erhält man mittels eines Modells einen Schätzer $\hat{y}$ und damit gilt: $y = \hat{y} + \varepsilon$. Dieses ε nennt man Residuum (engl.: *residual*) und man möchte es minimieren, um ein möglichst gutes Modell zu erhalten und damit eine möglichst gute Vorhersage zu erreichen. Für erwartungstreue Schätzer gilt, dass der Erwartungswert $E\{\varepsilon\} = 0$ und die Varianz $Var\{\varepsilon\} = \sigma^2$ sind. Der Erwartungswert ist gleich Null, da $E\{\varepsilon\} \neq 0$ einen erwarteten konstanten Offset von der richtigen Klasse y bedeuten würde. Die beiden statistischen Kenngrößen Erwartungswert und Varianz werden in Abschnitt 2.2 näher erläutert.

Zur Minimierung des Residuums verwendet man im einfachsten Fall die *least-squares*

Methode, bei der das quadrierte Residuum minimiert wird [HMS01]:

$$\sum_{n=1}^{N} \varepsilon_n^2 = \sum_{n=1}^{N} \left(y_n - \left(a_0 + \sum_{d=1}^{D} a_d \; x_{nd} \right) \right)^2$$

Dabei ist $y_n = \hat{y}_n + \varepsilon_n = a_0 + \sum_{d=1}^{D} a_d \; x_{nd} + \varepsilon_n, \quad 1 \leq n \leq N.$

2.2 Bias–Varianz Problem

Das sogenannte *Bias–Varianz–Problem* beziehungsweise *–Dilemma* tritt bei der Vorhersage von Werten immer wieder auf und soll deshalb im Folgenden erläutert und mittels eines Beispiels verdeutlicht werden.

Bei der vorhersagenden Modellierung möchte man anhand eines Prädiktors $\mathbf{x}$ der Zielvariablen y einen Wert zuzuordnen. Die Klassifikation/Regression wird dabei – wie bereits im vorherigen Abschnitt angesprochen – nicht immer korrekte Werte liefern, da es aufgrund von Ungenauigkeiten des Modells oder der Überlappung von Klassen immer zu Fehlklassifikationen kommen kann. Es gilt also $y = f(\mathbf{x}, \Theta) + \varepsilon$. Im Folgenden werden die Parameter Θ des Modells weggelassen, falls diese für die Betrachtung nicht relevant sind.

Es gibt zwei Maße, mit denen man die Genauigkeit eines Schätzers messen kann: Varianz und Bias. Die Varianz ist ein Maß für die Streuung und definiert als die erwartete quadrierte Abweichung des Schätzers vom Erwartungswert des Schätzers. Im Gegensatz dazu steht der Bias, ein Maß für die Genauigkeit und definiert als die Abweichung des Erwartungswert des Schätzers vom tatsächlichen Wert.

Für einen Prädiktor $\mathbf{x}$ ergibt sich für den erwarteten Fehler einer Klassifikation $\hat{f}(\mathbf{x})$ bei Verwendung der mittleren quadratischen Abweichung (engl.: *squared-error loss*) (nach [HTF01])

$$\begin{aligned} E\{(f(\mathbf{x}) - \hat{f}(\mathbf{x}))^2 \,|\, \mathbf{x}\} &= \gamma^2(\mathbf{x}) + [E\{\hat{f}(\mathbf{x}) \,|\, \mathbf{x}\} - f(\mathbf{x})]^2 + E\{\hat{f}(\mathbf{x}) - E\{\hat{f}(\mathbf{x}) \,|\, \mathbf{x}\} \,|\, \mathbf{x}\}^2 \\ &= \gamma^2(\mathbf{x}) + Bias^2\left(\hat{f}(\mathbf{x})\right) + Var\left(\hat{f}(\mathbf{x})\right) \\ &= \text{Minimaler Fehler} + \text{Bias}^2 + \text{Varianz} \end{aligned}$$

Der erste Term der Gleichung ist die Varianz der zu klassifizierenden Variablen um den exakten Wert $f(\mathbf{x})$ im Mittel über alle Trainingsdaten und kann nicht vermieden werden. Der zweite und dritte Term sind der quadrierte Bias respektive die Varianz von $\hat{f}(\mathbf{x})$. Diese beiden Terme kann man nun nicht gleichzeitig minimieren: Benutzt man ein einfaches Modell – im Extremfall immer den gleichen Wert y, ohne auf den Prädiktor zu achten – so ist die Varianz Null, allerdings kann der Bias dann einen hohen Wert haben. Bei einem komplexen Modell ist der Effekt umgekehrt: Die Varianz kann aufgrund der Schwankungen innerhalb der Trainingsdaten hoch sein, der Bias erreicht hingegen einen niedrigen Wert, weil die $\hat{f}(\mathbf{x})$ im Mittel über alle Trainingsdaten nahe an $f(\mathbf{x})$ liegen. Bei der Klassifikation muss man also immer darauf achten, einen Kompromiss zwischen Bias und Varianz und einfacher und hoher Modellkomplexität zu erreichen.

Das Bias-Varianz-Problem soll nun noch mittels eines Beispiels beschrieben werden: Das generelle Problem besteht darin, die Relation zwischen $\mathbf{x}$ und y mittels einer Funktion $f(\mathbf{x}, \Theta)$ zu beschreiben, dabei jedoch nicht zu sehr an den Trainingsdaten angepasst zu sein (*overfitting*) und ein möglichst gutes Ergebnis bei neu zu klassifizierenden Werten zu erhalten. Die Frage ist also: Welche Form soll die Funktion $f(\mathbf{x}, \Theta)$ haben?

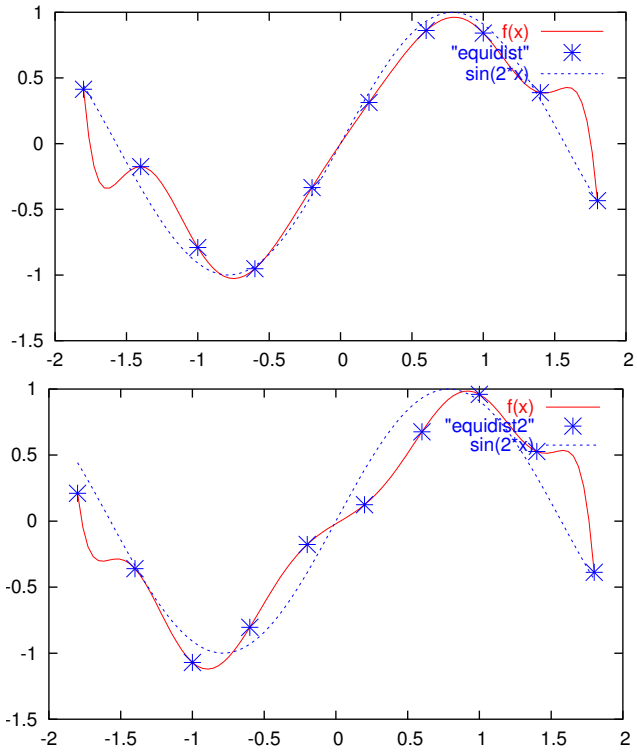

Abbildung 2: *Least-squares* Methode mit 10 Freiheitsgraden

Abbildung 2.2 zeigt blau gestrichelt die Funktion $\sin(2x)$ im Bereich [-2,2]. Diese Funktion soll nun mittels zwei verschiedener Mengen von Trainingsdaten, die sich aus der Funktion $\sin(2x) + \varepsilon$ mit $\epsilon \sim \mathcal{N}(0, \frac{1}{9})$ ergeben, geschätzt werden. Dazu wurden zweimal zufällig zehn Trainingsdaten erzeugt; diese sind in Abbildung 2.2 jeweils mit blauen Kreuzen markiert.

Abbildung 2.2 zeigt die Annäherung an diese zehn Trainingspunkte mittels einer polynomiellen Funktion der Ordnung neun: $f_{10}(\mathbf{x}) = \sum_{i=0}^{9} a_i \cdot x^i$. Man sieht, dass die Trainingspunkte jeweils exakt auf den roten Kurven liegen, da mittels einer Funktion vom Grad 9 die Trainingsdaten exakt nachgebildet werden können. Die blau gestrichelte Funktion $\sin(2x)$ wird jedoch in beiden Fällen nur schlecht nachgebildet – es findet das bereits oben erwähnte *overfitting* der Trainingsdaten statt: Die geschätzte Funktion $f_{10}(\mathbf{x})$ orientiert sich zu stark an den gegebenen Trainingswerten und liefert kein gutes Modell für neu zu klassifizierende Daten. Hier hat $f_{10}(\mathbf{x})$ eine hohe Varianz, da die Funktion stark mit verschiedenen Trainigsdaten schwankt. An der Abbildung sieht man diese hohe Varianz an der unterschiedlichen Formen der beiden Kurven. Allerdings ist der Bias in beiden Fällen im Mittel nahe bei Null, weil die Funktion im Mittel jeweils gut approximiert wird.

Mit einer polynomiellen Funktion der Form $f_2(\mathbf{x}) = a_0 + a_1 \cdot x$ werden in Abbildung 2.2 die zehn Trainingsdaten angenähert. Man erhält bei beiden Traingsmengen eine Gerade, die die ursprüngliche sinusförmige Funktion nur unzureichend nachbilden kann. Dies bezeichnet man im Gegensatz zum *overfitting* mit *underfitting*, da die gegebenen Trainingsdaten nur sehr schlecht nachgebildet werden und kein gutes Modell gefunden wird. $f_2(x)$ hat eine niedrige Varianz, da es nur geringe Schwankungen bei verschiedenen Trainingsdaten gibt: Bei anderen Trainingsdaten hat $f_2(\mathbf{x})$ immer eine ähnliche Form, eine

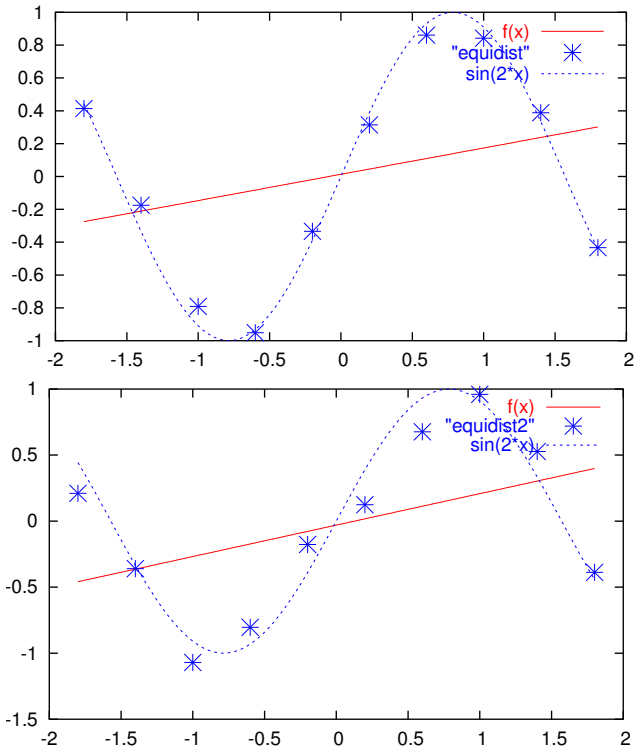

Abbildung 3: *Least-squares* Methode mit 2 Freiheitsgraden

leicht steigende Gerade. Der Bias ist allerdings hoch; im Mittel wird die Sinus-Funktion schlecht approximiert.

Die Abbildung 2.2 zeigt die Annäherung an die zehn Trainingspunkte mittels einer polynomiellen Funktion der Ordnung drei: $f_4(\mathbf{x}) = \sum_{i=0}^{3} a_i \cdot x^i$. Man erhält in beiden Fällen eine gute Annäherung an die ursprüngliche Funktion $\sin(2x)$ und vermeidet ein Overfitting der Trainingsdaten; für neu zu klassifizierende Daten liefern die beiden Annäherungen gute Werte und der quadrierte Fehler ist niedrig. Im vorliegenden Beispiel stellt dies einen guten Kompromiss aus Bias und Varianz dar, beide haben ein niedriges Niveau.

Abbildung 5 stellt diesen Zusammenhang noch einmal verallgemeinert dar: In dieser Abbildung wird die Komplexität eines Modells dem Vorhersagefehler gegenübergestellt. Bestimmt man nun anhand der Trainingsdaten die Parameter des Modells, so ist der Vorhersagefehler bei kleiner Modellkomplexität hoch, fällt allerdings mit zunehmender Modellkomplexität – das Modell passt sich immer mehr den Trainingsdaten an. Dieses Verhalten wurde auch im vorherigen Beispiel deutlich: Bei Verwendung des linearen Modells $f_2(\mathbf{x}) = a_0 + a_1 \cdot x$ ist der Vorhersagefehler groß (niedrige Varianz, hoher Bias), allerdings ist das Modell einfach zu beschreiben. Erhöht man die Modellkomplexität, so sinkt der Vorhersagefehler und damit sinkt der Bias, allerdings kann die Varianz steigen; bei Verwendung einer Funktion der Ordnung neun, $f_{10}(\mathbf{x}) = \sum_{i=0}^{9} a_i \cdot x^i$, liegt der Vorhersagefehler bei Null, da die Trainingspunkte genau nachgebildet werden können.

Betrachtet man nun allerdings den Vorhersagefehler des Modells in Bezug auf die Testmenge – eine Menge von Trainingsdaten, die nicht zur Bestimmung der Parameter, sondern nur zur Evaluierung der Güte des Modells benutzt werden – so zeigt sich ein anderer Effekt: Bei kleiner Modellkomplexität ist der Vorhersagefehler hoch; das Modell ist zu einfach, um

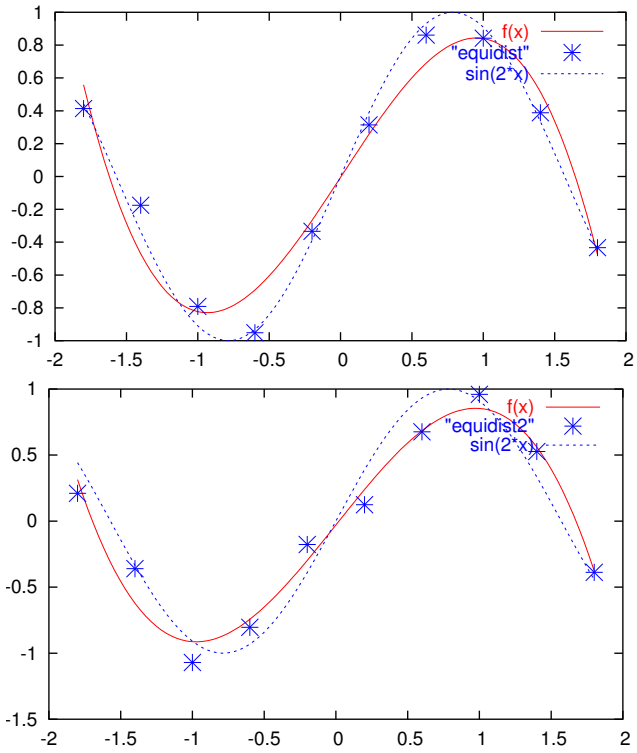

Abbildung 4: *Least-squares* Methode mit 4 Freiheitsgraden

eine korrekte Vorhersage durchführen zu können. Mit steigender Modellkomplexität fällt der Vorhersagefehler, steigt allerdings ab einem bestimmten Punkt wieder. Ab diesem Punkt kommt es zu *overfitting* und das möchte man unter anderem vermeiden. Man sucht also nach einem Kompromiss zwischen der Modellkomplexität und dem Vorhersagefeh-

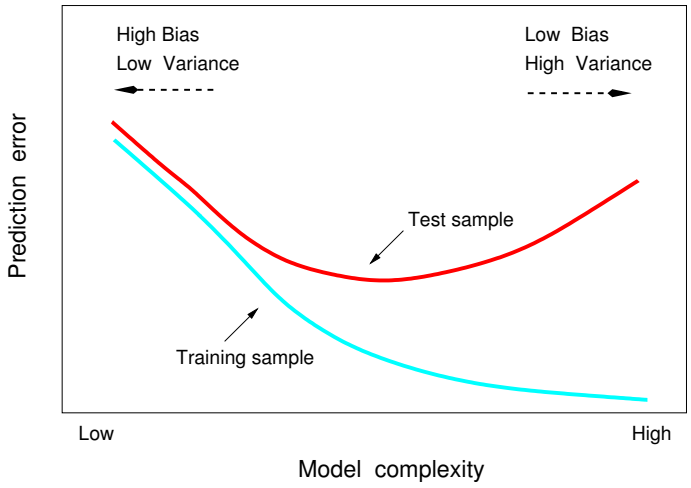

Abbildung 5: Zusammenhang zwischen Modellkomplexität und Vorhersagefehler (aus [HT90])

ler und damit verbunden einen Kompromiss zwischen Bias und Varianz. Im vorherigen Beispiel wurde dies mittels einer Funktion der Ordnung drei, $f_4(\mathbf{x}) = \sum_{i=0}^{3} a_i \cdot x^i$, erreicht.

Die Einteilung in Trainings- und Testmenge sowie einige Verfahren zur Messung der Güte eines Verfahrens werden in Abschnitt 4 betrachtet.

3 Predictive Modeling

Ein einfaches Modell für die Klassifikation von Werten bilden die sogenannten *linearen Diskriminaten* (engl.: *linear discriminants*), die lineare Entscheidungsgrenzen bestimmen und zunächst kurz vorgestellt werden. Das Prinzip der linearen Diskriminanzanalyse (LDA) beruht auf einer Arbeit von Sir Ronald A. Fisher, die dieser 1936 unter dem Titel *„The use of multiple measurements in taxonomic problems“* [Fis36] veröffentlichte. In dieser Arbeit führt er lineare Diskriminanten für den Fall mit zwei Klassen $c = 0$ und $c = 1$ ein; diese Methode soll hier kurz vorgestellt werden:

Lineare Diskriminanten suchen nach der Linearkombination der Prädiktoren, die die verschiedenen Klassen am besten voneinander „trennt“. Dabei benutzt Fisher implizit die Vorraussetzung von identischen Kovarianzmatrizen. Die Trennung soll die Eigenschaft haben, dass

1. die Abstände zwischen den verschiedenen Klassen maximiert werden, und
2. die Abstände innerhalb einer Klasse minimiert werden.

Dazu maximiert man das Verhältnis von Varianz zwischen Klassen zu Varianz in den Klassen. Auf eine mathematische Beschreibung soll verzichtet werden.

Im Folgenden sollen nun einige Modelle zur Vorhersage vorgestellt werden: In Abschnitt 3.1 wird die logistische Diskriminanzanalyse erläutert, mit der man nichtlineare Entscheidungsgrenzen bestimmt. Bei der Nächste-Nachbarn-Methode (Abschnitt 3.2) basiert die Vorhersage auf der Klassenzugehörigkeit der k nächsten Nachbarn eines Punktes. Der naive Bayes-Klassifizierer nimmt die lineare Unabhängigkeit der Prädiktoren an und erreicht so ein vereinfachtes, aber dennoch sehr leistungsfähiges Modell; die Vorgehensweise wird in Abschnitt 3.3 beschrieben. Schließlich stellt Abschnitt 3.4 noch die Grundideen einiger anderer Verfahren vor.

3.1 Logistische Diskriminanzanalyse

In Abschnitt 2 wurde das lineare Modell vorgestellt, mit dessen Hilfe man lineare Entscheidungsgrenzen finden kann. Um nun komplexere Funktionen $f(\mathbf{x})$ approximieren zu können und dennoch die Eigenschaft der Linearität zu erhalten, kann man dieses Modell noch erweitern zum *verallgemeinerten linearen Modell* (engl.: *generalized linear model*): Die Zielvariable y, bei der man eine zur Exponentialfamilie gehörende Verteilung annimmt, ist nur indirekt über eine sogenannten *Link*-Funktion $g(y)$ von der Linearkombination der $\mathbf{x}$ abhängig [HMS01]:

$$g(y) = a_0 + \sum_{d=1}^{D} a_d x_d$$

Die *logistische Diskriminanzanalyse* (engl.: *logistic discriminant analysis*) benutzt als Link-Funktion den Logarithmus und wird deshalb auch log-lineares Modell genannt. Zuerst wurde sie 1962 von Cornfield verwendet, um koronare Herzkrankheiten vorherzusagen und

seitdem wurde diese Art der Analyse auf eine große Anzahl von Verteilungen ausgeweitet, besonders im Zwei-Klassen-Fall wird sie häufig benutzt [McL92].

Logistische Diskriminaten $f_{log}(\mathbf{x})$ erlauben im einfachen Fall die Vorhersage einer kategorischen, binären Zielvariablen y, beispielsweise mit den beiden Klassen {Erfolg, Scheitern} oder {krank, gesund}. Dabei bedeutet $f_{log}(\mathbf{x}) = 1$, dass die Diskrimante der Zielvariablen y die Klasse $c = 1$ zuweist.

Dabei sind D Prädiktor-Variablen $\mathbf{x} = (x_1, \ldots, x_D)^T \in \mathbb{R}^D$ bekannt, mit deren Hilfe man nun eine möglichst gute Klassifikation durchführen möchte. Die idealen Werte der Diskriminante lassen sich in der Regel nur approximativ erreichen, da aufgrund der stochastischen Natur der Variablen eine exakte Vorhersage in der Praxis kaum möglich ist.

Im Gegensatz zum linearen Modell verfährt man bei der logistischen Diskriminanzanalyse wie folgt: Die Wahrscheinlichkeit für Klasse $c = 1$ bei gegebenem Prädiktor $\mathbf{x}$ beträgt

$$p_{log}(c = 1|\, \mathbf{x}) = \frac{1}{1 + \exp(\alpha^T \mathbf{x})}$$

Da es nur zwei mögliche Klassen $c = 1$ und $c = 0$ gibt, ist die Wahrscheinlichkeit für das Gegenereignis $p_{log}(c = 0 \,|\, \mathbf{x})$:

$$p_{log}(c = 0 \,|\, \mathbf{x}) = 1 - p_{log}(c = 1|\, \mathbf{x}) = \frac{\exp(\alpha^T \mathbf{x})}{1 + \exp(\alpha^T \mathbf{x})}$$

Die wesentliche Annahme der logistischen Diskriminanzanalyse ist die Linearität des logarithmierten *Chancenverhältnisses* (engl.: *odds-ratio)*:

$$log\left(\frac{p_{log}(c = 0 \,|\, \mathbf{x})}{p_{log}(c = 1|\, \mathbf{x})}\right) = a_0 + \sum_{d=1}^{D} a_d x_d$$

und damit ist bei der logistischen Diskriminanzanalyse die Link-Funktion der Logarithmus.

Odds-ratios bezeichnen dabei das Verhältnis der relativen Häufigkeiten der möglichen Ergebnisse und werden meistens in der Form „$a : b$“ dargestellt. Beispielsweise bedeutet „$4 : 1$ zugunsten von Klasse Null“, dass das erste Ergebnis (hier symbolisiert durch die Klasse Null) viermal so häufig auftritt wie das zweite Ergebnis, also Klasse Eins. Oddsratios stehen in direkter Beziehung zu Wahrscheinlichkeiten und können mittels folgender Rechnungen ineinander überführt werden:

- $p(c = 0 \,|\, \mathbf{x}) = \frac{a}{a+b}$,
  falls die odds-ratios in der Form „$a : b$ zugunsten von Klasse Null“ angegeben sind. „$4 : 1$“ liefert also eine Wahrscheinlichkeit von Klasse Null in Höhe von
  $p(c = 0 \,|\, \mathbf{x}) = \frac{4}{4+1} = 0,8.$

- $a : b = \frac{p(c=0 \,|\, \mathbf{x})}{1-p(c=0 \,|\, \mathbf{x})}$
  Ist zum Beispiel die Wahrscheinlichkeit für Klasse Null gleich $0,4$, also
  $p(c = 0 \,|\, \mathbf{x}) = 0,4$, so liefert dies odds-ratios in Höhe von $\frac{0,4}{1-0,4} = \frac{0,4}{0,6} = \frac{2}{3}$ oder „$2 : 3$“

Die oben definierte logistische Diskriminante trägt auch den Namen *Sigmoid-Funktion*:

$$p_{log}(c = 0 \,|\, \mathbf{x}) = \frac{\exp(\alpha^T \mathbf{x})}{1 + \exp(\alpha^T \mathbf{x})} = \text{sigmoid}(\alpha^T \mathbf{x})$$

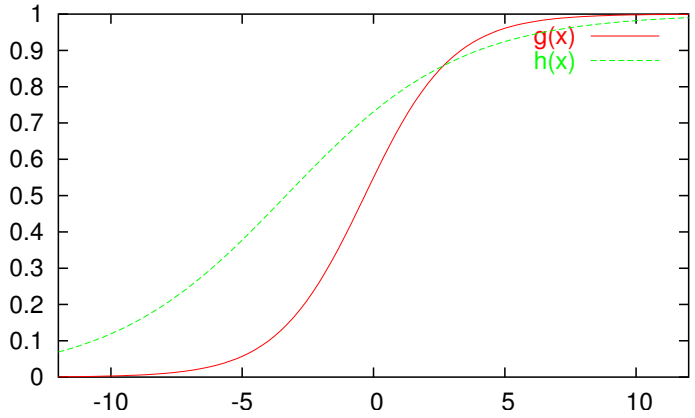

Abbildung 6: Beispiel für logistische Diskriminanten

Logistische Diskriminanten haben im zweidimensionalen Fall typischerweise eine „S-Form“ und der Grenzwert gegen $-\infty$ ist 0, der Grenzwert gegen ∞ ist 1. Abbildung 6 zeigt den Verlauf der Kurven für die beiden Funktion $g(x)$ und $h(x)$ mit

$$\begin{aligned} g(x) &= \frac{\exp(0,2+0,6x)}{1+\exp(0,2+0,6x)} \\ h(x) &= \frac{\exp(1+0,3x)}{1+\exp(1+0,3x)} \end{aligned}$$

im Bereich von $[-12, 12]$

Die Unterschiede zwischen einer linearen und einer logistischen Diskriminante liegen vor allem im Wertebereich des Erwartungswertes beziehungsweise in der Annahme über die Verteilung der Zufallskomponente, die eine exakte Klassifikation erschwert.

Bei einer linearen Diskriminante kann der Erwartungswert zwischen $-\infty$ und ∞ liegen, bei einer logistischen Diskriminanten hingegen nur im Bereich [0,1].

Bei einer linearen Diskriminante ergibt sich die Klasse als $y = f(\mathbf{x}) + \varepsilon$. Dabei wird ε als normalverteilt mit Erwartungswert 0 und Varianz σ^2 angenommen. Bei einer logistischen Diskriminanten $f_{log}(\mathbf{x}) + \varepsilon$ hingegen wird aufgrund der binären Entscheidungsvariablen eine Binomial-Verteilung der Zufallskomponente angenommen.

Man kann logistische Diskriminanten auch verallgemeinern und so die Fälle abdecken, bei denen mehr als zwei Klassen auftreten [McL92]: Seien beispielsweise drei Klassen $c = 0, 1, 2$ möglich, in denen y liegen kann. Um nun das logistische Modell anwenden zu können, muss y in zwei binäre Variablen y_1 und y_2 aufgeteilt werden. Dann gilt $y = 0$ genau dann, wenn $y_1 = 0$ und $y_2 = 0$; $y = 1$ genau dann, wenn $y_1 = 1$ und $y_2 = 0$ und analog $y = 2$ genau dann, wenn $y_1 = 0$ und $y_2 = 1$ ist. Die hierbei verwendeten Variablen $y_i, i = 1, 2$ nennt man *Design-* oder *Dummy-Variablen*. Nach dem gleichen Prinzip kann jede Variable mit $c > 2$ Klassen durch $c - 1$ binäre Variablen ersetzt werden. Andere Verfahren, um die logistische Diskriminanzanalyse auf den Mehr-Klassen-Fall zu übertragen, sind in der Literatur unter den Namen *Multi-Class Regression* und *Maximum Entropy*-Verfahren zu finden.

Als Beispiel zur logistischen Diskriminanzanalyse soll eine Analyse des Challenger-Unglücks dienen [HMS01]: Am 28. Januar 1986 explodierte die Raumfähre Challenger

nur wenige Minuten nach dem Start. Durch Rekonstruktion des Unfalls konnte herausgefunden werden, dass brüchige Gummiringe an einer der Antriebsraketen die unmittelbare Unglücksursache darstellten. Durch diese Ringe konnten heiße Verbrennungsgase entweichen, sich entzünden und dies führte letztendlich zur Explosion.

Aus Analysen von früheren Flügen wusste man, dass diese Gummiringe temperaturempfindlich waren, man hatte dazu die Daten aus 22 früheren Flügen und die zugehörige Temperatur ausgewertet; Abbildung 7 zeigt die Messwerte.

Für den Tag des Starts war eine Temperatur von 31° Fahrenheit vorhergesagt, also deutlich kühler als bei früheren Flügen. Anhand von Abbildung 7 kann man vermuten, dass für niedrigere Temperaturen ein höheres Risiko einer Beschädigung besteht.

Wendet man nun das Verfahren der logistischen Diskriminanzanalyse an, um ein Modell für die Datenpunkte zu erhalten, so ergibt sich als Näherung:

$$\begin{aligned} f(\mathbf{x}) &= \frac{\exp(\alpha_0 + \alpha_1 x)}{1 + \exp(\alpha_0 + \alpha_1 x)} \\ &= \frac{\exp(15 - 0.25x)}{1 + \exp(15 - 0.25x)} \end{aligned}$$

wobei x die Temperatur in Grad Fahrenheit bezeichnet.

Abbildung 8 zeigt diese Funktion zusammen mit den Messwerten der früheren Flüge. Anhand der Abbildung sieht man, dass die Wahrscheinlichkeit für eine Beschädigung der Gummiringe bei Verwendung dieses Modells bei einer Temperatur von 31° Fahrenheit nahe bei 1 lag. Deshalb hätte ein Start bei einer so niedrigen Temperatur – bei Annahme dieses Modells – nicht stattfinden dürfen.

3.2 Nächste-Nachbarn-Methode

Eine weitere Methode zur Klassifikation liefert die sogenannte *Nächste-Nachbarn-Methode* (engl.: *nearest neighbor method*). Die Grundidee hierbei ist sehr einfach zu beschreiben: Um ein neu zu klassifizierendes Objekt mit Prädiktor $\mathbf{x} \in \mathbb{R}^D$ einer von c Klassen, $c = 1, \ldots, C$, zuzuordnen, werden zunächst die k nächsten Nachbarn von $\mathbf{x}$ bezüglich eines festgelegten Distanz- oder Ähnlichkeitsmaßes untersucht und deren Klasse festgestellt. $\mathbf{x}$ wird dann genau der Klasse c^* zugeordnet, die bei den k Nachbarn am häufigsten auftritt; aufgrund dieser „Nachbarschaftsbeziehung“ zwischen $\mathbf{x}$ und den k Nachbarn trägt diese Methode ihren Namen.

Um die Methode einsetzen zu können müssen vor allem zwei Fragen geklärt werden:

1. Welches Distanzmaß wird zur Bestimmung der Nähe benutzt?

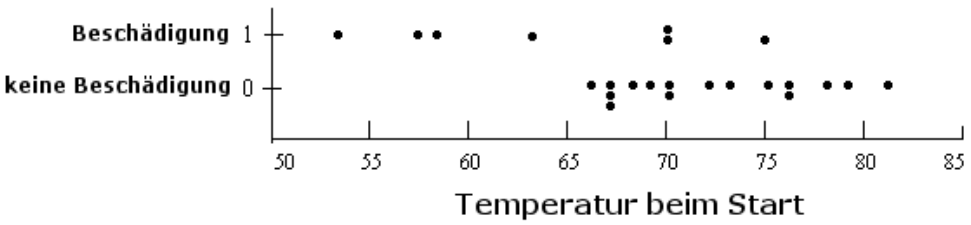

Abbildung 7: Häufigkeit der Beschädigung der O-Ringe bei gegebener Temperatur in °F

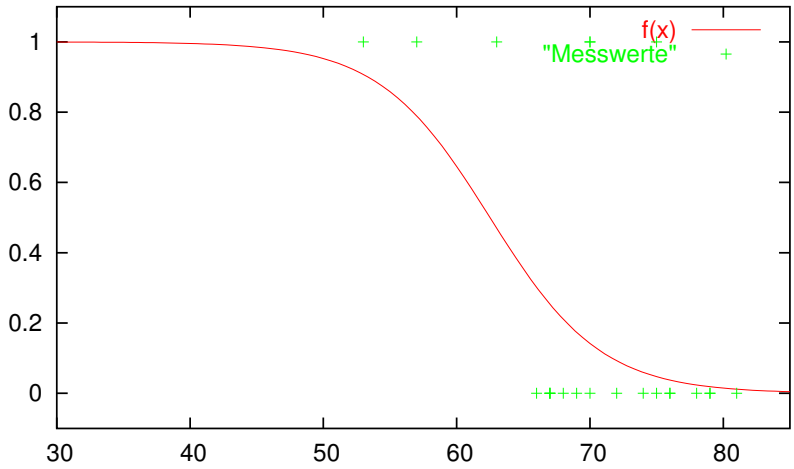

Abbildung 8: Logistische Diskriminante für die Beschädigung der O-Ringe

2. Wie groß soll k sein, wie viele Nachbarn werden also untersucht?

Ein Distanzmaß ist nötig, um Aussagen über die Nähe zwischen zwei Prädiktoren $\mathbf{x}$ und $\mathbf{x}'$ machen zu können und daran anschließend eine Zuordnung zu einer bestimmten Klassen durchführen zu können. Als Distanz zwischen zwei Prädiktoren $\mathbf{x}$ und $\mathbf{x}'$ können verschiedene Normen benutzt werden; die einfachsten sowie am häufigsten benutzen Distanzmaße stellen die euklidische Distanz

$$dist_{euk}(\mathbf{x}, \mathbf{x}') = \sqrt{\sum_{d=1}^{D} |x_d - x'_d|^2}$$

und die Manhattan-Distanz

$$dist_{man}(\mathbf{x}, \mathbf{x}') = \sum_{d=1}^{D} |x_d - x'_d|$$

dar. Diese nichtparametrischen Abstandmaße kann man verallgemeinern zur *Minkowski-Distanz*:

$$dist_p(\mathbf{x}, \mathbf{x}') = \left(\sum_{d=1}^{D} |x_d - x'_d|^p \right)^{\frac{1}{p}}, \quad p \in \mathbb{R}^+.$$

Im Unterschied zu diesen Normen ohne Parameter kann man auch Distanzmaße definieren, die von einem Parameter abhängig sind. Eine einfache Möglichkeit zur Einführung von Parametern stellt ein gewichtetes euklidisches Distanzmaß dar: Dabei wird jeder Summand von $dist_p(\mathbf{x}, \mathbf{x}')$ noch mit ein Gewichtungsfaktor $\beta_d, d = 1, \ldots, D$ multipliziert. Eine andere Möglichkeit bietet die sogenannte *Mahalanobis-Distanz*, eine Verallgemeinerung der euklidischen Distanz unter Berücksichtigung der Korrelationen der einzelnen Prädiktoren. Sie ist folgendermaßen definiert:

$$dist_{mah}(\mathbf{x}, \mathbf{x}') = (\mathbf{x} - \mathbf{x}')^T \Sigma^{-1} (\mathbf{x} - \mathbf{x}')$$

Der Parameter Σ der Mahalanobis-Distanz ist die Kovarianzmatrix der Prädiktoren, also

$$\Sigma = \begin{pmatrix} Var(x_1) & Cov(x_1, x_2) & \cdots & Cov(x_1, x_m) \\ Cov(x_2, x_1) & Var(x_2) & \cdots & Cov(x_2, x_m) \\ \vdots & \vdots & \ddots & \vdots \\ Cov(x_m, x_1) & Cov(x_m, x_2) & \cdots & Var(x_m) \end{pmatrix}$$

Dabei bezeichnet $Cov(x_i, x_j)$ die Kovarianz (engl.: *covariance*) von x_i und x_j:

$$Cov(x_i, x_j) = E\{(x_i - E\{x_i\}) \cdot (x_j - E\{x_j\})\}$$

und $Var(x_i)$ die Varianz (engl.: *variance*) der i-ten Komponente:

$$Var(x_i) = Cov(x_i, x_i)$$

Die Varianz $Var(x_i)$ ist ein Maß für die Streuung und gibt die erwartete quadrierte Abweichung des Messwerts vom Erwartungswert des Messwerts an; eine große Varianz bedeutet also eine große Streuung der Werte um den Erwartungswert. Die Kovarianz $Cov(x_i, x_j)$ hingegen ist ein Maß für die Abhängigkeit von x_i und x_j und wird zum Beispiel bei der Varianzberechnung von Summen benötigt: $Var(x + y) = Var(x) + Var(y) + 2 \cdot Cov(x, y)$. Durch Standardisierung erhält man aus der Kovarianz die Korrelation – ein normalisiertes Maß für den Zusammenhang von zwei Variablen:

$$Corr(x_i, x_j) := \frac{Cov(x_i, x_j)}{\sqrt{Var(x_i)}\sqrt{Var(x_j)}}.$$

Wie bereits oben angesprochen ist die Mahalanobis-Distanz eine Generalisierung der euklidischen Distanz: Durch Multiplikation mit dem Inversen der Kovarianzmatrix (und somit einer Gewichtung mittels der Varianzen und Kovarianzen) wird aus der euklidischen Distanz die Mahalanobis-Distanz. Sie ist invariant unter linearen Operationen auf der Variable, liefert also ein normiertes Entfernungsmaß.

Abbildung 9 zeigt ein Beispiel (angelehnt an: `http://www.galactic.com/Algorithms/discrim_mahaldist.htm`) für die beiden Distanzmaße: Mittels eines Spektrometers wird die Absorption von verschiedenen Stoffen bei verschiedenen Wellenlängen bestimmt. Bei der Analyse von gleichen Stoffen erwartet man ähnlichen Spektren – unterschiedliche Stoffe weisen Unterschiede in den Spektren auf. Diese Unterschiede in den Spektren liefern nun eine Möglichkeit, die einzelnen Stoffe mittels der Nächste-Nachbarn-Methode zu klassifizieren. Verschiedene Messungen beim gleichen Stoff liefern allerdings aufgrund von stochastischen Schwankungen im Messprozess nie das exakt gleiche Spektrum, man benötigt also ein Distanzmaß, um die Ähnlichkeit von zwei Messungen zu bestimmen. Abbildung 9 zeigt für zwei bestimmte Stoffe die Messergebnisse der Absorption bei 722 cm^{-1} und 1375 cm^{-1}. Man sieht, dass die verschiedenen Punkte eine ellipsenförmige Verteilung aufweisen; die Mittelwerte sind mit $\overline{\mathbf{X}}$ beziehungsweise $\overline{\mathbf{Y}}$ markiert. Verwendet man nun die euklidische Distanzfunktion $dist_{euk}$, so bildet sich aufgrund der quadratischen Form von $dist_{euk}$ eine kreisförmige Nachbarschaft rund um die Mittelwerte. Bei Verwendung der Mahalanobis-Distanz $dist_{mah}$ hingegen hat die Nachbarschaft die Form einer Ellipse und passt sich besser der Verteilung der Punkte an.

Um eine Klassifikation vornehmen zu können, muss man mittels Trainingsdaten für jede der c Klassen von möglichen Stoffen, $c = 1, \ldots, C$, einen Zusammenhang analog zu Abbildung 9 bestimmen. So erhält man C verschiedene Mittelwerte $\overline{\mathbf{X}}_c$. Möchte man nun

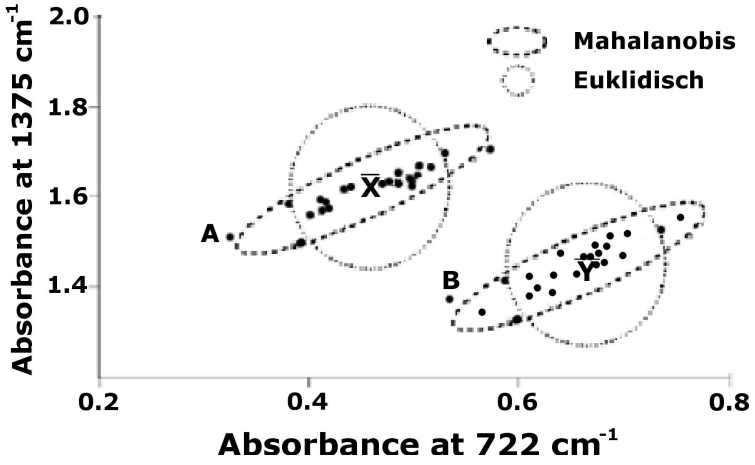

Abbildung 9: Absorption zweier Wellenlängen bei gleichem Stoff

einen Stoff einer Klasse zuordnen, so geht man wie folgt vor: Zunächst führt man die Spektroskopie durch und ermittelt die Absorption bei den beiden Wellenlängen $722\,cm^{-1}$ und $1375\,cm^{-1}$. Man berechnet die k nächsten Nachbarn zu diesen Messwerten und ordnet den Stoff zu der Klasse c^* zu, die unter den k Nachbarn am häufigsten vertreten ist.

Bei der Wahl der Nachbarschaft werden nun die Unterschiede zwischen der euklidischen und der Mahalanobis-Distanz deutlich: Verwendet man als Distanzmaß $dist_{euk}$, so wird die Verteilung der Punkte innerhalb einer Stoffgruppe nicht berücksichtigt, lediglich der relative Abstand des zu klassifizierenden Stoffes zu den verschiedenen Mittelwerten $\overline{\mathbf{X}}_c$, $c = 1, \ldots, C$, ist relevant. Außerdem gibt die euklidische Distanz nicht an, wie gut der zu klassifizierende Stoff zu den Trainingsdaten passt: In Abbildung 9 sind die beiden Punkte **A** und **B** eingetragen, die jeweils den gleichen euklidischen Abstand zum Mittelwert $\overline{\mathbf{X}}$ haben, bei Verwendung von $dist_{euk}$ also den gleichen Abstand zu $\overline{\mathbf{X}}$ aufweisen. **B** hat auch den gleichen Abstand zwischen $\overline{\mathbf{X}}$ und $\overline{\mathbf{Y}}$.

A liegt auf der verlängerten Achse der Punkte, gehört also wahrscheinlich zur Stoffklasse, die zu $\overline{\mathbf{X}}$ gehört; **B** hingegen liegt auf der verlängerten Achse des zu $\overline{\mathbf{Y}}$ gehörenden Stoffes. Bei Anwendung der euklidischen Distanzfunktion wäre die Wahrscheinlichkeit für die Zuordnung von **B** zur linken Stoffklasse allerdings genauso groß wie die Wahrscheinlichkeit der Zuordnung zum rechten Stoff, da der Abstand von **B** zu $\overline{\mathbf{X}}$ beziehungsweise $\overline{\mathbf{Y}}$ gleich groß ist.

Verwendet man hingegen $dist_{mah}$ als Distanzmaß, so werden die Varianzen und Kovarianzen der Trainingsdaten einer Klasse berücksichtigt. Die Multiplikation mit dem Inversen der Kovarianzmatrix Σ sorgt dafür, dass Werte mit hoher Korrelation nur ein kleines Gewicht haben und umgekehrt. Nun liegt **B** näher an $\overline{\mathbf{Y}}$ als an $\overline{\mathbf{X}}$, wird also mit einer höheren Wahrscheinlichkeit der richtigen Klasse zugeordnet.

Im Folgenden soll nun die Wahl von k, also die Anzahl der zu untersuchenden Nachbarn, näher betrachtet werden. Die nächsten k Nachbarn von $\mathbf{x}$ kann man definieren als

$$Neigh_k(\mathbf{x}) := \{\mathbf{x}_{i_1}, \mathbf{x}_{i_2}, \ldots, \mathbf{x}_{i_k} | dist(\mathbf{x}, \mathbf{x}_{i_1}) \leq \ldots \leq dist(\mathbf{x}, \mathbf{x}_{i_k}) \leq dist(\mathbf{x}, \mathbf{x}_{i'}), \forall i'\}$$

Dabei ist $dist(\mathbf{x}, \mathbf{x}_i)$ ein beliebiges Distanzmaß, beispielsweise eine der im vorhergehenden Abschnitt behandelten Normen.

Anhand der Trainingsmenge, bei der die Relation zwischen den Prädiktoren $\mathbf{x}_n, n =$

$1, \ldots, N$ und der zugehörigen Klasse c bekannt ist, kann nun ein neu zu klassifizierender Wert $\mathbf{x}$ einer der Klassen $c = 1, \ldots, C$ zugeordnet werden. Dazu ordnet die Nächste-Nachbarn-Methode $\mathbf{x}$ der Klasse c^* zu, die unter den k Nachbarn am häufigsten vertreten ist:

$$c^* := \underset{c=1,\ldots,C}{\operatorname{argmax}} \; |Neigh_k(\mathbf{x})_c|$$

wobei $Neigh_k(\mathbf{x})_c$ die Menge der $\mathbf{x}_{i_j}$ aus der Nachbarschaft $Neigh_k(\mathbf{x})$ ist, die zur Klasse c gehören.

In der einfachsten Form der Nächste-Nachbarn-Methode wählt man $k = 1$. So erhält man allerdings einen unstabilen Klassifikator, der sensibel auf die Daten reagiert, also eine hohe Varianz hat; schon kleine Änderungen an den Prädiktoren sorgen dann unter Umständen für eine Zuordnung zu einer anderen Klasse. Durch eine Erhöhung von k erreicht man mehr Stabilität der Klassifikation und eine fallende Varianz. Andererseits erreicht man durch Einbeziehung von mehr Nachbarpunkten eine immer dünner besetzte Nachbarschaft und der Bias wird erhöht, da eine „Mittelwertbildung“ stattfindet [HMS01]. Man steht also wieder vor dem Problem von Bias und Varianz, das bereits in Abschnitt 2.2 behandelt wurde: Eine Erhöhung von k verkleinert die Varianz, kann aber zu einem erhöhten Bias führen.

Kriterien für die optimale Wahl der Größe der Nachbarschaft können nicht gegeben werden, da k von den zu klassifizierenden Daten abhängt. Deshalb benutzt man häufig einen adaptive Ansatz, um ein passendes k zu finden: Man probiert mehrere k aus und misst deren Güte, beispielsweise per relativer Häufigkeit der falsch klassifizierten Punkte oder Kreuzvalidierung (mehr dazu in Abschnitt 4). So erhält man dann passend zu den Daten einen optimalen Wert für k. Dabei ist zu beachten, dass man zur Bestimmung der Güte nicht die Trainingsdaten benutzen darf, um *overfitting* zu vermeiden; dieser Test sollte mit unabhängigen Daten ausgeführt werden falls dies möglich ist. Dieses Thema wird ausführlicher in Abschnitt 4 behandelt.

Im letzten Teil dieses Abschnitts soll nun noch auf die Laufzeit sowie einige generelle Aspekte der Nächste-Nachbarn-Methode eingegangen werden:

- Diese Methode ist einfach zu implementieren; die Grundidee ist intuitiv. Bei Verwendung von komplizierten Nachbarschaften (beispielsweise $dist_{mah}$) oder einer großen Anzahl von Trainingsdaten kann jedoch der Rechenaufwand für das Distanzmaß groß werden, da die Trainingsmenge durchsucht werden muss: Die Komplexität ist im brute-force Fall $O(oN)$, wenn N die Anzahl der Trainingsdaten und o die Anzahl der Operationen zur Berechnung der Distanzfunktion angibt.

- Der Speicherverbrauch liegt im Bereich N, da die kompletten Trainingsdaten abgespeichert werden müssen. Somit skaliert die Nächste-Nachbarn-Methode nicht bei einer großen Anzahl von N oder bei Realzeit-Anwendungen [HMS01].

- Eine Ausmusterung von zu klassifizierenden Objekten oder die Erkennung von Ausreißern ist mittels dieser Methode sehr einfach zu realisieren: Falls die Distanz zu den k Nachbarn über einem zu definierendem Schwellenwert liegt, dann kann angenommen werden, dass es sich um einen Ausreißer handelt.

- Bei fehlenden Werten innerhalb eines Prädiktors gibt es eine einfache Möglichkeit, dennoch die Klassifikation durchzuführen: Man führt die Nächste-Nachbarn-Methode im Unterraum aus, der durch die vorhandenen Merkmale bestimmt ist.

- Bei Problemstellungen mit einer großen Anzahl von Variablen kann es aufgrund der vorher bereits angesprochenen Problematik der „Mittelwertbildung“ und dünn besetzten Nachbarschaft zu einer Glättung kommen, die zu einer erhöhten Fehlklassifikation führt [HMS01]. Dies liegt vor allem an der Tatsache, dass bei einer großen Anzahl von Variablen die k nächsten Nachbarn doch „weit“ entfernt sind. Die schlechte Skalierung bei hohen Dimensionen ist jedoch ein Problem, dass alle Klassifikatoren betrifft.

Durch Vorverarbeitung der Daten kann eine Verbesserung der Laufzeit sowie des Speicherbedarfs erreicht werden: Bei der *zusammengefassten* (engl.: *condensed*) Nächste-Nachbarn-Methode werden selektiv die Punkte aus der Nachbarschaft entfernt, ohne die immer noch eine korrekte Klassifikation erfolgen kann. Bei der *veränderten* (engl.: *edited*) Nächste-Nachbarn-Methode hingegen werden isolierte Punkte einer Klasse, die in einer dicht besetzten Region einer anderen Klasse liegen, entfernt.

Die Fehlerrate bei der Nächste-Nachbarn-Methode ist – wie bei allen anderen Verfahren auch – immer höher als die Bayes'sche Fehlerrate. Man kann allerdings zeigen, dass die Fehlerrate bei Verwendung der Nächste-Nachbarn-Methode mit $k = 1$ und einer unbegrenzten Anzahl von Trainingsdaten maximal das Doppelte der Bayes'schen Fehlerrate beträgt [DHS01].

3.3 Naives Bayes-Modell

Im folgenden Abschnitt wird kurz das *Naive Bayes-Modell* (engl.: *naive bayes model*) vorgestellt, ein in der Praxis sehr häufig eingesetztes Verfahren, das auf dem Bayes'schen Satz

$$p(c \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid c) \cdot p(c)}{p(\mathbf{x})} = \frac{p(\mathbf{x} \mid c)p(c)}{\sum\limits_{c} p(c)\ p(\mathbf{x} \mid c)}, \qquad c = 1, \ldots, C$$

beruht.

Dieses Verfahren heißt *naiv*, weil die stochastische Unabhängigkeit der Prädiktor-Variablen angenommen wird:

$$p(\mathbf{x} \mid c) = p(x_1, \ldots, x_D \mid c) \cong \prod_{d=1}^{D} p(x_d \mid c), \qquad c = 1, \ldots, C$$

Diese Annahme ist im Allgemeinen falsch, da bei einem Prädiktor $\mathbf{x}= (x_1, \ldots, x_D)$ durchaus eine Abhängigkeit innerhalb der x_d auftreten kann. Beispielsweise misst eine Ärztin während einer Untersuchung mehrere Merkmale (z.B.: Temperatur, Blutdruck, ...) und benutzt die Ergebnisse zur Diagnose. Allerdings sind diese Messergebnisse im Allgemeinen nicht stochastisch unabhängig voneinander. In vielen praktischen Fällen ist diese Annahme jedoch möglich und führt zu einer starken Vereinfachung des Verfahrens, da anstatt $O(D^N)$ nur noch $O(DN)$ Wahrscheinlichkeiten berechnet werden müssen. Zur Klassifikation benutzt man nun die sogenannte Bayes'sche Entscheidungsregel: Diese benutzt die höchste der auftretenden Wahrscheinlichkeiten, also $\hat{f}(\mathbf{x}) = \underset{c=1,\ldots,C}{\operatorname{argmax}}\ p(c \mid \mathbf{x})$, zur Klassifikation.

3.4 Andere Modelle

Nun wird noch kurz auf einige anderen Verfahren zur Vorhersage eingegangen und deren Idee erläutert: Das *perceptron* (Abschnitt 3.4.1) sowie dessen Erweiterung zum *feed-forward neural network/multi-layer perceptron* benutzt ein Verfahren, das ähnlich zu dem

„*accumulate and fire*“ der Nervenzellen des menschlichen Gehirns ist [HMS01]. *Support-Vektor-Maschinen* (Abschnitt 3.4.2) benutzen eine Transformation in einen höherdimensionalen Raum, um dort nach Hyperebenen zu suchen, die die Trainingsdaten möglichst gut voneinander trennen. Ein *Entscheidungs-* beziehungsweise *Klassifikationsbaum* (Abschnitt 3.4.3) ist eine spezielle Form der nichtlinearen Diskriminanzanalyse und benutzt Bäume zur Klassifikation von Objekten.

3.4.1 Perceptron/Feed-forward neuronales Netzwerk

Die Idee des Perceptron wurde schon in den 1940er Jahren entwickelt, als man versuchte, die Funktionsweise der Neuronen im Gehirn zu modellieren: Dabei „simuliert“ man die Neuronen als Knoten, die verschiedene Eingänge haben und ab einem bestimmten Schwellenwert ein Signal an den Ausgang weiterleiten. Dabei werden die Signale immer an eine höhere Schicht weitergegeben und es gibt keinen Rückkanal, deshalb der Name Feed-forward neuronales Netz. Die Erweiterung zu Multi-layer neuronalen Netzen besteht dann in der Einführung einer oder mehrerer verborgener Zwischenschichten (engl.: *hidden layer*), die die Signale weiter nichtlinear kombinieren und an die nächste Schicht weitergeben.

Dabei haben die Kanten zwischen den Knoten eine bestimmte Gewichtung $a_i \geq 0$, die den Einfluss der Eingänge auf den Ausgang steuern. Mittels der Trainingsdaten „*lernt*“ ein neuronales Netz nun die verschiedenen Gewichtungen: Der Prädiktor $\mathbf{x}$ liegt am Eingang an und falls dieser falsch klassifiziert wird, so werden die Gewichte an den Kanten angepasst – diese Lernphase wird solange fortgeführt bis eine möglichst gute Annäherung an die Trainingsdaten erreicht wurde.

Abbildung 10 zeigt ein Beispiel für ein neuronales Netz: Verschiedene Merkmale $\mathbf{x} = (x_1, \ldots, x_7)$ bilden die Eingänge und es gibt drei mögliche Klassen $c = 1, \ldots, 3$. Die Zwischenschicht sorgt für eine nichtlineare Kombination der Eingänge und die Gewichte an den Kanten werden in der Trainingsphase gelernt.

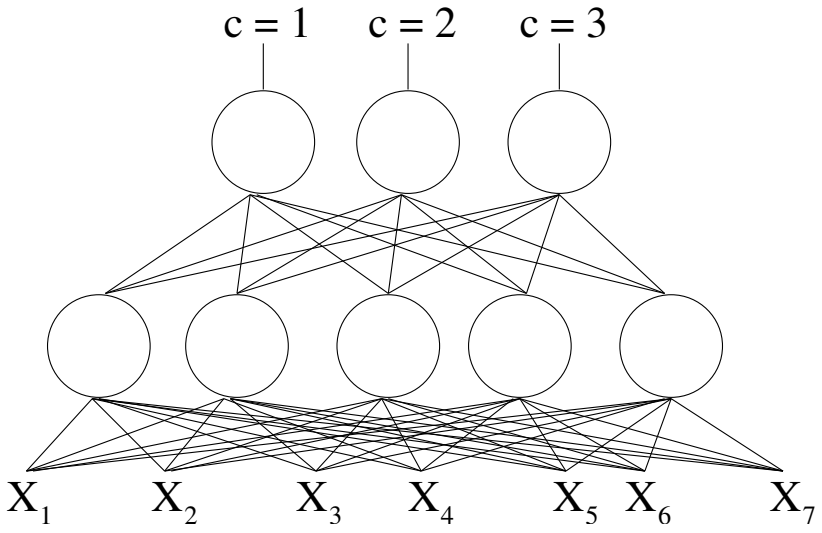

Abbildung 10: Beispiel für Klassifikation mittels eines feed-forward neuronalen Netzes

3.4.2 Support-Vektor Maschinen

Support-Vektor Maschinen (SVM, engl.: *support vector machines*) wurden im Jahr 1992 von Boser, Guyon und Vapnik vorgestellt [BGV92]. Die Idee von SVMs ist es, eine Hyperebene zu bestimmen, die die verschiedenen Klassen mit dem größtmöglichen Abstand voneinander trennt. Die so erhaltene Klassifikation hängt nur von den sogenannten *Support-Vektoren* ab und gibt der Methode ihren Namen. Die Support-Vektoren sind diejenigen Trainingsdaten, die am nächsten an der optimal trennenden Hyperebene liegen; sie sind gleich weit von dieser Ebene entfernt. Diese Trainingspunkte haben den meisten Einfluss auf die Klassifikation [DHS01].

Um nun eine solche Hyperebene zu finden, werden die Trainingsdaten anhand der Prädiktoren als Punktvektoren dargestellt. Mittels der *Kernel-Methode* werden die Daten mit einer nichtlinearen Transformation in einen höherdimensionalen, dualen Raum abgebildet. Erst durch diese Transformation wird es möglich, nach einer lineare Trennungsebene zu suchen; dies liefert dann im ursprünglichen Raum eine nichtlineare Trennung. Durch eine Methode, die ähnlich zum Ansatz beim Perceptron ist, wird dann die optimale Hyperebene gesucht [BGV92].

Abbildung 11 zeigt ein Beispiel für eine Klassifikation mittels Support-Vektor Maschinen: Es gibt 19 Trainingsdaten, die zu jeweils einer der beiden Klassen {„Punkte“, „Kreise“} gehören. Links in der Abbildung ist eine Hyperebene eingezeichnet, die einen deutlich geringeren Abstand zu den Trainingsdaten hat als die optimale Hyperebene im rechten Teil. Diese Hyperebene ist diejenige Trennebene mit dem größten Abstand zu den Trainingsdaten und die drei *Support Vektoren* sind mit Pfeilen markiert.

Effizienz und Laufzeit von SVMs sind in der gleichen Größenordnung wie andere Klassifizierer und das Training einer SVM ist polynomiell in der Anzahl der Trainingsdaten [BGV92]. Allerdings können in der Praxis Fälle auftreten, bei denen bei der Transformation in den dualen Raum komplizierte Optimierungsprobleme (Maximierung einer quadratischen Funktion mit Nebenbedingungen) gelöst werden müssen und dies kann zu einem Speicherbedarf in Höhe von $O(N^2)$ und Zeitbedarf von $O(N^3)$ führen [HMS01].

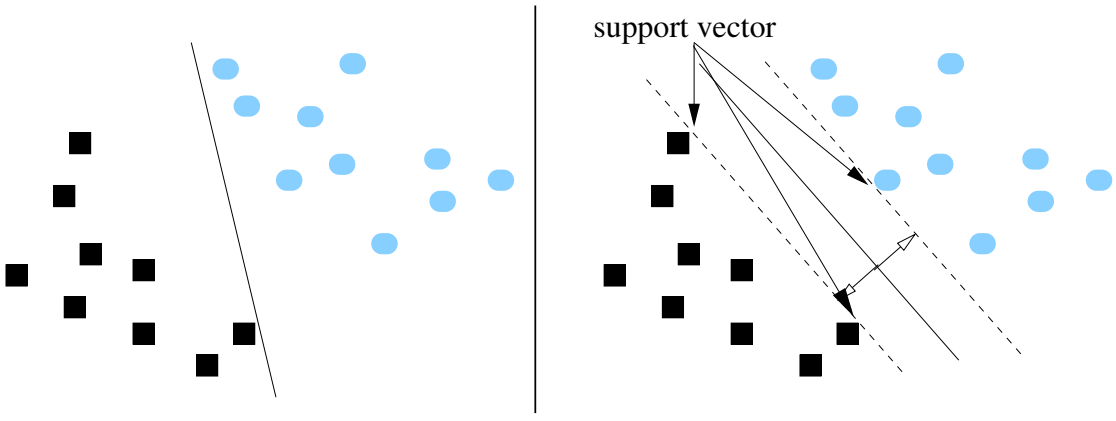

Abbildung 11: Beispiel für Klassifikation mittels SVM

3.4.3 Entscheidungsbäume

Entscheidungsbaum-Verfahren (engl.: *decision trees*) generieren zunächst anhand der Trainingsdaten einen Entscheidungsbaum und ordnen die zu klassifizierenden Objekte dann unter Berücksichtigung des Prädiktors $\mathbf{x}$ und den Verzweigungsregeln den Klassen zu.

Dazu wird ausgehend von den gegebenen Trainingsdaten ein Entscheidungsbaum aufgebaut: Die Blätter dieses Baumes repräsentieren die einzelnen Klassen, die Knoten stellen Zwischenzustände dar. Die Kanten zwischen den einzelnen Knoten bilden verschiedene Entscheidungen und bestimmen die Verzweigungsrichtung innerhalb des Baumes. Ein solcher Entscheidungsbaum ordnet neuen Objekten eine der Klassen zu und liefert so eine implizite Beschreibung der vorgegebenen Klassen

Zur Konstruktion des Entscheidungsbaums wird meistens die *pruning*-Methode benutzt: Jeder Knoten wird solange verzweigt, bis keine weitere Aufteilung mehr möglich ist. Dadurch entsteht ein Baum, der zu sehr an die Trainingsdaten angepasst ist (es kommt also zu *overfitting*) und deshalb wird im zweiten Schritt ein *pruning* durchgeführt. Dabei werden nacheinander je zwei Knoten wieder zusammengefasst, die die Fehlerrate bei den Trainingsdaten am wenigsten erhöhen. Alternativ können auch andere Methoden wie beispielsweise *cross-validation* benutzt werden, um einen Kompromiss zwischen der Komplexität des Modells und der Fehlerrate zu erreichen [HMS01].

In Abbildung 12 ist ein Beispiel für einen einfachen Entscheidungsbaum zu sehen: Anhand der beiden Merkmale x_1 und x_2 sollen Objekte klassifiziert werden, dabei sind zwei verschiedene Klassen möglich. Anhand der Trainingsdaten wurden die beiden Verzweigungsregeln $x_1 \leq 100$ und $x_2 \geq 42$ ermittelt. Ein zu klassifizierendes Objekt wird nun zur Klasse $c = 1$ zugeordnet, wenn $x_1 > 100$ oder $x_1 \leq 100 \wedge x_2 \geq 42$ ist; sonst wird das Objekt zur Klasse $c = 2$ zugeordnet. Wie man hieran sieht können Entscheidungsbäume einfach in Regelmengen (engl.: *association rules*) überführt werden, indem für jeden Pfad zu einem Blatt eine solche Regel aufgestellt wird.

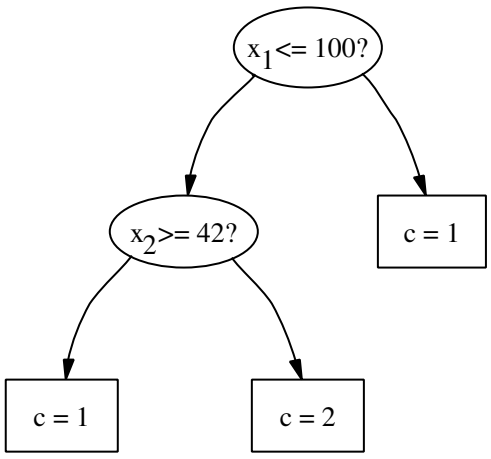

Abbildung 12: Beispiel für Entscheidungsbaum mit 2 möglichen Klassen

4 Evaluierung und Vergleich von Klassifikatoren

Im vorherigen Abschnitt wurden einige Verfahren vorgestellt, mit deren Hilfe man eine Klassifikation beziehungsweise Regression durchführen kann. Dazu wurde anhand der Trainingsmenge $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)\}$ mit Prädiktoren $\mathbf{x}_n$ und zugehörigen Klassen y_n ein Modell entwickelt und dieses dann zur Vorhersage von zukünftigen Werten benutzt. Wie kann man nun die Leistungsfähigkeit eines solchen Modells abschätzen?

Ein intuitiver Ansatz ist es, einige der statistischen Kennzahlen zu benutzen. Dies sind beispielsweise die Anzahl der falsch klassifizierten Trainingsdaten oder die *Fehlerrate*, also das Verhältnis der falsch klassifizierten Objekte zur Anzahl der Trainingsdaten. Benutzt man die Trainingsdaten, um eine Aussage über die Leistung eines Modells zu machen, so wird diese Vorhersage allerdings zu optimistisch sein, da das Modell auf diesen Daten basiert (vergleiche dazu Abbildung 5) – man benötigt also eine unabhängige *Testmenge*, um Aussagen über die Leistungsfähigkeit machen zu können.

Dabei sollte man auch die Kosten für falsch und richtig klassifizierte Daten berücksichtigen: Bei einer ärztlichen Untersuchung (z.B. in der Vorhersage von Tumoren) ist beispielsweise ein nicht diagnostizierter Tumor (*false negative*) wesentlich schlimmer als ein falsch diagnostizierter positiver Befund (*false positive*). Bei der Berechnung der Leistungsfähigkeit ist also möglicherweise eine Gewichtung der falsch klassifizierten Daten erforderlich. Die Bestimmung der Gewichte ist im Allgemeinen allerdings nicht einfach, da die tatsächlichen Kosten für ein falsch klassifiziertes Objekt in der Regel schwer zu bestimmen ist. Hat man jedoch entsprechende Gewichtungen gefunden, so müssen die statistischen Kennzahlen lediglich mit ihnen multipliziert werden.

In den folgenden Abschnitten werden verschiedene Verfahren vorgestellt, mit denen man die Leistungsfähigkeit von Modellen vergleichen kann: Abschnitt 4.1 stellt das Prinzip der *minimum description length*, ein auf der Informationstheorie beruhendes Verfahren, vor. Die Einteilung in Validierungs- und Testmenge wird in Abschnitt 4.2 erläutert. Abschnitt 4.3 beschreibt die *Kreuzvalidierung*, das in der Praxis am häufigsten verwendete Verfahren und in Abschnitt 4.4 wird das *Bootstrap*-Verfahren vorgestellt. Im Anschluss daran werden in Abschnitt 4.5 die *Receiver–Operating-Characteristics*–Kurven (ROC-Kurven) vorgestellt, mit deren Hilfe man die Sensitivität und Spezifität eines Modells visualisieren kann.

4.1 Minimum Description Length

Die Größe des Modells sollte bei der Bewertung der Leistungsfähigkeit berücksichtigt werden: Ein komplexes Modell ist stärker an die Trainingsdaten angepasst und führt zu „*overfitting*“; der Vorhersagefehler erhöht sich bei Verwendung der Testmenge. Dieser Sachverhalten wurde bereits in Abbildung 5 verdeutlicht. Ein Prinzip, um die Modellgröße zu bewerten, ist das Prinzip der minimalen Beschreibungslänge (engl.: *minimum description length*). Hierbei hat eine hohe Modellkomplexität einen negativen Einfluss auf die Güte des Modells, ein *overfitting* wird also „bestraft“. Das Prinzip besagt, dass unter Modellen mit gleicher Vorhersagequalität dasjenige Modell mit der kürzesten Beschreibungslänge als das Beste angenommen werden soll. Dieses Vorgehen ist analog zu *Ockham's Razor*: „Pluralitas non est ponenda sine neccesitate“ (William of Ockham, ca. 1285-1349) [HTF01]. Es beruht auf der Definition des *Informationsgehalts* nach Shannon und misst die Länge mit Hilfe der Wahrscheinlichkeit der Modellparameter. Die Beschreibungslänge setzt sich

dabei aus zwei Teilen zusammen:

$$length(M) = -\log p(y|\Theta, M, \mathbf{x}) - \log p(\Theta|M)$$

Dabei bezeichnet Θ die Parameter, um das Modell M zu beschreiben, und $\mathbf{x}$ ist der Prädiktor. Der zweite Term ist die minimale Länge, um die Parameter Θ des Modells zu kodieren, und der erste Term beschreibt die minimal erforderliche Länge der Kodierung, um die Abweichung zwischen dem Modell und den tatsächlichen Werten der Zielvariablen darzustellen. Diese Definition der minimalen Beschreibungslänge soll noch an einem kleinen Beispiel erläutert werden: Ein einzelner Wert y mit $y \sim \mathcal{N}(\Theta, \sigma^2)$, Modellparameter $\Theta \sim N(0, 1)$ und ohne Prädiktor (aus Gründen der Einfachheit) soll kodiert werden. Dann ist die Länge

$$\begin{aligned}
length(M) &= -\log p(y|\Theta, M) - \log p(\Theta|M) \\
&= -\log\left(\frac{1}{\sigma\sqrt{2\pi}} \cdot \exp\left(-\frac{(y-\Theta)^2}{2\sigma^2}\right)\right) - \log\left(\frac{1}{\sqrt{2\pi}} \cdot \exp\left(-\frac{\Theta^2}{2}\right)\right) \\
&= -\log\left(\frac{1}{\sqrt{2\pi}}\right) + \log(\sigma) - \log\left(\exp\left(-\frac{(y-\Theta)^2}{2\sigma^2}\right)\right) \\
&\quad -\log\left(\frac{1}{\sqrt{2\pi}}\right) - \log\left(\exp\left(-\frac{\Theta^2}{2}\right)\right) \\
&= \text{const} + \log(\sigma) + \frac{(y-\Theta)^2}{\sigma^2} + \frac{\Theta^2}{2}
\end{aligned}$$

Die Länge des Modells setzt sich also aus einem konstanten Teil, der sich durch Summation aller konstanten Terme ergibt, und einem variablen Teil zusammen: Die Größe des Parameters Θ des Modells geht zum einen quadratisch in das Ergebnis ein und zum anderen als quadrierte Abweichung zwischen dem Wert y und Θ.

4.2 Validierungs- und Testdaten

Falls man eine große Datenmenge $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)\}$ von Prädiktoren $\mathbf{x}_n$ und Zielvariablen y_n zur Verfügung hat, so geht man am besten wie folgt vor: Die Datenmenge wird in drei Teile aufgeteilt

1. eine Trainingsmenge, um damit passende Modelle zu finden
2. eine Validierungsmenge, um einen Schätzer für den Fehler der verschiedenen Modelle zu finden
3. eine Testmenge, um den Vorhersagefehler des gewählten Modells zu bestimmen

Bei genügend großer Datenmenge ist diese Partitionierung in drei voneinander unabhängige Menge kein Problem und als Erfahrungsregeln kann man 50% der Datenmenge als Trainings-, und je 25% als Validierungs- bzw. Testmenge benutzen [HTF01]. Dabei muss darauf geachtet werden, dass die Testmenge nur zum Test des endgültigen Modells benutzt wird, da es ansonsten zu einer zu positiven Einschätzung des Vorhersagefehlers kommt: Findet ein sogenanntes *„training on the testing data“* statt, das heisst werden Entscheidungen über die Struktur oder die Parameter des Modells auf Grund der Ergebnisse

für die Testdaten gefällt, so beeinflusst die Testmenge die Parameter und dies beeinflusst die abschliessende Evaluierung; diese fällt zu positiv aus.

Auch bei Anwendungen im Data-Mining kann es jedoch vorkommen, dass nur sehr wenige Trainingsdaten zur Verfügung stehen und deshalb eine solche Partitionierung nicht möglich ist. Dann muss die Trainingsmenge sowohl zum Training als auch zum Bewerten der Modelle dienen. Um in diesem Fall dennoch verlässliche Vorhersagen über die Leistungsfähigkeit der Modelle zu bekommen, wurden einige Verfahren entwickelt; diese werden in den beiden nächsten Abschnitten vorgestellt.

4.3 Kreuzvalidierung

Die vermutlich einfachste und am meisten benutzte Methode zur Schätzung des Fehlers bei einer Vorhersage ist die Kreuzvalidierung (engl.: *cross-validation*). Bei der Kreuzvalidierung wird die Gesamtmenge der Daten in k etwa gleich große Mengen aufgeteilt. Dann werden $k-1$ dieser Mengen als Trainingsmenge benutzt, die übriggebliebene Menge bildet die Testmenge. Dieses Verfahren wird so lange wiederholt, bis jede der k Mengen einmal als Testmenge benutzt wurde. Schließlich werden die Ergebnisse der k verschiedenen Durchgänge kombiniert um das endgültige Modell zu erhalten.

Doch wie groß soll man k wählen? Eine Möglichkeit für die Wahl des Parameters ist $k = |\text{Gesamtmenge}| = N$, die sogenannte *leave-one-out* Kreuzvalidierung. Dabei wird nur ein Objekt zum Testen des Modells benutzt, die übrigen Daten bilden die Trainingsmenge. Außerdem ist der Rechenaufwand im Allgemeinen hoch, da insgesamt N Durchführungen des Verfahrens nötig sind. Der Vorteil der *leave-one-out*-Methode ist allerdings der Determinismus, da kein Zufall einen Einfluss auf die Partitionierung hat. Außerdem wird die größtmögliche Datenmenge zum Training benutzt.

Natürlich gibt es auch noch andere Möglichkeiten zur Wahl von k, dabei muss man jedoch zwei Dinge beachten: Ein zu groß gewähltes k erhöht den Rechenaufwand; ein zu klein gewähltes k hingegen sorgt nur für eine kleine Anzahl von Trainingsdaten und kann deshalb zu einem schlechten Modell führen. In der Praxis wird meistens $k = 10$ (*engl.: 10-fold cross-validation*) oder $k = 5$ gewählt. Dies stellt einen guten Kompromiss zwischen Trainings- und Testdatenmenge dar und liefert stabile Ergebnisse.

4.4 Bootstrap

Das *Bootstrap*-Verfahren ist eine geglättete Version der Kreuzvalidierung mit einigen Änderungen, um einen besseren Bias zu erhalten. Es ist ein allgemeines Verfahren, um statistische Schwankungen zu berechnen und wurde 1979 von Bradley Efron vorgestellt [ET93]. Bootstrap beruht darauf, zufällige Stichproben der Größe N (*bootstrap samples* genannt) mit Zurücklegen aus der Menge der Trainingsdaten $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)\}$ zu ziehen, und diese dann als Trainingsdaten zu benutzen. Die so gewonnene Trainingsmenge ist also genauso groß wie die ürsprünglichen Trainingsdaten, kann aber manche $(\mathbf{x}_n, y_n)$ mehrfach enthalten. Das ganze Verfahren wird B mal wiederholt (beispielsweise $B = 100$) und man erhält B sogenannte *bootstrap*-Datensätze. Diese werden dann benutzt, um die Modelle zu trainieren und zu testen.

Doch wie kann man diese Bewertung nun durchführen, um den Fehler abschätzen zu können? Dazu kann man beispielsweise das Modell an einige der bootstrap-Datensätze anpassen und dann vergleichen, wie sich diese Veränderungen auf die Leistungsfähigkeit des Modells auswirkt. Dabei kommt es jedoch zu einer zu positiven Abschätzung der Leistungsfähigkeit des Modells, da die bootstrap-Datensätze zum Modellieren benutzt werden

und gleichzeitig Teil der Testdaten sind – sie sind also nicht unabhängig von den Trainingsdaten. Aus genau diesem Grund benutzt die Kreuzvalidierung auch die Partitionierung in Test– und Trainingsdaten.

Also braucht man einen anderen Ansatz: Man benutzt zum Testen nur diejenige Datensätze, bei denen ein bestimmtes $(\mathbf{x}_n, y_n)$ nicht im bootstrap-Datensatz vorkommt (ähnlich der leave-one-out–Methode). Die Wahrscheinlichkeit, dass ein bestimmter Datensatz nicht Teil der bootstrap-Datensätze ist, beträgt

$$\left(1 - \frac{1}{N}\right)^N \approx e^{-1} \approx 0,368,$$

da für jede einzelne Auswahl die Wahrscheinlichkeit $1 - \frac{1}{N}$ beträgt, dass ein bestimmter Datensatz *nicht* gezogen wird. Bei hinreichend großer Datenmenge enthält die bootstrap-Menge im Mittel $63,2\%$ aller Trainingsdaten, es bleiben als im Mittel $36,8\%$ der Daten als Testdaten, die im Training nicht eingesetzt wurden.

4.5 Receiver-Operating-Characteristic (ROC) – Kurven

Eine Möglichkeit der Visualisierung der Sensitivität und Spezifität von vorhersagenden Modellen stellen die *Receiver-Operating-Characteristic (ROC)*–Kurven dar, die zur Leistungsbewertung von Modellen zur Trennung von zwei Klassen benutzt werden können. Es werden also immer Ja-Nein–Entscheidungen betrachtet, beispielsweise *relevant/irrelevant* oder *krank/gesund*. Bei der Klassifikation wird jeweils ein Schwellenwert definiert, der die Zuordnung zu einer der beiden Klassen bestimmt; liegt $\hat{y}$ über diesem Schwellenwert, so erfolgt die Zuordnung zu der einen Klasse und sonst zur Anderen.

Zur Bestimmung der ROC-Kurve wird auf der x-Achse die relative Anzahl der falsch klassifizierten Werte (engl.: *false positive*; *1 - Spezifität*) und auf der y-Achse die relative Anzahl der richtig klassifizierten Werte (engl.: *true negative*; *Sensitivität*) gegeneinander aufgetragen; der Bereich eines solchen Diagramms ist also auf beiden Achsen der Bereich [0,1]. Variiert man nun den bei der Klassifikation benutzten Schwellenwert und trägt jeweils das bei einem bestimmten Schwellenwert resultierende Verhältnis von Sensitivität und Spezifität gegeneinander auf, so erhält man eine Kurve. Diese Kurve beginnt immer im Punkt [0,0], endet in [1,1], ist degressiv steigend und liefert eine Möglichkeit, die Güte verschiedener Modelle zu vergleichen. Der optimale Punkt einer solchen Kurve liegt in der linken oberen Ecke: Dort ist die relative Anzahl der false positives gleich null und gleichzeitig die relative Anzahl der true negatives gleich eins, das Modell klassifiziert also immer korrekt und liefert eine optimale Vorhersage. Verläuft die Kurve unterhalb dieses Punkts, so kommt es zu Fehlklassifikationen und diese kann man mittels ROC-Kurven visualisieren.

Abbildung 13 zeigt ein Beispiel für eine ROC-Kurve (Quelle: `http://vision.ai.uiuc.edu/mhyang/face-detection-survey.html`), bei der zwei Modelle zur Gesichtserkennung einander gegenübergestellt werden. Anhand von 24045 Bildern, darunter 472 Gesichter und 23573 andere Bilder, wurden zwei Support-Vektor-Maschinen trainiert. Die rot gezeichnete SVM benutzt eine lineare Kernel-Methode, die grün gefärbte SVM eine polynomielle Kernel-Methode zum Training. Anhand der Kurve sieht man, dass bei diesem Beispiel die polynomielle Kernel-Methode zum Training einer SVM der linearen Methode überlegen ist: Die grüne Kurve liegt weit links oben – also näher am optimalen Punkt [1,1] – als die rote Kurve.

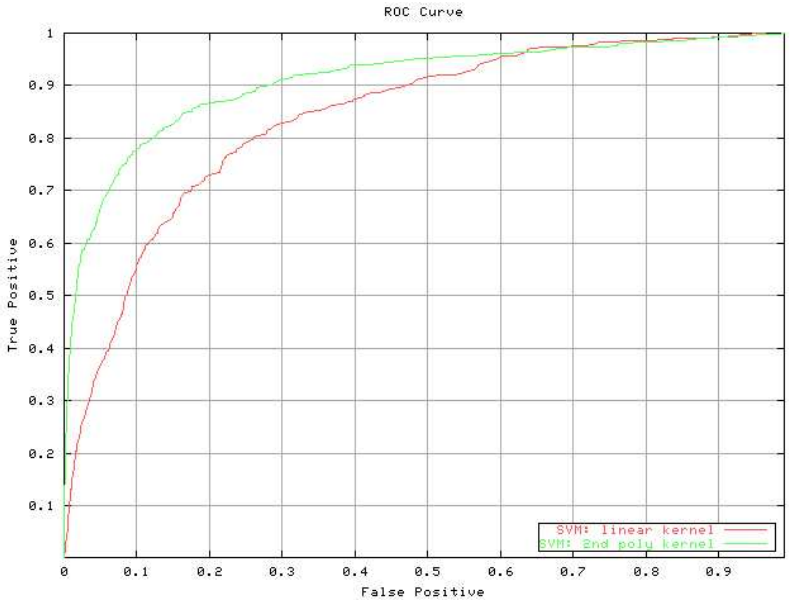

Abbildung 13: Beispiel für Receiver-Operating-Characteristic – Kurve

5 Zusammenfassung

Diese Seminararbeit stellt einige Verfahren zum Predictive Modeling – also der Vorhersagen zukünftiger Werte von Variablen anhand einer gegebenen Datenmenge – vor. Beim Predictive Modeling versucht man, durch vorgegebene Eigenschaften eines Prädiktors $\mathbf{x} = (x_1, \ldots, x_N)$ ein Modell – also eine allgemeine Beschreibung einer Menge von Daten – für eine Zielvariable y zu finden. Es wird also eine Funktion der Form

$$y = f(\mathbf{x}, \Theta)$$

gesucht, bei der die Parameter Θ des Modells gesucht werden. Dazu wurden einige Verfahren vorgestellt:

- Die logistische Diskriminanzanalyse benutzt eine Linearkombination der Prädiktor-Variablen x_n zur Vorhersage der logarithmierten Zielvariable y und wird deshalb auch log-lineares Modell genannt.
- Die Nächste-Nachbarn-Methode untersucht die k nächsten Nachbarn des Prädiktors $\mathbf{x}$ bezüglich eines bestimmten Distanzmaßes und ordnet $\mathbf{x}$ derjenige Klasse zu, die am häufigsten in der Nachbarschaft vorkommt.
- Das Naive Bayes Modell nimmt die stochastische Unabhängigkeit der Prädiktorvariablen x_n an und erreicht so eine Vereinfachung.
- Neuronale Netze bedienen sich der Grundidee von Neuronen: Das Signal an den Eingängen wird ab einem bestimmten Schwellenwert an den Ausgang weitergeleitet. Durch Kombination vieler solcher Neuronen entsteht ein neuronales Netz, das zur Klassifikation benutzt werden kann.

- Support-Vektor Maschinen führen eine Transformation in einen höherdimensionalen Raum durch und suchen dort nach optimal trennenden Hyperebenen.

- Entscheidungsbäume benutzen Bäume zur Klassifikation; die Blätter bilden die verschiedenen Klassen, Knoten repräsentieren Zwischenzustände und Kanten stellen verschiedene Entscheidungen dar.

Einige Verfahren, um Aussagen über die Leistungsfähigkeit eines Modells treffen zu können, wurden ebenfalls vorgestellt. Die beiden bekanntesten Verfahren sind die Kreuzvalidierung, bei der eine Partitionierung der Trainingsdaten in Trainings- und Testmenge vorgenommen wird, und das Bootstrap-Verfahren. Bei diesem werden zufällig Stichproben mit Zurücklegen aus der Menge der Trainingsdaten gezogen und dann als Trainings- und Testdaten benutzt.

Eine Möglichkeit zur Visualisierung der Leistungsfähigkeit verschiedener Modelle stellen Receiver-Operating-Characteristic – Kurven dar. Bei ihnen wird die relative Anzahl der false positives und true negatives gegenübergestellt und anhand des Kurvenverlaufs kann man Klassifikatoren evaluieren.

Beim Predictive Modeling muss man immer das Problem zwischen Bias und Varianz bedenken: Eine niedrige Varianz hat möglicherweise einen hohen Bias zur Folge und umgekehrt. Oft ist ein adaptiver Ansatz nötig, um einen Kompromiss zwischen diesen beiden Größen zu finden und ein möglichst gutes Modell zu erhalten, dass nicht zu sehr an die Trainingsdaten angepasst ist und auch einen niedrigen Vorhersagefehler bei Testdaten hat.

Immer mehr Daten werden produziert und abgespeichert. Das Erfassen und Speichern von Daten nützt wenig, die Daten müssen verarbeitet und dann verstanden werden, damit sie nützlich sind. Verfahren des Predictive Modeling tragen dazu bei, dass aus dieser Datenmenge *Wissen* wird.

Wir müssen wissen. Wir werden wissen.
- David Hilbert

Literatur

[BGV92] B. E. Boser, I. Guyon, and V. Vapnik, *A training algorithm for optimal margin classifiers*, Computational Learning Theory, 1992, pp. 144–152.

[DHS01] R.O. Duda, P.E. Hart, and D.J. Stork, *Pattern Classification*, Wiley, 2001.

[ET93] B. Efron and R.J. Tibshirani, *An Introduction to the Bootstrap*, Chapman & Hall, 1993.

[Fis36] R.A. Fisher, *The use of multiple measurements in taxonomic problems*, Annals of Eugenics, vol. 7, 1936, pp. 179–188.

[HMS01] D. Hand, H. Mannila, and P. Smyth, *Principles of Data Mining*, MIT Press, 2001.

[HT90] T. Hastie and R.J. Tibshirani, *Generalized Additive Models*, Chapman and Hall, 1990.

[HTF01] T. Hastie, R.J. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, Springer, 2001.

[McL92] G.J. McLachlan, *Discriminant Analysis and Statistical Pattern Recognition*, Wiley, 1992.