Wissensextraktion / Data Mining

SS 2015

Prof. Dr. Jürgen Cleve

Hochschule Wismar, Fakultät für Wirtschaftswissenschaften, PF 1210, 23952 Wismar
email : juergen.cleve@hs-wismar.de
KIWI-Homepage: http://www.wi.hs-wismar.de/kiwi

Stand: 26. Januar 2015

Inhaltsverzeichnis

Vorbemerkung

Wie ist das Skript aufgebaut? Das Skript entspricht im Prinzip einer typischen Einführung in das Gebiet des Data Minings:

- Einführung, Motivation, Grundlagen
- Datenvorbereitung
- Eigentliches Data Mining
- Bewertung der Resultate

Nur den Block zum *eigentlichen* Data Mining habe ich etwas anders aufgebaut. Meistens findet man in Lehrbüchern Abschnitte, die sich ausschließlich z.B. mit Klassifizierung beschäftigen. Dort werden auch alle Verfahren behandelt, die für Klassifizierung geeignet sind. Dies hat sich aus meiner Sicht als ungeschickt herausgestellt, da z.B. Entscheidungsbäume sowohl für Klassifizierung als auch die numerische Vorhersage geeignet sind.
Folglich habe ich die Aufteilung etwas anders vorgenommen. Zunächst wird im Kapitel 3 etwas zu Anwendungsklassen gesagt. Im Kapitel 4 wird auf geeignete Darstellungsmöglichkeiten für Data-Mining-Modelle eingegangen. Im Kapitel 5 werden dann die Algorithmen und Verfahrensklassen behandelt. Man wird also zum Thema Assoziation in allen 3 Kapiteln etwas finden.
Die Datenvorbereitung (Kapitel 6) habe ich, obwohl diese Phase natürlich in einem Data-Mining-Prozess die erste ist, weiter hinten platziert, da dies für den Leser (nach dem Beherrschen der DM-Algorithmen) wohl einfacher zu verstehen ist.
Das Standard-Buch zur Vorlesung ist [CL14].
Für Hinweise, Kritiken bezüglich dieses Skripts bin ich sehr dankbar.

1 Einführung

Data you don't need is never lost.
Ander's first negative Principle of Computers

1.1 Data Mining und Business Intelligence

Business Intelligence (BI) ist ein relativ neuer Begriff. Ausgangspunkt ist die Beobachtung, dass in Zeiten der Globalisierung und des Internets ein effektiver und effizienter Umgang mit dem in einem Unternehmen verfügbaren Wissen nicht nur ein Wettbewerbsvorteil, sondern für das Überleben wichtig ist.
Deshalb entstand eine Reihe von Techniken, Programmen etc., die sich diesem Problem widmeten. Unter Business Intelligence werden heute diese Techniken und Architekturen für eine effiziente Verwaltung des Unternehmenswissens zusammengefasst, natürlich einschließlich verschiedener Auswertungsmöglichkeiten. Die Aufgaben von BI sind also: Wissensgewinnung, -verwaltung und -verarbeitung. BI hat somit (aus Informatik-Sicht) viele Querbezüge zum Informationsmanagement, zu Datenbanken/Data Warehouse, zur Künstlichen Intelligenz, zum Data Mining (inkl. OLAP, Statistik).
Es gibt bis heute nicht **die** Definition des Begriffs *Business Intelligence*.
Unter Business Intelligence im engeren Sinn versteht man die Kernapplikationen, die eine Entscheidungsfindung direkt unterstützen. Hierzu zählt man z.B. das Online Analytical Processing (OLAP), die Management Information Systems (MIS) sowie Executive Information Systems (EIS).
Ein etwas weiterer BI-Begriff stellt die Analysen in den Vordergrund. Folglich gehören hierzu alle Anwendungen, bei denen der Nutzer Analysen durchführt bzw. vorbereitet. Neben den oben genannten Anwendungen zählt man nun auch z.B. das Data Mining, das Reporting sowie das analytische Customer Relationship Management dazu.
BI im weiten Verständnis umfasst nun alle Anwendungen, die im Entscheidungsprozess benutzt werden, also z.B. auch Präsentationssysteme sowie die Datenspeicherung und -verwaltung.
In der folgenden Abbildung aus [KMU06] sind die 3 Sichtweisen dargestellt.

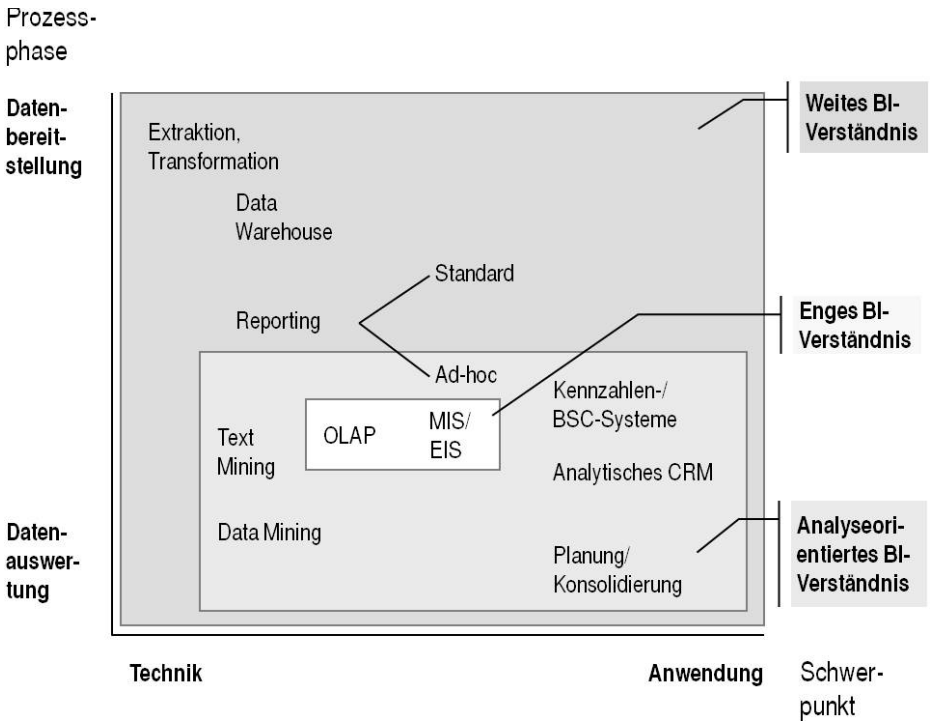


Abbildung Business Intelligence [KMU06]

Schwerpunkt dieser Vorlesung sind die Techniken zur Wissensextraktion, also dem Data Mining. Wir betrachten also nur einen kleinen Ausschnitt aus dem BI-Spektrum.

Für weiterführende Informationen sei auf [ML13] und [TSDK11] verwiesen.

1.2 Auswertung von Massendaten

Die Menge an verfügbaren Daten verdoppelt sich in immer kürzeren Abständen. Jedes Unternehmen, jede Institution sammelt freiwillig oder aufgrund rechtlicher Bestimmungen Unmengen an Daten. Zunächst einige Beispiele aus einem zwar älteren Buch (2000); die Situation dürfte sich aber bis heute weiter verschärft haben.

Industrielle Prozessdaten Zur Analyse der Altpapieraufbereitung in einer Papierfabrik stehen an jeder der 8 DeInking-Zellen jeweils 54 Sensoren zur Verfügung. 3800000 Messwerte pro Tag ([Run00]).

Umsatzdaten WalMart führte eine Warenkorb-Analyse durch, 20 Mio Transaktionen pro Tag, eine 24 TByte-Datenbank ([Run00]).

Genom-Daten In vielen Genom-Projekten wird versucht, aus den Genomen Informationen zu extrahieren. Das menschliche Genom enthält 60000-80000 Gene; 3 Milliarden DNA-Basen ([Run00]).

Bilder Die NASA nimmt mit ihrem *Earth Observing System* Oberflächenbilder der Erde auf. 50 GByte pro Sekunde ([Run00]).

Textinformationen Das WWW enthält eine Unmenge von Daten und Informationen.

Die Frage, die sich natürlich sofort ergibt: Was machen wir mit den ganzen Daten, die wir täglich sammeln? Zur Auswertung von umfangreichen Daten reicht „Draufschauen“ nicht mehr aus. Auch die Statistik stößt häufig an ihre Grenzen. Vielmehr braucht man weitere **Datenanalyse-Techniken**. Man sucht nach Mustern, nach Zusammenhängen etc., um so z.B. Vorhersagen für ein bestimmtes Kundenverhalten treffen zu können. Dies ist Gegenstand des *Data Minings*. Häufig weiß man am Anfang gar nicht, wonach man eigentlich sucht. Das können unbekannte Muster bzw. Abhängigkeiten oder auch die Einteilung von Objekten in unbekannte oder bekannte Klassen sein. Folgende reale Anwendungsbeispiele verdeutlichen die praktische Relevanz dieses hochaktuellen Gebiets:

- Erzeugen eines Entscheidungsbaums (generiert aus alten Kreditdaten) als Entscheidungshilfe für die Bewertung der Kreditwürdigkeit eines Kunden
- Generierung von Mustern von typischen Reisenden, um den Verkauf von Billigflügen oder -urlauben zu managen
- Windeln und Bier: Analyse des Kaufverhaltens ergibt, dass derjenige, der Windeln kauft, sehr häufig auch Bier kauft, aber nicht umgekehrt.
- Analyse der Gene bei Diabetes-Kranken, um typische Gene zu erkennen

Aber auch bei kleinen Problemen kann Data Mining hilfreich sein. In Abbildung 1 auf der nächsten Seite ist für die Klausur *Theoretische Informatik, SG WI, 2013/14* ein Entscheidungsbaum zur Vorhersage der Note dargestellt.

1.3 Ablauf einer Datenanalyse

Folgende Phasen unterscheidet man beim Data Mining:

Selektion Auswahl der geeigneten Datenmengen

Datenvorverarbeitung Skalierung, Ausreißer ...

Transformation Umwandlung in adäquate Datenformate

Data Mining eigentliches Suchen nach Mustern etc.

Interpretation / Evaluation Interpretation der Ergebnisse und Auswertung

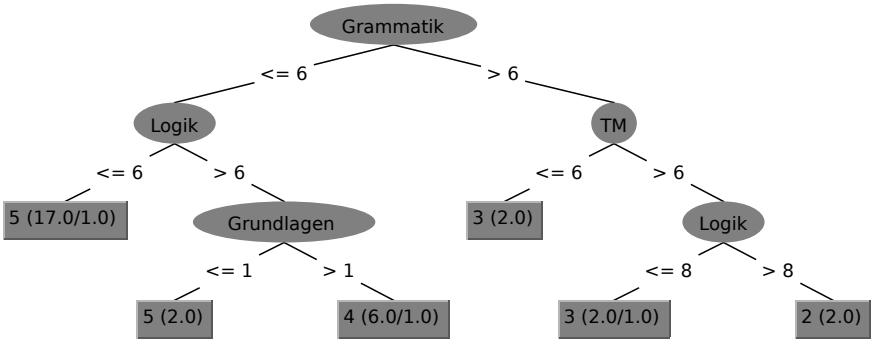


Abbildung 1: Entscheidungsbaum für die TI-Klausur 2013/14

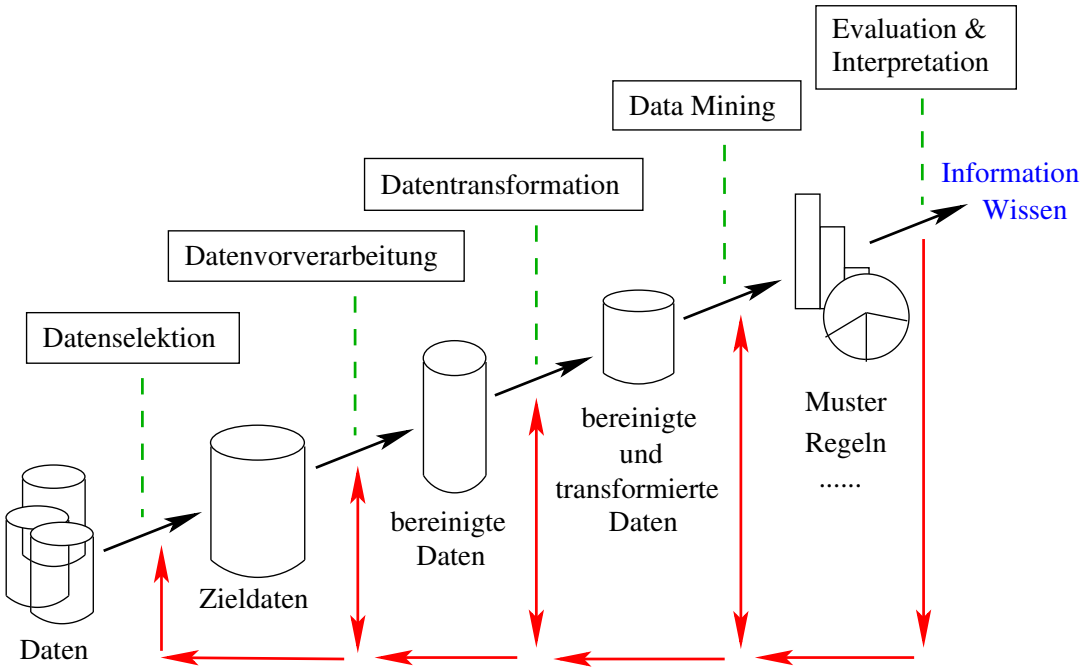


Abbildung 2: Ablauf eines Data-Mining-Prozesses [FPSS96]

In Abbildung 2 ist der Ablauf dargestellt.
Häufig bezeichnet man den Gesamtprozess auch als **Knowledge Discovery in Databases (KDD)** und nur das eigentliche Data Mining als solches.
Einige Erläuterungen zu den in Abb. 2 dargestellten Schritten: Im Vorfeld des Prozesses wird Wissen über den gewünschten Anwendungsbereich gesammelt sowie die Zielsetzung der Analyse festgelegt. Die Analyse von Rahmenbedingungen bildet einen weiteren Bestandteil der Vorbereitung. Diese lassen sich mit Hilfe eines Lösungsszenarios herauskristallisieren. Dabei wird festgestellt, welche Optionen der KDD-Prozess beinhalten kann und welche aus finanziellen, organisatorischen oder auch politischen Gründen nicht herangezogen werden.

CRISP Data-Mining-Modell

Ein zweites DM-Prozess-Modell ist das CRISP-Modell. Das CRISP-DM-Modell wurde durch NCR, Daimler-Benz, ISL, OHRA entwickelt. CRISP-DM steht für **C**ross **I**ndustry **S**tandard **P**rocess for **D**ata **M**ining (http://www.crisp-dm.org/). Ziel ist, einen Data-Mining-Prozess zu definieren und das berechnete Modell zu validieren.
Man geht von einem Lebenszyklus in 6 Etappen aus (vgl. Abb. 3 auf Seite 9):

1. Verstehen der Aufgabe
2. Verständnis der Daten

3. Datenvorbereitung
4. Data Mining (Modellbildung)
5. Evaluation
6. Einsatz im & Konsequenzen für Unternehmen

Man kann diese Etappen noch weiter untersetzen:

1. Verstehen der Aufgabe
   - Bestimmen der Firmenziele (Hintergrund, Ziele, Erfolgskriterien)
   - Assess Situation (welche Ressourcen, Restriktionen, Risiken, Begriffe, Kosten und Nutzen)
   - Formulierung der Data-Mining-Ziele (Ziele und Erfolgskriterien)
   - Projektplan erstellen
2. Verständnis der Daten
   - Initiale Daten sammeln
   - Daten beschreiben
   - Daten untersuchen
   - Datenqualität prüfen
3. Datenvorbereitung
   - Datenselektion
   - Daten säubern
   - Neue Daten/Attribute einfügen
   - Daten zusammenführen
   - Daten formatieren
4. Data Mining (Modellbildung)
   - Technik wählen
   - Tests konzipieren
   - Parameter für Verfahren setzen
   - Tests durchführen und ggf. mit modifizierten Parametern wiederholen
5. Evaluation
   - Erfolgskriterien auf Resultate anwenden
   - Prozess analysieren, Fehleranalyse
   - Nächsten Schritt festlegen, z.B. zurück zu Phase 3 oder 4
6. Einsatz, Konsequenzen für Firma
   - Einsatz planen
   - Monitoring und Unterstützung planen
   - Endbericht erstellen
   - Projekt analysieren

Die CRISP-Untergliederung unterscheidet sich von Abb. 2 auf der vorherigen Seite, wo die Punkte 1-2 und 6 des CRISP-Modells nicht enthalten sind. Der Schritt 3 des CRISP-Modells enthält die ersten 3 Phasen des obigen Modells. Wir werden uns im folgenden am in Abb. 2 dargestellten Ablauf orientieren.
Wir werden auch zwischen den Begriffen *Data Mining* und *KDD* nicht unterscheiden und diese häufig synonym verwenden.
Im folgenden wenden wir uns nun detaillierter den Teilphasen des Fayyad-Modells (Abb. 2) zu.

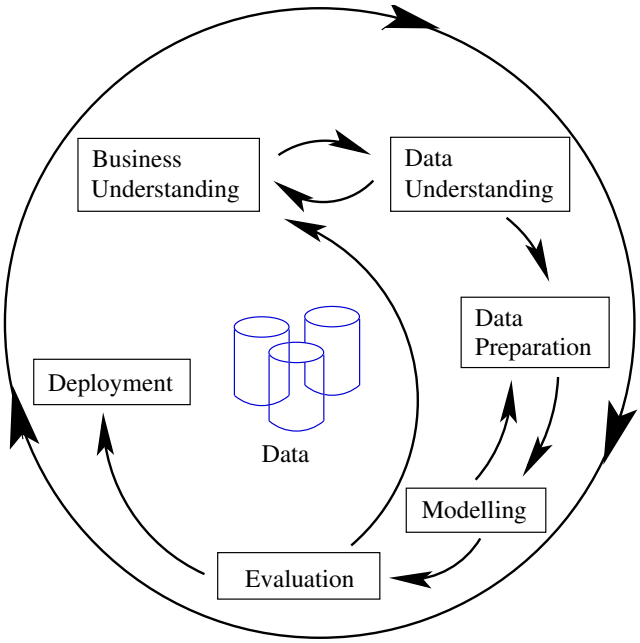


Abbildung 3: CRISP-Modell

1.3.1 Datenselektion

In der ersten Phase des KDD-Prozesses sind die Daten, die für die vom Anwender angeforderte Analyse benötigt werden oder geeignet erscheinen, zu bestimmen und aus den gegebenen Datenquellen zu extrahieren. Neben dem Basisdatenbestand können auch externe Daten für die Analyse herangezogen werden. So bieten z.B. Adressbroker Informationen an, mit denen Kunden oder Interessenten zusätzlich qualifiziert werden können. In der Phase der Datenselektion wird geprüft, welche Daten nötig und verfügbar sind, um das gesetzte Ziel zu erreichen. Können die selektierten Daten aufgrund technischer oder rechtlicher Restriktionen nicht in einen Zieldatenbestand überführt werden, ist die Datenselektion erneut vorzunehmen. Technische Restriktionen, die die Überführung in einen Zieldatenbestand verhindern, sind z.B. Kapazitäts- und Datentypbeschränkungen des Zielsystems oder fehlende Zugriffsrechte des Anwenders. Eine Möglichkeit, diese Probleme (zumindest zum Teil) zu umgehen, ist die Beschränkung der Auswahl auf eine repräsentative Teildatenmenge des Gesamtdatenbestands.

1.3.2 Datenvorverarbeitung

Da die Zieldaten aus den Datenquellen lediglich extrahiert werden, ist im Rahmen der Datenvorverarbeitung die Qualität des Zieldatenbestands zu untersuchen und – sofern nötig – durch den Einsatz geeigneter Verfahren zu verbessern. Aufgrund technischer oder menschlicher Fehler können die Daten operativer Systeme *fehlerhafte Elemente* enthalten. Man rechnet damit, dass bis zu 5% der Felder eines realen Datenbestands falsche Angaben aufweisen. Die Kenntnis der Schwächen der Analysedaten ist für die Qualität der Untersuchungsergebnisse wichtig. Die Anwender der Analysewerkzeuge müssen auf die Zuverlässigkeit und Korrektheit der Daten vertrauen können. Fehlerhafte Daten verfälschen möglicherweise die Resultate, ohne dass der Anwender von diesen Mängeln Kenntnis erlangt.
Fehlende Informationen verhindern eventuell die Berechnung wichtiger Kennzahlen. Die zunehmende Durchführung (teil-)automatisierter Datenanalysen hat eine erhöhte Anfälligkeit gegenüber Datenmängeln zur Folge, der durch geeignete Mechanismen zur Erkennung und Beseitigung solcher Schwächen zu begegnen ist. Eine häufige, leicht zu identifizierende Fehlerart besteht in *fehlenden Werten*. Zur Behandlung von fehlenden Werten stehen unterschiedliche Techniken zur Verfügung. Gängige Ersetzungsstrategien für numerische Attributausprägungen sind das Einsetzen eines Nullwertes, eines Mittel-, Maximal- oder Minimalwertes oder des Medians von Attributwerten innerhalb der Grundgesamtheit, einer repräsentativen Teilmenge oder einer Klasse. Bei nichtnumerischen At-

tributausprägungen kann es dagegen sinnvoll sein, die häufigste Attributausprägung einzusetzen. Eine weitere Möglichkeit, Attribute zu ersetzen, ist die nachträgliche manuelle Erhebung der fehlenden Daten. Das kann aber zu einem unverhältnismäßig hohen Aufwand führen. Eine weitere potentielle Fehlerart wird durch *Ausreißer* hervorgerufen. Dabei handelt es sich um Wertausprägungen, die stark vom Niveau der übrigen Werte abweichen. Bei diesen Ausprägungen kann es sich um korrekt erfasste Daten handeln, die damit Eingang in die Analyse finden oder aber um falsche Angaben, die nicht berücksichtigt werden dürfen und daher aus dem Datenbestand zu löschen sind. Die Erkenntnisse, die der Benutzer eines Data-Mining-Systems in dieser Phase über den Datenbestand gewinnt, können Hinweise auf die Verbesserung der Datenqualität der operativen Systeme geben.

1.3.3 Datentransformation

Die im Unternehmen verfügbaren Rohdatenbestände erweisen sich häufig in ihrer Ursprungsform als nicht für Data-Mining-Analysen geeignet. In der Phase der Datentransformation wird der analyserelevante Zieldatenbestand in ein Datenbankschema transformiert, das von dem verwendeten Data-Mining-System verarbeitet werden kann. Dabei können neue Attribute oder Datensätze generiert bzw. vorhandene Attribute transformiert werden. Dieser Schritt ist nötig, da Analyseverfahren spezifische Anforderungen an die Datenstruktur der Eingangsdaten stellen. Ziel der Transformation ist insbesondere die Gewährleistung invarianter Datendarstellungsformen (z.B. durch Übersetzung textueller Informationen in eindeutige Schlüssel oder Kodierungen) sowie die Einschränkung von Wertebereichen zur Verringerung der Anzahl zu betrachtender Ausprägungen (Dimensionsreduktion). Letzteres kann durch Verallgemeinerung von Attributwerten auf eine höhere Aggregationsstufe, z.B. durch Nutzung von Taxonomien oder durch Bildung von Wertintervallen geschehen, wodurch sich die Granularität der Daten ändert.

1.3.4 Data Mining

Liegen geeignete Datenbestände in akzeptabler Qualität vor, können die Analysen durchgeführt werden. In dieser Phase erfolgt die Verfahrensauswahl und deren Einsatz zur Identifikation von Mustern auf der Basis des vorbereiteten Datenbestandes. In einem ersten Schritt wird zunächst entschieden, welche grundlegende Data-Mining-Aufgabe (z.B. Klassifizierung oder Cluster-Bildung) vorliegt. Daran schließt sich die Auswahl eines geeigneten Data-Mining-Verfahrens an. Nach der Auswahl eines für die konkrete Problemstellung geeigneten Verfahrens muss dieses konfiguriert werden. Diese Parametrisierung bezieht sich auf die Vorgabe bestimmter methodenspezifischer Werte, wie z.B. die Festlegung minimaler relativer Häufigkeiten für einen Interessantheitsfilter, die Auswahl der bei der Musterbildung oder -beschreibung zu berücksichtigenden Attribute oder die Einstellung von Gewichtungsfaktoren für einzelne Eingabevariablen. Wenn eine zufriedenstellende Konfiguration gefunden wurde, kann mit der Suche nach interessanten Mustern in den Daten begonnen werden.

1.3.5 Evaluation und Interpretation

In dieser Phase des KDD-Prozesses werden die entdeckten Muster und Beziehungen bewertet und interpretiert. Diese Muster sollen den Anforderungen der *Gültigkeit, Neuartigkeit, Nützlichkeit* und *Verständlichkeit* genügen, um neues Wissen zu repräsentieren und einer Interpretation zugänglich zu sein. Letztere ist Voraussetzung für die Umsetzung der gewonnenen Erkenntnisse im Rahmen konkreter Handlungsmaßnahmen. Bei weitem nicht alle der aufgedeckten Muster erfüllen jedoch diese Kriterien. Die Analyseverfahren fördern häufig viele Regelmäßigkeiten zutage, die irrelevant, trivial, bedeutungslos oder bereits bekannt waren, aus denen dem Unternehmen folglich kein Nutzen erwachsen kann, oder die unverständlich bzw. nicht nachvollziehbar sind. Die Bewertung von Mustern kann anhand des Kriteriums der Interessantheit vollzogen werden. Folgende Dimensionen der *Interessantheit* sind sinnvoll:

- Die **Validität** eines Musters ist ein objektives Maß dafür, mit welcher Sicherheit das gefundene Modell (z.B. ein Muster oder eine Assoziationsregel) auch in Bezug auf neue Daten gültig ist.

- Das Kriterium der **Neuartigkeit** erfasst, inwieweit ein Muster das bisherige Wissen ergänzt oder im Widerspruch zu diesem steht.
- Die **Verständlichkeit** misst, wie gut eine Aussage von einem Anwender verstanden werden kann.
- Das Kriterium der **Nützlichkeit** eines Musters erfasst den praktischen Nutzen für den Anwender.

Die korrekte Interpretation von Data-Mining-Ergebnissen erfordert ein hohes Maß an Domänenkenntnissen. Die Interpretation dient dazu, das Domänenwissen des Anwenders effektiv zu verändern. Im Idealfall sollte ein Team von Experten aus unterschiedlichen Bereichen gebildet werden, um sicherzustellen, dass die Bewertung korrekt ist und die gewonnenen Informationen bestmöglich genutzt werden. Die Interpretationsphase lässt sich durch geeignete Präsentationswerkzeuge sowie durch die Verfügbarkeit zusätzlicher Informationen über die Anwendungsdomäne unterstützen. Typischerweise erfolgt in dieser Phase ein Rücksprung in eine der vorherigen Phasen. So ist meist eine Anpassung der Parameter oder die Auswahl einer anderen Data-Mining-Technik nötig. Es kann auch erforderlich sein, zur Datenselektionsphase zurückzukehren, wenn festgestellt wird, dass sich die gewünschten Ergebnisse nicht mit der benutzten Datenbasis erreichen lassen.

2 Data Mining – Grundlagen

If you file it, you'll know where it is but you'll never need it. If you don't file it, you'll need it but never know where it is.
Tillis's Organizational Principle

2.1 Begriffe

Definition 2.1 (Daten)**.** Ansammlungen von Zeichen mit der dazugehörigen Syntax werden Daten genannt.

Daten ist der Plural des lateinischen *Datum*, d.h. ein Informationselement. Man unterscheidet:

- unstrukturierte Daten (Bild, Text)
- semistrukturierte Daten (WWW-Seiten)
- strukturierte Daten (Datenbanken)

Definition 2.2 (Information)**.** Werden Daten mit einer Bedeutung gekoppelt, handelt es sich um Informationen.

Eine Information ist also die zweckbestimmte Interpretation von Daten durch den Menschen. Daten bestehen zunächst nur aus Fakten und werden erst dann zur Information, wenn sie im Kontext betrachtet werden und eine Bedeutung für den Menschen erhalten.

Definition 2.3 (Wissen)**.** Eine Information in Verbindung mit der Fähigkeit, diese zu benutzen, wird als Wissen bezeichnet.

Eine Information wird folglich erst dann zu **Wissen**, wenn man mit ihr etwas anzufangen weiß.

Definition 2.4 (Data Mining)**.** Beim Data Mining (Datenschürfen) handelt es sich um die Extraktion von Wissen aus Daten.

Data Mining ist die nichttriviale und automatische Suche nach Wissen in Massendaten. Man unterscheidet:

- Data Mining i.e.S. (strukturierte Daten)
- Web Mining (semistrukturierte Daten)
- Text Mining (unstrukturierte Daten)

2.2 Zwei Beispiele

In Warenhäusern werden an den Kassen die verkauften Waren elektronisch erfasst. Diese Daten werden in Datenbanken abgelegt, wodurch riesige Datenbestände über Verkaufsumsätze zur Verfügung stehen.
Mit Hilfe von Data-Mining-Verfahren können nun verschiedene Analysen auf diesen Daten durchgeführt werden:

- Welche Waren werden häufig gemeinsam mit anderen gekauft? Das Ergebnis dieser Frage kann dazu verwendet werden, Waren im Warenhaus diesem Zusammenhang entsprechend anzuordnen. So kann bei einem Kunden, der ursprünglich lediglich Produkt A kaufen wollte, auch das latente Bedürfnis nach Produkt B geweckt werden.
- Wann werden welche Waren in welchen Mengen gekauft? Mit einer guten Vorhersage von Verkaufszahlen lässt sich das Lagermanagement optimieren. Zum einen können die Lagerhaltungskosten verringert werden. Zum anderen kann aber auch die Kundenzufriedenheit und somit die Kundenbindung dadurch erhöht werden, indem Waren zum Nachfragezeitpunkt auch tatsächlich vorrätig sind.

Bei Versandhäusern werden persönliche Daten von Kunden erhoben und gesammelt – auch solche, die für den eigentlichen Bestellvorgang nicht unbedingt nötig sind (Haben Sie schon einmal in einem Versandhaus eingekauft, ohne Ihr Geburtsdatum und Ihre Telefonnummer angeben zu müssen?). Viele Versandhäuser versenden in regelmäßigen Abständen Werbebriefe an all ihre Kunden. Nicht selten landen diese ungelesen im Altpapier, da sich ein schlanker Mann nach der dritten Kleider-Reklame für mollige Frauen vom Versandhaus belästigt fühlt. So bleibt auch die Werbung ungelesen, die vielleicht sein Interesse gefunden hätte. Verschwendung – nicht nur aus ökonomischer, sondern auch aus ökologischer Sicht. Abhilfe dabei kann ein gezieltes Data Mining z.B. auf folgende Weise schaffen:

- Erkennen von Kundengruppen
- Zuschneiden von Werbeprospekten auf diese Kundengruppen
- gezieltes Versenden von Werbeprospekten an konkrete Zielgruppen

2.3 Interdisziplinarität

Es gibt eine ganze Reihe von Bezügen des Data Mining zu anderen Disziplinen. Data Mining ist höchst interdisziplinär (s. Abb. 4).

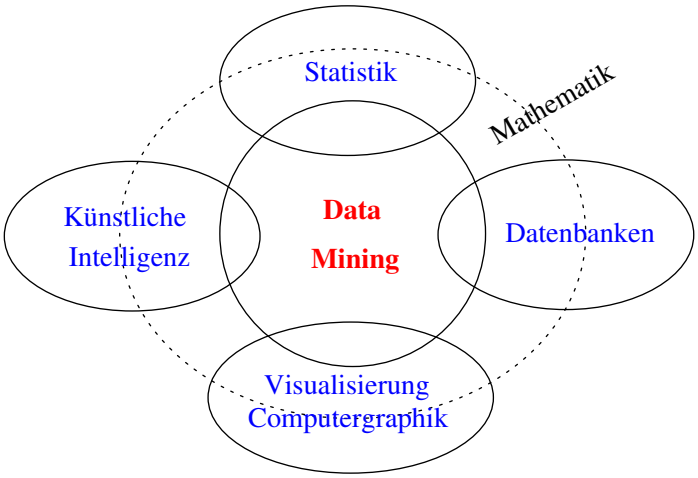


Abbildung 4: Interdisziplinarität

2.3.1 Datenbanken und Data Warehouses

Datenbanken bilden in vielen Fällen die Grundlage des Data Minings. Häufig wird in bereits existierenden Datenbeständen nach neu zu entdeckenden Zusammenhängen oder Auffälligkeiten gesucht. Im Sinne einer umfassenden Informationsbasis vereint ein *Data Warehouse* interne Datenbanken mit externen Daten. Zwei Ziele des Data Warehousing werden hier exemplarisch vorgestellt:

- Ein wesentliches Problem, das bei der Vereinigung von Daten verschiedener Quellen auftreten kann, betrifft die unterschiedliche Bezeichnung identischer Entitäten. So werden z.B. in drei verschiedenen Mathematikbüchern mindestens zwei verschiedene Formelzeichen für ein und denselben Sachverhalt verwendet. Dieses ist eins der Probleme, die ein Data Warehouse lösen soll.
- Ein weiteres Ziel eines Data Warehouse ist die Historisierung von Informationen, damit Änderungen auch zu späteren Zeitpunkten nachvollziehbar sind.

Neben Datenbanken können natürlich auch Textdateien oder WWW-Seiten Basis eines Data-Mining-Prozesses sein.

2.3.2 Expertensysteme

Expertensysteme versuchen, einen oder mehrere qualifizierte menschliche Experten bei der Problemlösung in einem abgegrenzten Anwendungsbereich zu simulieren. Sie enthalten große Wissensmengen über ein eng begrenztes Spezialgebiet. Sie berücksichtigen auch Faustregeln, mit denen Erfahrungen aus den Teilgebieten für spezielle Probleme nutzbar gemacht werden.

2.3.3 Maschinelles Lernen

Der Begriff *Lernen* umfasst viele komplexe Aspekte. Nicht jeder davon kann auf einem Rechner nachgebildet werden. Beim *Maschinellen Lernen* (engl. Machine Learning) geht es im Wesentlichen darum, Lernverfahren (computerbasiert) verfügbar zu machen, so dass das Programm aus Eingabeinformationen *Wissen* konstruieren oder verbessern kann.
Bei maschinellen Lernsystemen ist – wie auch in der menschlichen Psychologie – die einfachste Lernstrategie das Auswendiglernen. Dabei wird das präsentierte Wissen einfach in einer Liste oder Datenbank abgespeichert. Eine ebenso einfache Form des Maschinenlernens ist das unmittelbare Einprogrammieren des Wissens in den Sourcecode des entsprechenden Programms.
Dies ist jedoch nicht das, was in der Künstlichen Intelligenz mit Maschinellem Lernen gemeint ist. Hier wird mehr ein Verständnis von Zusammenhängen und Hintergründen (z.B. das Erkennen von Mustern oder Abhängigkeiten) angestrebt, sprich das Erzeugen von *neuem* Wissen (induktives Lernen).

2.3.4 Statistik

Statistik ist ein weiterer wichtiger Baustein des Data Mining. In der Datenverarbeitung steht in vielen Fällen neben der Genauigkeit der Ergebnisse auch die Geschwindigkeit sowohl der Vorbereitung des Verfahrens als auch der Entscheidungsfindung selbst im Vordergrund. Nicht immer ist es möglich oder sinnvoll, ein maschinelles Lernverfahren zu entwickeln und anzuwenden. Manchmal bringen auch schon statistische Lösungen einen ausreichenden Erfolg. Des weiteren können statistische Verfahren dabei helfen zu erkennen, ob Data Mining überhaupt zu einem gewünschten Ergebnis führen könnte.

2.3.5 Visualisierung

Visualisierung spielt eine große Rolle, da Data Mining meistens zur Entscheidungsfindung oder -unterstützung eingesetzt wird. Entscheidungen werden nicht immer von den Personen getroffen, die direkt am Prozess des Data Minings beteiligt sind. Gefundenes Wissen anschaulich und nachvollziehbar darzustellen, ist folglich für die Akzeptanz der Resultate wichtig.

2.4 Datentypen

Welche Arten von Daten können durch DM-Verfahren behandelt werden? Man unterscheidet folgende wichtige Datentypen:

nominal Nominale Daten unterliegen keinerlei Rangfolge. Sie können lediglich nach dem Kriterium *gleich* bzw. *nicht gleich* sortiert werden.

ordinal Ordinale Daten haben zumindest eine Ordnungsrelation (wie $<$).

metrisch Metrische Daten besitzen alle Ordnungsmerkmale der reellen Zahlen. Man kann mit ihnen rechnen.

Man kann diese Einteilung noch verfeinern. Z.B. kann man auch Intervall-basierte Datentypen betrachten. Gute Beispiele findet man z.B. in [Pyl99, S. 67].
In Tabelle 1 auf der nächsten Seite sind Beispiele für die Datentypen angegeben. Beachten Sie, dass die Grenze zwischen ordinal und metrisch durchaus fließend sein kann. Wir könnten z.B. die Schulnoten oder die Erdbebenstärke auch als metrische Attribute auffassen. Insbesondere kann man leicht ordinale in metrische Daten umwandeln.
Betrachten wir ein Attribut mit den Ausprägungen *klein*, *mittelgroß*, *groß*, *sehr groß*, so können wir dies leicht in ein metrisches Attribut umwandeln, indem wir die Werte durch Zahlen ersetzen:

nominal	ordinal	metrisch
Farbe	Schulnote	Fläche
Beruf	Schuhgröße	Geschwindigkeit
Familienstand	Erdbebenstärke	Körpergröße
Staatsangehörigkeit	Altersgruppe	Kinderzahl

Tabelle 1: Beispiel Datentypen

- *klein* $\rightarrow 0$
- *mittelgroß* $\rightarrow 0{,}3$
- *groß* $\rightarrow 0{,}7$
- *sehr groß* $\rightarrow 1$

Beachten Sie aber, dass wir durch die (willkürliche) Wahl der Zahlen den Abstand zwischen den Werten festlegen und somit eventuell das Resultat beeinflussen.

2.5 Abstands- und Ähnlichkeitsmaße

Um Datensätze miteinander vergleichen zu können, braucht man Maße, die die Ähnlichkeit von Datensätzen messen können.

- dist(v, w): Abstand zweier Datensätze
- simil(v, w): Ähnlichkeit zweier Datensätze

Zur Bestimmung der Ähnlichkeit wird meist der Abstand herangezogen:

$$\text{simil}(v, w) = f(\text{dist}(v, w))$$

Eine Abstandsfunktion (auch **Distanzfunktion** genannt) sollte folgende Eigenschaften erfüllen:

- $\text{dist}(x, x) = 0$
- $\text{dist}(x, y) = \text{dist}(y, x)$
- $\text{dist}(x, y) \leq \text{dist}(x, z) + \text{dist}(z, y)$

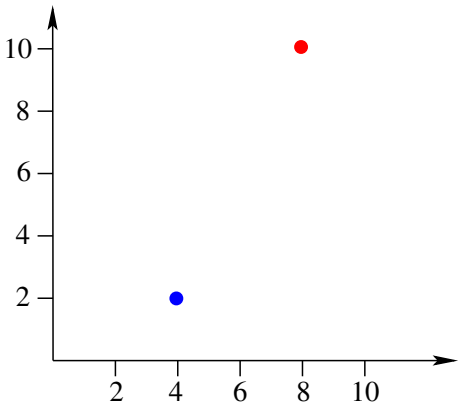
Folgende typische Distanzfunktionen gibt es:

- Hamming-Distanz: $\text{dist}_H(v, w) = \text{count}_i(v_i \neq w_i)$
- Euklidische Distanz: $\text{dist}_E(v, w) = \sqrt{\sum_i (v_i - w_i)^2}$
- Manhattandistanz: $\text{dist}_S(v, w) = \sum_i (|v_i - w_i|)$
- Maximumdistanz: $\text{dist}_{\text{Max}}(v, w) = \max_i (|v_i - w_i|)$
- ...

Die Maximumdistanz wird häufig auch als Tschebyscheff-Distanz bezeichnet.
Es gibt eine Erweiterung der euklidischen Distanz, die *Minkowski-Distanz*:

$$\text{dist}_{Minkowski}(v, w) = \sqrt[p]{\sum_i |v_i - w_i|^p}$$

Diese ist nur auf numerische Attribute anwendbar.
In Abb. 5 auf der nächsten Seite ist ein Beispiel für die 4 Distanzen gegeben.
Beachten Sie, dass man bei den Beispielen im Kapitel 5 durchaus auch andere Distanzfunktion verwenden kann. Man muss also bei numerischen Attributen nicht zwingend immer die Euklidische Distanz nehmen.
Einige weitere Ähnlichkeitsmaße werden in [Run10] behandelt, z.B. Cosinus, Überlapp, Dice, Jaccard.



$\text{dist}_H = 2$, $\text{dist}_E \approx 8.9$, $\text{dist}_{\text{Man}} = 12$, $\text{dist}_{\text{Max}} = 8$

Abbildung 5: Beispiel Distanzen

Aufgabe 2.1 (Distanz)**.** Wenn man die Schritte des Königs auf einem Schachbrett als Distanz wählt, welchem Distanzbegriff entspricht das? Und welchem Begriff entspricht die Anzahl der Felder, die der Turm passieren müsste?

Aufgabe 2.2 (Distanz)**.** Berechnen Sie die Distanz zwischen den Punkten (0,1,2), (1,5,3) und (4,-2,3). Verwenden Sie dabei alle 4 aufgeführten Distanzfunktionen.

Aufgabe 2.3 (Distanz)**.** Suchen Sie weitere Abstandsmaße.

Aufgabe 2.4 (Datentypen)**.** Welchen Typ von Daten haben wir bei den Postleitzahlen: nominal, ordinal, metrisch?

2.6 Weitere Grundbegriffe

Data Mining beginnt mit einer gegebenen Menge von Beispielen. Diese nennt man **Instanzenmenge** oder **Beispielmenge** E. Auf einer Teilmenge von E lernt man, d.h. man benutzt diese, um z.B. Klassen zu bilden oder Assoziationen herzustellen. Dies ist die **Trainingsmenge** $T \subset E$. Eine weitere Teilmenge von E (meist $E \setminus T$) benutzt man, um das Gelernte zu validieren. Diese Menge heißt folglich **Validierungsmenge** (oder Testmenge) $V \subset E$.
Folgende (Lern-)**Strategien** gibt es:

- Nicht-überwachtes Lernen: Die zu entdeckenden Muster sind unbekannt (vgl. Clustern, Abschnitt 3.2).
- Überwachtes Lernen: Es werden Beispiele vorgegeben, z.B. Beispiele für Nadel- oder Laubbäume (vgl. Klassifikation, Abschnitt 3.1).

3 Anwendungsklassen

All great discoveries are made by mistake.
Young's Law

3.1 Klassifikation

Ziel der Klassifikation ist die Einteilung eines **Gegenstandsbereichs** (z.B. Kunden) in **Klassen** (normale / sehr gute Kreditwürdigkeit). Anhand einer vorgegebenen Trainingsmenge von bereits bekannten Objekten wird ein Modell (vgl. Abb. 6) aufgebaut. Dieses wird dann an Testdaten geprüft (vgl. Abb. 7) und gegebenenfalls korrigiert, bis es auf den Testdaten eine nur noch geringe Fehlerrate erfüllt. Dabei sind sowohl die Klassen als auch die Zugehörigkeit der Trainingsobjekte zu einer dieser Klassen vorher bekannt. Man spricht daher auch von *überwachtem Lernen*. Ist die gelernte Klassifikation gut genug, kann man unbekannte Objekte aufgrund ihrer Eigenschaften in die Klassen einordnen (vgl. Abb. 8).

Trainingsproben $\Longrightarrow$ **Klassifikations-algorithmus** $\Downarrow$ **Klassifikations-regeln** (z.B.)
WENN Alter = 31...40 UND Einkommen = hoch DANN Kreditwürdigkeit = sehr gut

Name	Alter	Ein-kommen	Kredit-würdigkeit
Adam	≤ 30	niedrig	normal
Beate	≤ 30	niedrig	sehr gut
Clemens	31...40	hoch	sehr gut
Diana	> 40	mittel	normal
Egon	> 40	mittel	normal
Frank	31...40	hoch	sehr gut
...	...	...	...

Abbildung 6: Klassifikation – Lernphase

Testproben

Name	Alter	Ein-kommen	Kredit-würdigkeit	Bewertung durch Regeln
Gerda	31...40	hoch	sehr gut	sehr gut
Hanno	31...40	hoch	normal	sehr gut
Inge	> 40	hoch	sehr gut	...
...	...	...	...	...

Abbildung 7: Klassifikation – Testphase

Neue Daten

Name	Alter	Ein-kommen	Kredit-würdigkeit	Bewertung durch Regeln
Jochen	31...40	hoch	??	sehr gut
Karl	...	...	...	...
...	...	...	...	...

Abbildung 8: Klassifikation – Anwendungsphase

Die Rolle des Lernverfahrens besteht darin zu erkennen, welche der gegebenen Eigenschaften (Alter, Einkommen, ...) bestimmend für die Klasseneinordnung (Kreditwürdigkeit) sind.
Ergebnis der Klassifikation muss nicht in jedem Fall eine eindeutige Klasse sein. Oftmals ist auch die Wahrscheinlichkeit für alle Klassen von Interesse.

Beispiel 3.1 (Klassifikation).

- Vorhersage, ob ein bestimmter Kunde auf eine Werbeaktion reagieren wird
- Zeichenerkennung (Kfz-Kennzeichen, Adressen etc.)
- Vorhersage von Erdbebenwahrscheinlichkeiten

Einige Verfahren:

- Induktion von **Entscheidungsbäumen**
- Induktion von **Klassifikationsregeln**
- Neuronale **Feedforward Netze**
- **Bayes-Verfahren**

3.2 Clustering

Ziel der Clusteranalyse ist es, eine gegebene Instanzenmenge E ($E \subseteq X$) in **verschiedene Teilmengen** (Cluster) zu zerlegen. Die Individuen innerhalb eines Clusters sollen dabei möglichst ähnlich sein, wohingegen Individuen verschiedener Cluster möglichst unähnlich sein sollen.
Neben der Instanzenmenge selbst sind für das Clustering eine *Distanz-/Abstandsfunktion* und eine *Qualitätsfunktion* nötig (vgl. Abschnitt 7).
Das Ziel des Clusterings lässt sich mit Hilfe einer Abstandsfunktion `dist` so formulieren, dass der Abstand der Individuen innerhalb eines Clusters kleiner als der Abstand zu den Individuen anderer Cluster sein soll.

$$\forall\, C_i, C_j, (i \neq j) \in C : \forall\, x_k, x_l \in C_i, x_m \in C_j : \mathrm{dist}(x_k, x_l) < \mathrm{dist}(x_k, x_m)$$

Die Qualitätsfunktion beschreibt die Güte des Clusterings.

$$\mathrm{quality}(C = \{C_1 \subseteq E, \ldots, C_k \subseteq E\}) \rightarrow \Re$$

Meistens basiert die Qualitätsfunktion auf der Abstandsfunktion `dist` bzw. auf einem Ähnlichkeitsmaß `simil` (vgl. Abschnitte 2.5 und 7).
In Abbildung 9 sind 2 Beispiel-Clusterungen dargestellt.

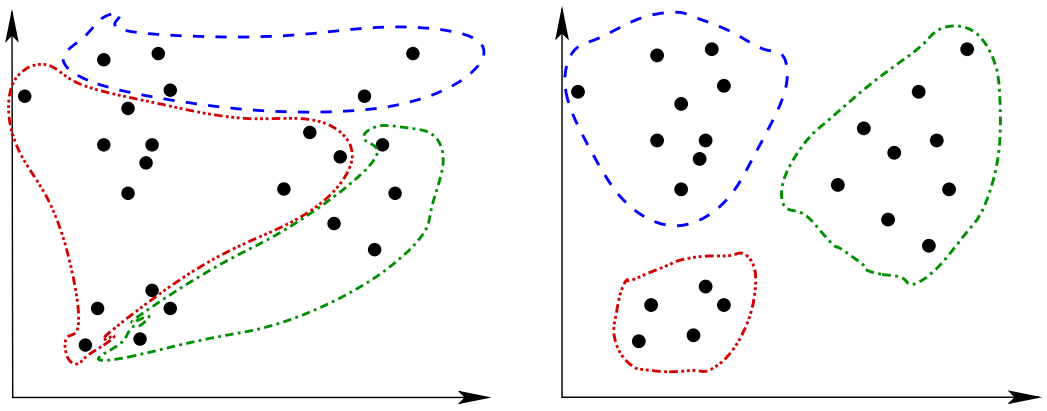


Abbildung 9: Schlechtes und gutes Clustering

Beispiel 3.2 (Clustern).

- Finden homogener Kundengruppen zur gezielten Angebotsgestaltung
- OCR: Finden von Buchstabengruppen, die ähnliche Bilder haben, um spezialisierte Klassifikatoren zu entwickeln

Einige Verfahren:

- **k-Means-Algorithmus**
- **Selbstorganisierende Neuronale Netze** (z.B. Kohonen Map, Growing Neural Gas)

3.3 Numerische Vorhersage

Ziel der numerischen Vorhersage ist die **Approximation** einer Funktion aus Beispielen. Dabei sollen aus einer Reihe bekannter Instanzenbeschreibungen und Funktionswerte die Werte zukünftiger, bislang unbekannter Instanzenbeschreibungen berechnet werden.
Von einer Funktion $y = f(x)$ ist lediglich eine Instanzenmenge E bekannt, die aus Instanzenbeschreibungen $X = \{x_1, \ldots, x_n\}$ sowie zugehörigen Zielwerten $Y = \{y_1, \ldots, y_n\}$ besteht. Gesucht ist nun eine Funktion $y' = h(x)$, die die Zusammenhänge zwischen den Instanzenbeschreibungen und Zielwerten möglichst genau widerspiegelt. Der Fehler $\text{error}(f(x), h(x))$ zwischen berechnetem und tatsächlichem Wert soll also minimiert werden.

Beispiel 3.3 (Numerische Vorhersage)**.**

- Vorhersage von Verkaufszahlen zur Lageroptimierung
- Vorhersage von Aktienkursen
- Zeitreihenanalyse

Einige Verfahren:

- **Lineare Regression**
- **Regressionsbäume**
- **Neuronale Netze** (Feed forward)

3.4 Assoziationsanalyse

Mit dem Aufkommen der Barcodetechnik und der immer größer werdenden Verbreitung von Lesegeräten ist es möglich geworden, das Kaufverhalten von Kunden ohne eine unangemessene Steigerung des Arbeitsaufwands zu dokumentieren. Große Datenmengen fallen an. Diese Daten gilt es zu analysieren, um Antworten auf Fragen wie: „Wie ordne ich meine Waren optimal an?“, „In welche Kategorien lässt sich die Kundschaft einordnen?“ oder „Welche Artikel sollten aus dem Sortiment genommen werden?“ zu finden. Sinn dieser Warenkorbanalyse ist also die Gewinnsteigerung bzw. die Verbesserung der Kundenzufriedenheit.
Mit der Anwendung in anderen Bereichen entwickelte sich die Warenkorbanalyse schließlich zur *Assoziationsanalyse*. Diese ist der Versuch, Regionen bzw. Datenbereiche in einer Datenbank zu identifizieren und zu beschreiben, in denen mit hoher Wahrscheinlichkeit mehrere Werte *gleichzeitig* auftreten.
Der wesentliche Unterschied zur Warenkorbanalyse besteht in der großen Zahl der Anwendungsgebiete. Denkbare Anwendungsmöglichkeiten sind beispielsweise die Risikoabschätzung in der Versicherungsbranche oder die Analyse der Spielweise einer gegnerischen Fußballmannschaft.

Beispiel 3.4 (Assoziationsanalyse)**.** Ein im Internet präsentes Versandhaus analysiert seine Verkaufsdaten und stellt fest, dass ein Produkt A häufig gemeinsam mit einem anderen Produkt B gekauft wird. Das Unternehmen bietet dann in Zukunft automatisch beim Kauf von A auch das Produkt B an.

Mit Hilfe der Assoziationsanalyse können Zusammenhänge zwischen verschiedenen Waren erkannt und das Kundenverhalten analysiert werden. Die Assoziationsanalyse ist ein vorhersagendes Data-Mining-Verfahren, d.h. es analysiert die Daten, um Regelmäßigkeiten zu identifizieren und das Verhalten neuer Datensätze vorherzusagen.
Einige Verfahren:

- **A-Priori-Verfahren**
- **ART-Netze**

3.5 Text Mining

Text Mining beschäftigt sich mit der **Analyse von Textdokumenten**. Texte sind im Gegensatz zu Datenbanken und Web-Seiten unstrukturiert. Häufig ist man an einer Klassifizierung des Dokuments interessiert, also z.B. nach Themengebiet und fachlichem Niveau. Ein Ansatz ist, zunächst relevante Begriffe aus dem Dokument zu extrahieren, um anhand dieser das Dokument einzuordnen. Auf Text Mining wird im Rahmen dieses Skripts nicht näher eingegangen.

3.6 Web Mining

Anwendungen des Data Minings, die das Internet als Datenquelle für die Mustererkennung heranziehen, werden unter dem Themengebiet des Web Minings zusammengefasst. In Abhängigkeit von der inhalts- oder nutzungsorientierten Analyse des World Wide Web (WWW) lassen sich die Teilgebiete *Web Content Mining* und *Web Usage Mining* voneinander abgrenzen.

Web Content Mining befasst sich mit der Analyse von den im WWW befindlichen Daten. Dazu gehören textuelle und multimediale Informationen jeglichen Formats und auch die Verbindungen (Links) zu den Nachbarseiten.

Web Usage Mining dagegen beschäftigt sich mit dem Verhalten von Internet-Nutzern. Bei dieser Ausprägungsform des Web Mining werden Data-Mining-Methoden auf die Protokolldateien des Webservers angewandt, um Aufschlüsse über Verhaltensmuster und Interessen der Online-Kunden zu erhalten. Eine Ausprägungsform des Web Usage Mining, bei der sich die Analyse ausschließlich auf die Protokolldateien des Web-Servers beschränkt, wird als **Web Log Mining** bezeichnet. Sofern neben den Protokolldateien noch weitere Datenbestände in den Mustererkennungsprozess einfließen, wird diese Ausprägung als **Integrated Web Usage Mining** bezeichnet. Die Taxonomie des Web Minings ist in Abbildung 10 dargestellt.

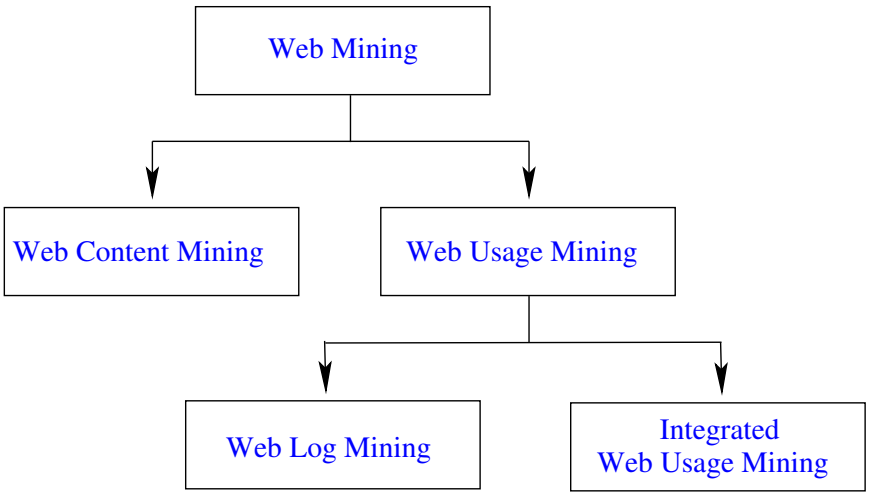


Abbildung 10: Web Mining

Insbesondere das Web Log Mining hat an Bedeutung gewonnen, da sich das Internet in den letzten Jahren zu einem bedeutenden Medium für die Abwicklung geschäftlicher Prozesse entwickelt hat. Da die Webpräsenz eines Unternehmens immer häufiger den ersten Kontakt zwischen einem potentiellen Kunden und dem Unternehmen herstellt, hat diese natürlich mittlerweile einen hohen Stellenwert. Gerade in einem so stark umkämpften Markt wie dem Internet ist es von immenser Bedeutung, sich Wettbewerbsvorteile gegenüber der Konkurrenz zu verschaffen, denn der Kunde ist nur einen Mausklick von dem nächsten Angebot entfernt. Unternehmen, die über eigene Webpräsenzen verfügen, sammeln automatisch Nutzungsdaten in sogenannten Logdateien über die virtuellen Besuche ihrer (potentiellen) Kunden. Die hierbei anfallenden Daten werden aber häufig nur unzureichend verwertet. Da sich die Nutzungsdaten aus wirtschaftlichem Hintergrund auf das Verhalten von Marktpartnern beziehen, sind sie zur Unterstützung wirtschaftlicher Entscheidungen von großer

Bedeutung. Das Management muss wissen, wer die Website besucht und – was noch wichtiger ist – wer etwas kauft bzw. warum nichts gekauft wird. Websites werden heute als Investition gesehen und müssen ihre Notwendigkeit, wie jede andere Marketinginvestition, begründen. Je mehr man darüber weiß, wie viele Kunden die Website besuchen, wer sie sind und für welche Bereiche sie sich interessieren, desto mehr wird die Website davon profitieren. Werden diese Informationen zur Optimierung der Website benutzt und mit anderen gängigen Marketingaktionen verbunden, kann der gesamte WWW-Auftritt stark verbessert werden.
Web Log Mining beschäftigt sich mit der Untersuchung von Data-Mining-Verfahren zur Auswertung von internetbasierten Nutzungsdaten (Logdateien) und deren Nutzen zur Unterstützung unternehmerischer Entscheidungen im Kontext der Optimierung von Internetangeboten.

4 Wissensrepräsentation

Quality is inversely proportional to the time left for completion of the project.
Wright's first law of quality.

Nachdem wir uns damit beschäftigt haben, für welche Aufgaben Data Mining eingesetzt werden kann (Kapitel 3), befassen wir uns – bevor wir auf die Verfahren (Kapitel 5) eingehen – nun mit der Frage, wie man Resultate unserer DM-Analysen *darstellen* kann. Dies ist nicht trivial, da man z.B. einen Entscheidungsbaum durchaus unterschiedlich darstellen kann, natürlich zunächst als Baum, aber auch durch Regeln oder eine Tabelle. Im folgenden betrachten wir also unterschiedliche Formen der Wissensdarstellung im Kontext *Data Mining*.

4.1 Entscheidungstabellen

Eine Entscheidungstabelle ist die tabellarische Auflistung möglicher Bedingungen (Eingaben) und des gewünschten Ergebnisses (Ausgabe), das jeder Bedingung entspricht.

Beispiel 4.1 (Entscheidungstabelle)**.** In Tabelle 2 ist eine Entscheidungstabelle für das Golfspiel gegeben. Wann wird gespielt?

outlook	temperature	humidity	windy	play
sunny	hot	high	false	no
sunny	hot	high	true	no
sunny	mild	high	false	no
sunny	mild	normal	true	yes
sunny	cool	normal	false	yes
overcast	hot	high	false	yes
overcast	hot	normal	false	yes
overcast	mild	high	true	yes
overcast	cool	normal	true	yes
rainy	mild	high	false	yes
rainy	mild	normal	false	yes
rainy	mild	high	true	no
rainy	cool	normal	false	yes
rainy	cool	normal	true	no

Tabelle 2: Entscheidungstabelle für Golf-Spiel

4.2 Entscheidungsbäume

Ein Entscheidungsbaum ist eine Repräsentationsform, bei der die Ergebnisse einer Bedingung verzweigt dargestellt werden. Diese Verzweigungen können wiederum andere Verzweigungen generieren. Entscheidungsbäume sind sehr gut zum Visualisieren, damit zum Verstehen und Begründen von Entscheidungen geeignet, da sie den Weg zur Entscheidung in graphisch aufbereiteter Form repräsentieren.

Beispiel 4.2 (Golfspiel)**.** In Abbildung 11 auf der nächsten Seite ist ein möglicher Entscheidungsbaum für das Golf-Beispiel angegeben.

WEKA liefert mit dem J48-Algorithmus den in Abb. 12 auf der nächsten Seite dargestellten Entscheidungsbaum.

4.3 Klassifikationsregeln

Die Einteilung in Klassen wird mittels Regeln dargestellt.

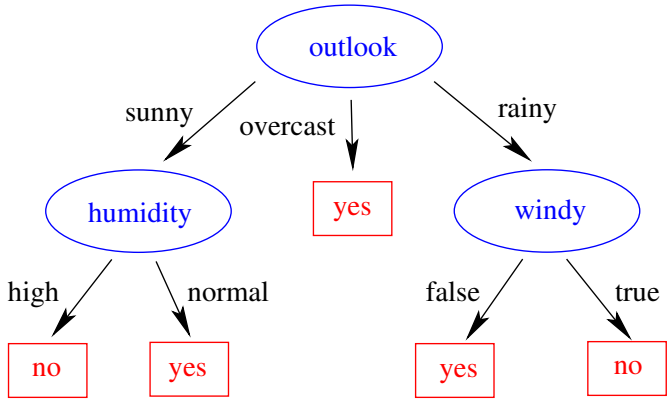


Abbildung 11: Entscheidungsbaum Golf-Spiel

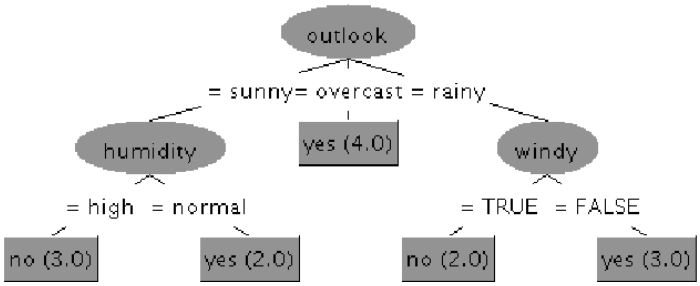


Abbildung 12: WEKA-Entscheidungsbaum Golf-Spiel

Beispiel 4.3 (Golfspiel).

$$\begin{aligned} IF\ outlook = sunny\ AND\ humidity = high &\quad THEN \quad play = no \\ IF\ outlook = rainy\ AND\ windy = true &\quad THEN \quad play = no \\ IF\ outlook = overcast &\quad THEN \quad play = yes \\ IF\ humidity = normal &\quad THEN \quad play = yes \\ IF\ \text{none of the above} &\quad THEN \quad play = yes \end{aligned}$$

4.4 Assoziationsregeln

Im Gegensatz zu den Klassifikationsregeln, die nur auf das Zielattribut ausgelegt sind, interessieren bei Assoziationsregeln auch Zusammenhänge zwischen beliebigen Attributen.
Warenkorbanalyse: In einem Supermarkt (vgl. Beispiel 3.4 auf Seite 21) werden an der Kasse die Warenkörbe aller Kunden erfasst. Mit Hilfe von Assoziationsregeln lassen sich nun Zusammenhänge zwischen den einzelnen Artikeln darstellen. Ein Ergebnis könnten folgende Erkenntnisse sein:

- Wenn Waschpulver gekauft wird, wird i. allg. auch Weichspüler gekauft:
  IF waschpulver THEN weichspüler
- Wenn Fisch gekauft wird, wird i. allg. kein Fleisch gekauft:
  IF fisch THEN ¬ fleisch
- Wenn Sekt gekauft wird, werden i. allg. auch Pralinen gekauft:
  IF sekt THEN pralinen

Beispiel 4.4 (Golfspiel). Man kann Assoziationsregeln auch folgendermaßen darstellen.

$$\begin{aligned} IF\ temperature = cool &\quad THEN \quad humidity = normal \\ IF\ humidity = normal\ AND\ windy = false &\quad THEN \quad play = yes \\ IF\ outlook = sunny\ AND\ play = no &\quad THEN \quad humidity = high \\ IF\ windy = false\ AND\ play = no &\quad THEN \quad outlook = sunny \\ &\quad AND \quad humidity = high \end{aligned}$$

4.4.1 Einfache Assoziationsregeln

Eine Assoziationsregel ist eine Implikation, gepaart mit Angaben über die Häufigkeit ihres Auftretens in einer Menge von Transaktionen. Gegeben seien eine Menge von Transaktionen T und eine Menge von sogenannten Items I. Die Items können als die Artikel verstanden werden, die in einem Supermarkt verkauft werden. Eine Transaktion ist in diesem Fall ein Einkauf bzw. ein Warenkorb.
Das Aussehen von Assoziationsregeln lässt sich folgendermaßen beschreiben: Eine Assoziationsregel besteht aus einer Prämisse A und einer Konsequenz B. A und B sind Konjunktionen (also durch Und verknüpft) von Items, die ihrerseits die Waren des Supermarkts darstellen. Die Regel hat dann die Form $A \to B$, mit z.B. $A = \{I_1, I_2, I_3\}$ und $B = \{I_7\}$. Die Schnittmenge von A und B muss leer sein.
Die Regel

$$\{\text{bier, chips}\} \to \{\text{tvzeitung}\}$$

ist also als abkürzende Schreibweise für die Regel

$$IF \text{ bier=yes } AND \ \ldots \ THEN \ \ldots$$

zu verstehen.
Außerdem werden zwei Interessantheitsmaße (vgl. Abschnitt 7.1 auf Seite 73) benötigt, welche die Qualität einer Assoziationsregel bestimmen.
Der Support eines Items oder Itemsets ist die Anzahl der Transaktionen, die das Item bzw. das Itemset als Teilmenge enthalten, im Verhältnis zur Gesamtzahl der Transaktionen der Menge T. Der Support einer Assoziationsregel ist gleich dem Support der Vereinigung von Prämisse und Konsequenz der Regel $(A \cup B)$.

$$\text{supp}(A \to B) = P(A \cup B)$$

Man zählt, wieviele Datensätze ALLE Attribute aus $A \cup B$ enthalten und teilt dies durch die Gesamtanzahl der Datensätze in der DB.
Die Konfidenz einer Assoziationsregel berechnet sich aus dem Verhältnis zwischen den Transaktionen, die sowohl Prämisse als auch Konsequenz enthalten, und den Transaktionen, die nur die Prämisse enthalten. Die Konfidenz misst, wie oft die Regel wirklich zutrifft in Relation zur Anzahl, wie oft sie hätte zutreffen müssen.

$$\text{conf}(A \to B) = \frac{\text{supp}(A \to B)}{\text{supp}(A)}$$

Beispiel 4.5 (Support und Konfidenz)**.** Wie hoch sind Support und Konfidenz der Regel

$$IF \ temperature = cool \ THEN \ humidity = normal$$

in Beispiel 4.1 auf Seite 25?

$$\text{supp}(temperature = cool \ \to \ humidity = normal) = P(A \cup B) = \frac{4}{14}$$

$$\text{supp}(temperature = cool) = P(A) = \frac{4}{14}$$

$$\text{conf}(A \to B) = \frac{\text{supp}(A \to B)}{\text{supp}(A)} = \frac{\frac{4}{14}}{\frac{4}{14}} = 1$$

Die Regel ist also absolut sicher, sie hat einen Support von $\frac{2}{7}$.

4.4.2 Schwellwerte

Um die wertvollen von den weniger wertvollen Assoziationsregeln zu trennen, müssen Schwellwerte für Konfidenz und Support eingeführt werden, die nicht unterschritten werden dürfen. Diese seien als conf_{min} und supp_{min} bezeichnet. Die Festlegung dieser Werte erfolgt zumeist durch den Benutzer des Data-Mining-Tools.
Meist gilt: Je größer Support und Konfidenz, umso wertvoller ist die Assoziationsregel. Hier kann es zu Ausnahmen kommen, so haben beispielsweise Regeln wie {Person lebt} → {Person atmet} trivialerweise eine hohe Konfidenz und sind dennoch uninteressant.

Beispiel 4.6 (Assoziationsregeln)**.** In Abb. 13 sind die Assoziationsregeln, die WEKA mit dem apriori-Algorithmus (vgl. Abschnitt 5.3.1 auf Seite 46) findet, dargestellt.

```
Minimum support: 0.15      Minimum metric <confidence>: 0.9
Best rules found:
 1. humidity=normal windy=FALSE 4 ==> play=yes 4    conf:1
 2. temp=cool 4 ==> humidity=normal 4    conf:1
 3. outlook=overcast 4 ==> play=yes 4    conf:1
 4. temp=cool play=yes 3 ==> humidity=normal 3    conf:1
 5. outlook=rainy windy=FALSE 3 ==> play=yes 3    conf:1
 6. outlook=rainy play=yes 3 ==> windy=FALSE 3    conf:1
 7. outlook=sunny humidity=high 3 ==> play=no 3    conf:1
 8. outlook=sunny play=no 3 ==> humidity=high 3    conf:1
 9. temp=cool windy=FALSE 2 ==> humidity=normal play=yes 2 conf:1
10. temp=cool humidity=normal windy=FALSE 2 ==> play=yes 2 conf:1
```

Abbildung 13: Assoziationsregeln für Golf-Spiel

4.4.3 Arten von Assoziationsregeln

Neben den ursprünglichen, klassischen Assoziationsregeln – auch *boolesche Assoziationsregeln* genannt – wurden eine Reihe von Variationen entwickelt, um bestimmte Nachteile zu beseitigen. Folgende Variationen werden in diesem Abschnitt vorgestellt:

- hierarchische Assoziationsregeln (Taxonomien)
- temporale Assoziationsregeln (Sequenzanalyse)
- quantitative Assoziationsregeln
- unscharfe Assoziationsregeln

Wieso betrachtet man weitere Formen von Assoziationsregeln? Zum einen geht es darum, die Aussagekraft von Regeln zu erhöhen. Außerdem hat man bei numerischen Attributen häufig das Problem, dass ein Wert knapp außerhalb eines Intervalls liegt und damit nicht als zu diesem Intervall zugehörig betrachtet wird, was mathematisch zwar korrekt ist, praktisch aber häufig unsinnig ist.

Hierarchische Assoziationsregeln Bei den *hierarchischen Assoziationsregeln* – auch Taxonomien genannt – werden mehrere Begriffe (Items) zu einem Oberbegriff zusammengefasst (z.B. einzelne Produkte zu Warengruppen/Kategorien). Es ist auch möglich, dass ein Begriff mehrere Oberbegriffe hat. Zum Beispiel kann ein Begriff zusätzlich zum Oberbegriff *Sonderangebot* gehören.
Der Vorteil der hierarchischen Assoziationsregeln ist, dass sich durch das Zusammenfassen von Begriffen der Support der Regel erhöht und dadurch mehr Regeln gefunden werden. Außerdem ist es nun möglich, Regeln auf einer abstrakteren Stufe zu finden.

Beispiel 4.7 (Hierarchische Assoziationsregeln)**.** Um hierarchische Assoziationsregeln aufzustellen, werden einzelne Items zu einer Gruppe zusammengefasst. Folgende Beispiele sind denkbar.

Items	Oberbegriff
{Messer, Gabel, Löffel}	Besteck
{Brot, Milch, Käse}	Lebensmittel
{Doppelpass, Flanke}	Angriff
...	...

Statt viele Regeln mit den Items der linken Tabellenseite zu erzeugen, genügen nun einige wenige.

Quantitative Assoziationsregeln Einfache und hierarchische Assoziationsregeln geben keine Auskunft über die Quantität eines Begriffs. Es wird nur angegeben, ob der Begriff in der Transaktion auftritt oder nicht. *Quantitative Assoziationsregeln* berücksichtigen auch die Quantität des Begriffs und ermöglichen dadurch eine genauere Unterscheidung. Sinnvoll ist dies bei numerischen Begriffen wie zum Beispiel Anzahl, Alter, Preis usw. Man geht dabei folgendermaßen vor:

1. Einteilung des Wertebereichs in Intervalle (Klassifizierung).
2. Für jedes Intervall wird ein neuer Begriff geschaffen.
3. Die originalen Begriffe werden durch die neuen ersetzt.

Beispiel 4.8 (Quantitative Assoziationsregeln)**.** Gegeben sei folgende Datenbasis:

Alter	Anzahl Kinder	Einkommen
23	0	2000
28	1	2500
34	0	4500
45	2	3500
65	5	4000

Der Begriff `Alter` wird in 3 Intervalle unterteilt: `unter 30, 30 bis 50, über 50`. Der Begriff `Anzahl Kinder` wird in 2 Intervalle eingeteilt: `kein oder 1 Kind`, `mehr als 1 Kind`. Das Einkommen wird ebenfalls in 2 Intervalle eingeteilt: `bis 3000`, `mehr als 3000` (vgl. Binärkodierung in Abschnitt 6.2.4 auf Seite 66). Anhand dieser Daten wird eine neue Datenbasis erstellt.

Alter1	Alter2	Alter3	Kinder1	Kinder2	Einkommen1	Einkommen2
1	0	0	1	0	1	0
1	0	0	1	0	1	0
0	1	0	1	0	0	1
0	1	0	0	1	0	1
0	0	1	0	1	0	1

Eine quantitative Regel könnte dann so aussehen:

$$(Alter \in [0, 29], Einkommen \in [0, 2999]) \rightarrow (Kinder = 0/1)$$

Unscharfe Assoziationsregeln Die *unscharfen Assoziationsregeln* – auch fuzzy association rules genannt – sind eine Weiterentwicklung der quantitativen Assoziationsregeln. Die Intervallgrenzen sind nun nicht starr, sondern fließend. Zum Beispiel wird der Begriff Alter in `jung`, `mittel` und `alt` eingeteilt. Bis 25 gehört man zu jung, ab 35 zu mittel und dazwischen gehört man zu beiden Gruppen.
Durch die fließenden Intervallgrenzen ist der Support einer Regel größer als bei quantitativen Assoziationsregeln, da nun auch Begriffe die bei den quantitaven Assoziationsregeln knapp außerhalb der Intervallgrenzen lagen, berücksichtigt werden.
Unscharfe Regeln eignen sich besonders für Datenanalysen, bei denen mit Ausreißern oder Messfehlern gearbeitet wird, wie zum Beispiel in der Physik.

Beispiel 4.9 (Unscharfe Assoziationsregeln)**.** Ein Call-Center plant, Daten der eingehenden Anrufe zu speichern. Zu diesen Daten zählt unter anderem auch der Zeitpunkt, an dem der Anruf angenommen wurde. Angenommen, die Leitung des Centers möchte die Anrufe nach Tageszeiten

sortieren. Das Vorgehen nach dem quantitativen Schema würde so ablaufen, dass die 24 Stunden des Tages in Intervalle aufgeteilt würden, beispielsweise in `Nacht`, `Morgen`, `Nachmittag` und `Abend`. Das Intervall `Nacht` endet um 6 Uhr, ihm folgt das Intervall `Morgen`. Der `Morgen` endet um 12 Uhr und geht in den `Nachmittag` über usw.
Das Charakteristikum dieser Vorgehensweise ist, dass es zu Überschneidungen kommen kann. So könnte es eine Gruppe von Anrufern geben, die morgens zwischen 6 und 12 anrufen, um z.B. Brötchen zu bestellen. Eine Regel der Form Zeit = Morgen → Bestellung = Brötchen wäre die Folge. Der Nachteil dieser Vorgehensweise liegt darin, dass es Kunden geben kann, die kurz vor 6 oder kurz nach 12 anrufen, um Brötchen zu bestellen. Da der Zeitpunkt dieser Anrufe nicht mehr in dem vorgegebenen Intervall liegt, gehören diese Anrufe auch nicht mehr zur Kundengruppe `Morgen`, auch wenn sie sonst alle Eigenschaften dieser Gruppe erfüllen. Die Folge ist, dass der Support obiger Regel sinkt.

Mit unscharfen Regeln kann diesem Verhalten entgegengewirkt werden. Anstelle von festen Intervallen wird mit Zugehörigkeitsgraden gearbeitet. Ein Anruf um 11 Uhr kann sowohl der Gruppe `Morgen` als auch der Gruppe `Nachmittag` zugeordnet werden. Die Zuordnung geschieht mit Methoden aus der Fuzzy-Logik. Der wesentliche Unterschied zwischen Fuzzy und quantitativen Assoziationsregeln liegt in der Regelgenerierung, im weiteren Verhalten sind sich beide Formen recht ähnlich.

Temporale Assoziationsregeln Bei den *temporalen Assoziationsregeln* bzw. bei der *Sequenzanalyse* wird neben der Information, dass ein Begriff in einer Transaktion auftrat, auch der Zeitpunkt bzw. die Reihenfolge erfasst. Damit ist die Sequenzanalyse im Gegensatz zur normalen Assoziationsanalyse eine Zeitraumanalyse.
Ziel der Sequenzanalyse ist es, Sequenzen zu finden, die einen Mindestsupport aufweisen. Wenn beispielsweise jeden Freitag 6 Kästen Bier gekauft werden und am nächsten Tag die Verkaufszahlen für Kopfschmerztabletten in die Höhe schnellen, besteht ein temporaler Zusammenhang, der berücksichtigt werden sollte. In temporalen Datenbanken sind Daten in einem zeitlichen Kontext gespeichert. Die Assoziationsregeln können als *Schnappschüsse* der sich verändernden Zusammenhänge zwischen den Daten aufgefasst werden. Dadurch ist es möglich, Veränderungen und Fluktuationen dieser Zusammenhänge zu betrachten und zu erforschen.
Ein typisches Einsatzgebiet für die Sequenzanalysen sind die Logfile-Analysen, bei denen das Verhalten von Besuchern einer Webseite untersucht wird.

4.5 Instanzenbasierte Darstellung

Bei der instanzenbasierten Darstellung werden – ähnlich wie beim Auswendiglernen – einfach alle Individuen gespeichert, z.B. in einer relationalen Datenbank.

4.6 Cluster

Wird eine Grundgesamtheit in Teilmengen zerlegt, deren Individuen zueinander ähnlicher als zu den Individuen der anderen Teilmengen sind, bezeichnet man diese Teilmengen als **Cluster**.
Darstellen lässt sich ein Cluster

- als **Menge** der ihm zugeordneten Individuen, aber auch
- als **Vektor der Mittelwerte** der Attribute seiner Individuen (Centroid) oder
- durch **einen typischen Vertreter** der Cluster-Elemente (Medoid),
- über **Wahrscheinlichkeiten** (jedes Individuum kann verschiedenen Clustern mit einer bestimmten Wahrscheinlichkeit angehören)

In Abb. 14 auf der nächsten Seite sind für einen Beispiel-Cluster sowohl der Centroid als auch der Medoid dargestellt. Der Centroid muss nicht in der Datenmenge vorkommen, aber der Medoid. Als Medoid kann man beispielsweise den Punkt bzw. Datensatz wählen, der am nächsten zum Centroid liegt.
Mit dem k-Means-Algorithmus (vgl. Abschnitt 5.6.1 auf Seite 51) erreicht WEKA für das Wetter-Beispiel die in Abb. 15 auf der nächsten Seite dargestellten Cluster.

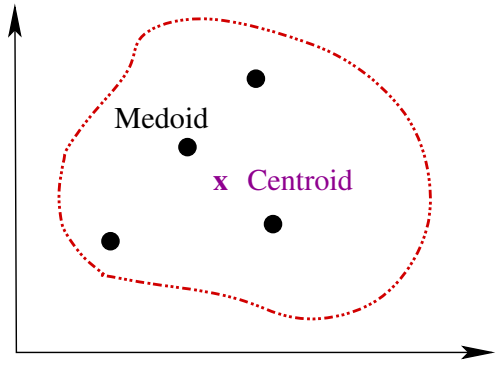


Abbildung 14: Centroid und Medoid

```
k-Means
======
Number of iterations: 4
Within cluster sum of squared errors: 26.0
Cluster centroids:
Cluster 0   Mean/Mode:  sunny mild high FALSE yes
            Std Devs:   N/A   N/A   N/A   N/A   N/A
Cluster 1   Mean/Mode:  overcast cool normal TRUE yes
            Std Devs:   N/A   N/A   N/A   N/A   N/A
Clustered Instances
0      10 ( 71%)           1       4 ( 29%)
```

Abbildung 15: Cluster für das Wetter-Beispiel

5 Methoden und Verfahren

A carelessly planned project takes three times longer to complete than expected; a carefully planned project takes only twice as long.
Golub's Second Law of Computerdom

Wir gehen davon aus, dass wir unsere Daten bereits gesammelt und vorverarbeitet haben. D.h. wir haben (i. allg.) eine einzige Datentabelle mit unseren kompletten Daten vorliegen. Dies ist in der Praxis natürlich nicht der Fall; vielmehr müssen wir uns unsere Daten i. allg. erst zusammensuchen und diese vorverarbeiten (Kapitel 6). Wir konzentrieren uns jetzt zunächst auf die Verfahren.
Es gibt viele Data-Mining-Methoden. In diesem Kapitel werden einige dieser Methoden beschrieben sowie Verfahren zur praktischen Umsetzung mit dazu verwendbaren Werkzeugen vorgestellt. Weder die Menge der Methoden noch die der Verfahren und Werkzeuge können vollständig sein, da das Gebiet des *Data Mining* extrem dynamisch ist und sich immer noch stürmisch weiterentwickelt.
In Tabelle 3 ist eine Übersicht über die behandelten Verfahren(sklassen) sowie die Problemklassen, für die sie einsetzbar sind, dargestellt.

	Klassifikation	Assoziation	Clustering	Numer. Vorhersage	Text Mining	Web Mining
Instanzenbasiertes Lernen	x					
k Nearest Neighbour	x		(x)	(x)		
Entscheidungsbaumlernen	x			(x)		
a priori		x			x	x
Lineare Regression				x		
Überwachte Neuronale Netze	x	x		x		
Selbstorganisierende Neuronale Netze	(x)		x			
k-means			x			
Naive Bayes	x				x	

Tabelle 3: Data-Mining-Verfahren und Anwendungsklassen

5.1 Instanzenbasiertes Lernen

Beim instanzenbasierten Lernen handelt es sich um das einfachste Verfahren zur Klassifikation. Alle Individuen der Trainingsmenge werden gespeichert. Zur Klassifikation eines unbekannten Individuums wird das ähnlichste Individuum der bekannten Menge gesucht und dessen Klasse vorhergesagt.

5.1.1 k Nearest Neighbour

Das k-nearest-neighbour-Verfahren (kNN) ist ein instanzenbasiertes Verfahren. Es setzt voraus, dass die zu erlernenden Objekte durch eine Menge von reellwertigen Attributen beschrieben sind. Der Lernschritt des Verfahrens ist trivial: Es werden lediglich Beispielobjekte gespeichert. Unbekannte Objekte werden klassifiziert, indem die *Ähnlichkeit* (vgl. Abschnitt 2.5) der Attributwerte des unbekannten Objekts zu denen der bereits gespeicherten Objekte berechnet wird. Die k ähnlichsten unter den gegebenen Objekten werden dann zur Vorhersage der Klasse des neuen Objekts herangezogen. Allerdings kann es dabei zu Problemen kommen. In Abb. 16 auf der nächsten Seite ist zweidimensional dargestellt, dass sich für den zu klassifizierenden Vektor je nach Wahl von k unterschiedliche

Klassifikationen ergeben können. Die Objekte, die mit $\oplus$ gekennzeichnet sind, gehören zur einen Klasse, die mit einem $\bullet$ gekennzeichneten Objekte zur anderen Klasse. Für das Objekt $\circ$ ist keine Klassenzugehörigkeit bekannt. Anhand der Klassenzugehörigkeit der anderen Objekte soll das neue Objekt einer dieser beiden Klassen zugeordnet werden.

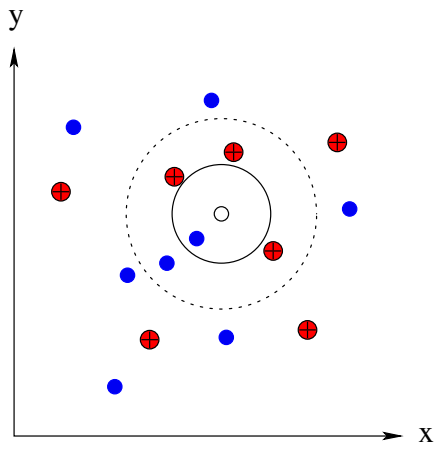


Abbildung 16: Beispiel k Nearest Neighbour

Bei $k = 1$ (veranschaulicht mittels der durchgezogenen Kreisbahn) gilt klasse$(\circ) = \bullet$, bei $k = 5$ (gestrichelter Kreis) hingegen ergibt sich klasse$(\circ) = \oplus$.
Für das Resultat kann also die (un)geschickte Wahl von k maßgeblich sein. Man rechnet deshalb in der Praxis häufig mehrere Varianten mit unterschiedlichen k und vergleicht die vorhergesagten Klassen.
Die Komplexität des Verfahrens wird weniger durch den Lernschritt, als vielmehr durch den Klassifikationsschritt bestimmt. Weil aber keine komplette und kompakte Beschreibung der zu erlernenden Funktion generiert werden muss, kann sich die unter Umständen teure Berechnung der Ähnlichkeit auf einen als relevant ermittelten Suchbereich beschränken.
Diese zentralen Merkmale machen das kNN-Lernen zu einem engen Verwandten z.B. des fallbasierten Schließens und der lokal gewichteten Regression.

5.1.2 Der kNN-Algorithmus

Der Lernschritt beim kNN-Lernen ist sehr einfach. Er lautet:
Sei $f(x)$ die zu erlernende Funktion. Für jedes Trainingsbeispiel $(x, f(x))$ speichere das Beispiel in einer Liste *Trainingsbeispiele*.
Für verschiedene Arten von Funktionen gibt es nun unterschiedliche Klassifikationsschritte.

Diskrete Funktion Sei $V := \{v_1, v_2, ..., v_m\}$ eine endliche Menge (dies sind die Werte, die das Zielattribut annehmen kann) und $f : \Re^n \to V$ die zu erlernende Funktion. Sei weiter der Vektor y das zu klassifizierende Beispiel. Dann hat der kNN-Algorithmus die folgende Form:
Für alle x_i in der Menge *Trainingsbeispiele* berechne die Ähnlichkeit zu y. Wähle diejenigen k Beispiele $x_1, x_2, \ldots, x_k$ aus, die zu y am ähnlichsten sind. Dann gib zurück:

$$\text{klasse}(y) := \max_{v \in V} \sum_{p=1}^{k} \delta(v, f(x_p)) \qquad \text{mit} \quad \delta(a,b) := \begin{cases} 1, \text{ falls a=b} \\ 0, \text{ sonst} \end{cases}$$

Die Funktion `klasse` berechnet also denjenigen Wert, der unter den Klassifikationswerten der k Traingsbeispiele am häufigsten vorkommt. D.h. y wird der Klasse zugeordnet, die unter den k ähnlichsten Beispielen *am meisten* vertreten ist. Diese wird dann für den neuen Datensatz vorhergesagt. Beachten Sie, dass das k nichts über den Abstand der Trainingsbeispiele vom neuen Datensatz aussagt. Es geht allein um die k ähnlichsten Objekte, egal wie weit sie vom neuen Datensatz entfernt sind.

Beispiel 5.1 (kNN)**.** Folgende Daten sind gegeben:

Nr	Alter	verheiratet	Eigenheim	Akademiker	Einkommen
1	alt	ja	ja	ja	hoch
2	alt	ja	nein	nein	gering
3	mittel	nein	nein	nein	gering
4	mittel	ja	ja	ja	hoch
5	jung	nein	nein	nein	gering
6	jung	ja	nein	nein	mittel
7	jung	ja	ja	ja	mittel
8	alt	nein	ja	nein	hoch

Sei das vorherzusagende Attribut die *Gehaltsgruppe* (hoch, mittel gering). Wir wollen das Einkommen für einen *jungen, verheirateten Akademiker ohne Eigenheim* vorhersagen. Zunächst werden die zum gegebenen Datensatz k ähnlichsten Datensätze bestimmt. Sei k=2. Es werden die 2 ähnlichsten Datensätze (ohne das Einkommen-Attribut) bestimmt. Dazu benutzen wir die Hamming-Distanz.

Nr	Alter	verheiratet	Eigenheim	Akademiker	Abstand
neu	jung	ja	nein	ja	
1	alt	ja	ja	ja	2
2	alt	ja	nein	nein	2
3	mittel	nein	nein	nein	3
4	mittel	ja	ja	ja	2
5	jung	nein	nein	nein	2
6	jung	ja	nein	nein	**1**
7	jung	ja	ja	ja	**1**
8	alt	nein	ja	nein	4

In diesen 2 Datensätzen (6 und 7) zählt man nun, welche Gehaltsgruppe am häufigsten vertreten ist: Gehaltsgruppe *mittel*.

Reellwertige Funktion Für reellwertige Funktionen $f : \Re^n \to \Re$ unterscheidet sich der kNN-Algorithmus vom diskreten Fall nur darin, dass nun der Mittelwert der ausgewählten k Beispiele zurückgegeben wird:
Für alle x_i in der Menge *Trainingsbeispiele* berechne die Ähnlichkeit zu y. Wähle diejenigen k Beispiele $x_1, x_2, \ldots, x_k$ aus, die zu y am ähnlichsten sind. Dann gib zurück:

$$\text{f'}(y) := \frac{\sum_{p=1}^{k} f(x_p)}{k}$$

Den in Abb. 16 auf der vorherigen Seite dargestellten Unwägbarkeiten – den möglichen Schwankungen bei unterschiedlichen k – wirkt die im nächsten Abschnitt beschriebene Verfeinerung des kNN-Algorithmus entgegen.

5.1.3 Ein verfeinerter Algorithmus

Eine offensichtliche Schwäche des elementaren kNN-Algorithmus ist der Umstand, dass alle der k ausgewählten Beispiele zu gleichen Teilen zum Ergebnis des Klassifikationsschritts beitragen. Sollte jedoch die grundsätzliche Annahme, dass ein geringer Abstand eine hohe Ähnlichkeit impliziert, ihre Berechtigung haben, dann erschiene es sinnvoll, jedem einzelnen Ergebnisvektor gemäß seinem Abstand zum Anfragevektor ein Gewicht zuzuordnen. Eine derartige Vorgehensweise ermöglicht auch, alle Trainingsbeispiele zur Ermittlung des Ergebnisses heranzuziehen, weil solche mit hohem Abstand kaum noch zum Resultat beitragen; im reellwertigen Fall spricht man dann von der sogenannten *Shepard's method*. Allerdings impliziert diese Methode einen höheren Rechenaufwand im Klassifikationsschritt.

Diskrete Funktionen Als Gewicht wird das Inverse des Quadrats der Distanz zwischen Trainings- und Anfragevektor gewählt. Dann lautet im diskreten Fall der durch Distanzgewichtung verfeinerte Klassifikationsschritt wie folgt:
Sei wieder $V := \{v_1, v_2, ..., v_m\}$ die Menge aller Werte, die das Zielattribut annehmen kann. Für alle x_i in der Menge *Trainingsbeispiele* berechne die Ähnlichkeit zu y. Wähle diejenigen k Beispiele $x_1, x_2, \ldots, x_k$ aus, die zu y am ähnlichsten sind.

$$\text{klasse}(y) := \begin{cases} f(x_i) & \text{falls} \quad y = x_i \text{ für ein } i \\ \max\limits_{v \in V} \sum\limits_{p=1}^{k} w_p \times \delta(v, f(x_p)) & \text{sonst} \end{cases}$$

$$\text{mit } \delta(a,b) := \begin{cases} 1, & \text{falls} \quad \text{a=b} \\ 0, & \text{sonst} \end{cases} \qquad \text{und} \qquad w_p := \frac{1}{\text{dist}(y,x_p)^2}.$$

Reellwertige Funktionen Im reellwertigen Fall lautet der verfeinerte Algorithmus:
Für alle x_i in der Menge *Trainingsbeispiele* berechne die Ähnlichkeit zu y. Wähle diejenigen k Beispiele $x_1, x_2, \ldots, x_k$ aus, die zu y am ähnlichsten sind.

$$\text{f'}(y) := \begin{cases} f(x_i) & \text{falls} \quad y = x_i \text{ für ein } i \\ \dfrac{\sum\limits_{p=1}^{k} w_p \times f(x_p)}{\sum\limits_{p=1}^{k} w_p} & \text{sonst} \end{cases}$$

mit $w_p := \frac{1}{\text{dist}(y,x_p)^2}$.

5.1.4 Anmerkungen

- Für $k \gg 1$ arbeitet der kNN-Algorithmus i. allg. auch bei verrauschten Trainingsdaten sehr gut.
- Im Gegensatz beispielsweise zum Entscheidungsbaum werden alle Attribute in die Berechnung einbezogen. Dies bedeutet, dass der Algorithmus stark an Zuverlässigkeit verliert, wenn einige der Attribute für die Bestimmung der Ähnlichkeit irrelevant sind. Dieses Problem lässt sich dadurch abmildern, dass man den Attributen Gewichte zuweist.
- Die Auswahl der Trainingsdaten verdient einige Beachtung. So ist z.B. von Bedeutung, dass die Trainingsvektoren den Lösungsraum möglichst gleichmäßig aufspannen.

Aufgabe 5.1 (kNN)**.** Klassifizieren Sie folgende Datensätze auf Basis der Tabelle 4 auf der nächsten Seite (vgl. Aufgabe 5.3 auf Seite 45) mittels kNN. Testen Sie das Verfahren mit unterschiedlichen k. Welche Distanzen schlagen Sie vor?

Alt.	Fr/Sa	Hung.	Gäste	Reserv.	Typ	Zeit	Warten
ja	nein	ja	einige	nein	Franz.	30-60	
ja	ja	ja	voll	ja	Chin.	10-30	
nein	nein	nein	keine	nein	Burger	0-10	

Die Tabelle enthält 12 Datensätze mit diesen Attributen:

- Alternative: Gibt es in der Nähe ein geeignetes anderes Restaurant? (ja/nein)
- Fr/Sa: Ist Freitag oder Samstag? (ja/nein)
- Hungrig: Bin ich hungrig? (ja/nein)
- Gäste: Wieviele Leute sind im Restaurant? (keine/einige/voll)
- Reservierung: Habe ich reserviert? (ja/nein)
- Typ: Um welche Art von Restaurant handelt es sich? (Franz./Chin./Ital./Burger)
- Wartezeit: Welche Wartezeit wird vom Restaurant geschätzt? (0-10/10-30/30-60/>60)
- Warten (Zielattribut): Warte ich, wenn alle Tische besetzt sind? (ja/nein)

Alt.	Fr/Sa	Hung.	Gäste	Reserv.	Typ	Zeit	Warten
ja	nein	ja	einige	ja	Franz.	0-10	ja
ja	nein	ja	voll	nein	Chin.	30-60	nein
nein	nein	nein	einige	nein	Burger	0-10	ja
ja	ja	ja	voll	nein	Chin.	10-30	ja
ja	ja	nein	voll	ja	Franz.	>60	nein
nein	nein	ja	einige	ja	Ital.	0-10	ja
nein	nein	nein	keine	nein	Burger	0-10	nein
nein	nein	ja	einige	ja	Chin.	0-10	ja
nein	ja	nein	voll	nein	Burger	>60	nein
ja	ja	ja	voll	ja	Ital.	10-30	nein
nein	nein	nein	keine	nein	Chin.	0-10	nein
ja	ja	ja	voll	nein	Burger	30-60	ja

Tabelle 4: Restaurant-Beispiel

5.2 Entscheidungsbaumlernen

5.2.1 Erzeugen eines Entscheidungsbaums

In Abb. 11 auf Seite 26 haben wir bereits einen Entscheidungsbaum gesehen. Wir schauen uns nun an, wie man einen Entscheidungsbaum generieren kann. Ausgehend von der Attributmenge A und der Beispielmenge E entwickelt man einen Entscheidungsbaum, indem man zunächst ein Attribut $a \in A$ als Wurzel auswählt. Sei ω_a die Menge aller Werte, die das Attribut a annehmen kann. Für jede Ausprägung $\omega \in \omega_a$ dieses Attributs wird die Menge $E^\omega \subseteq E$ gebildet, für die gilt $\forall e \in E^\omega : \omega_a(e) = \omega$. Des weiteren wird eine mit dieser Ausprägung markierte Kante an die Wurzel gelegt. Ist $E^\omega = \emptyset$, beendet man die Kante mit NIL. Entfallen *alle* Beispiele aus E^ω auf die *gleiche* Klasse, wird die Kante mit einem mit der Klasse markierten Blatt abgeschlossen. In jedem anderen Fall wird ein weiterer Entscheidungsbaum generiert, dessen Wurzel als Knoten an das Ende der Kante gehängt wird. Dieser neue Teilbaum wird nun analog – ausgehend von der Attributmenge $A' = A \setminus \{a\}$ und der Beispielmenge E^ω – erzeugt.
Wir betrachten zur Erläuterung die Daten des Golfspiels (Tabelle 5).

Tag	outlook	temperature	humidity	windy	play
1	sunny	hot	high	false	no
2	sunny	hot	high	true	no
3	overcast	hot	high	false	yes
4	rainy	mild	high	false	yes
5	rainy	cool	normal	false	yes
6	rainy	cool	normal	true	no
7	overcast	cool	normal	true	yes
8	sunny	mild	high	false	no
9	sunny	cool	normal	false	yes
10	rainy	mild	normal	false	yes
11	sunny	mild	normal	true	yes
12	overcast	mild	high	true	yes
13	overcast	hot	normal	false	yes
14	rainy	mild	high	true	no

Tabelle 5: Daten Golfspiel

Am bereits vorgestellten Entscheidungsbaum (Abb. 17 auf der nächsten Seite) soll das Vorgehen veranschaulicht werden. Als obersten Knoten hat man *outlook* gewählt. Dieses Attribut hat 3 Ausprägungen. Folglich werden 3 Kanten erzeugt (*sunny, overcast, rainy*). Nun werden für jede Ausprägung die Datensätze aus der ursprünglichen Datenbank selektiert, die als Attributwert für *outlook*

genau diesen Wert haben. Für *overcast* können wir nun sofort *yes* eintragen, da in allen Datensätzen, in denen der Attributwert *overcast* ist, gespielt wird. Für die beiden anderen Fälle muss das Verfahren mit den jeweiligen Teil-Datenbanken fortgesetzt werden.

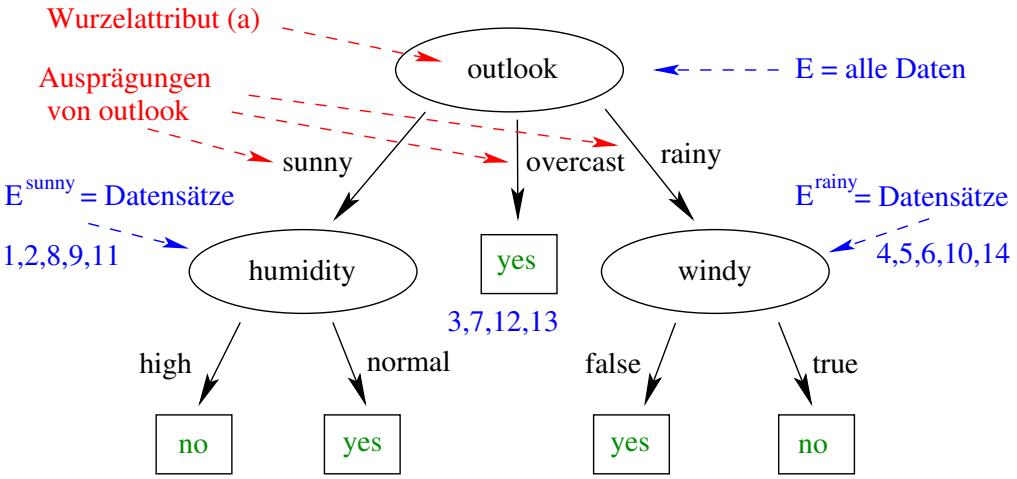


Abbildung 17: Entscheidungsbaum Golf-Spiel

5.2.2 Auswahl eines Attributs

In den Pfaden eines Entscheidungsbaums können die Attribute in unterschiedlicher Reihenfolge auftreten. Daher gibt es für die Lösung eines Problems nicht *den* einzig richtigen Entscheidungsbaum, sondern eine Menge möglicher Entscheidungsbäume.
Welchen Baum sollten wir anstreben? Natürlich möchten wir einen solchen Baum haben, der möglichst kompakt ist, der uns also nach wenigen Fragen eine Klasse liefert.
Unter Betrachtung des Aufwands ist es jedoch unmöglich, alle diese Bäume zu generieren und zu vergleichen. Um dieses Problem zu lösen gibt es drei grundsätzliche Möglichkeiten.

manuell Das auszuwählende Attribut wird vom Benutzer manuell vorgegeben. Dies kann nur bei kleinen Attributmengen eine sinnvolle Lösung sein.

zufällig Eine naheliegende Variante zur Attributwahl ist der Zufallsgenerator. Dabei können aber auch Bäume mit sehr langen Pfaden entstehen, die unübersichtlich und somit schlecht nachvollziehbar sind.

berechnet Bei dieser Form der Attributauswahl wird automatisch nach dem optimalen (wir nehmen an, wir hätten schon ein Optimalitätskriterium) Attribut für die Wurzel eines Baums gesucht.

Beispiel 5.2 (Automatische Attributwahl)**.** Man sucht ein geeignetes Attribut natürlich unter der Maßgabe, dass dieses *lokal* die Klassifikationsleistung am meisten verbessern würde. Dazu wird probeweise jedes Attribut a als Wurzel angenommen. Anhand der Ausprägungen ω_a des Attributs teilt man die Beispielmenge. Für jede dieser Teilmengen wird nun die in der Teilmenge häufigste Klasse vorhergesagt. Aufgrund dieses naiven Verfahrens ermittelt man die Fehlerrate $\text{error}(\omega_a)$. Die Fehlerrate error_a des Attributs wird als Summe der mit der relativen Häufigkeit gewichteten Fehlerraten seiner Ausprägungen berechnet. Für die Wurzel des Entscheidungsbaums wählt man nun dasjenige Attribut mit der geringsten Fehlerrate.

$$\text{error}(\omega_a) = \frac{falsch}{alle}$$

$$\text{error}_a \;=\; \sum_i \left(\frac{|A_{\omega_{ai}}|}{|A|} * \text{error}(\omega_{ai})\right) \;\rightarrow\; min.$$

5.2.3 Metrische Attribute

Bei metrischen Attributen, insbesondere bei großen Wertebereichen, ist es häufig nicht nur unangebracht, sondern gar unmöglich, für jede Ausprägung eine einzelne Kante anzulegen (z.B. reelle Zahlen). Zur Lösung dieses Problems gibt es zwei Möglichkeiten:

Gruppierung Es werden die Ausprägungen in Intervallen zusammengefasst. Für jedes der Intervalle wird nun eine eigene Kante angelegt (vgl. Abschnitt *Quantitative Assoziationsregeln* auf Seite 29).

Schwellwerte Es werden genau zwei Kanten erzeugt, die sich dadurch unterscheiden, dass die Attribute einen bestimmten Zahlenwert unter- bzw. überschreiten. Es ist bei diesem Vorgehen möglich, innerhalb eines Pfads des Baums ein Attribut mehrfach mit verschiedenen Schwellwerten zu untersuchen.

5.2.4 Der ID3-Algorithmus zur Erzeugung eines Entscheidungsbaums

Seien die in Tabelle 6 dargestellten Daten zur Bewertung des Kreditrisikos gegeben.

Nr.	Zielattr. Risiko	frühere Kreditwürdigkeit	Verschuldung	Sicherheiten	Einkommen
1	hoch	schlecht	hoch	keine	0 bis 15
2	hoch	unbekannt	hoch	keine	15 bis 35
3	mittel	unbekannt	niedrig	keine	15 bis 35
4	hoch	unbekannt	niedrig	keine	0 bis 15
5	niedrig	unbekannt	niedrig	keine	über 35
6	niedrig	unbekannt	niedrig	angemessen	über 35
7	hoch	schlecht	niedrig	keine	0 bis 15
8	mittel	schlecht	niedrig	angemessen	über 35
9	niedrig	gut	niedrig	keine	über 35
10	niedrig	gut	hoch	angemessen	über 35
11	hoch	gut	hoch	keine	0 bis 15
12	mittel	gut	hoch	keine	15 bis 35
13	niedrig	gut	hoch	keine	über 35
14	hoch	schlecht	hoch	keine	15 bis 35

Tabelle 6: Kreditrisiko

Wie generiert man daraus einen Entscheidungsbaum, der uns bei der Entscheidung hilft, ob aus der Sicht der Bank das Risiko einer Kreditvergabe hoch ist oder nicht? Dieser Entscheidungsbaum soll uns insbesondere auch bei Fällen, die in der obigen Tabelle nicht vorkommen, helfen. In Abb. 18 auf der nächsten Seite ist ein Entscheidungsbaum dargestellt, der alle Beispiele der Tabelle 6 erfasst.
Fast immer hängt die Größe des Entscheidungsbaums davon ab, in welcher Reihenfolge die Eigenschaften (Attribute) für die Fallunterscheidung benutzt werden. In Abb. 19 auf der nächsten Seite ist ein wesentlich kleinerer Entscheidungsbaum dargestellt.
Welcher Baum ist für die Klassifikation der unbekannten Datensätze optimal? Der ID3-Algorithmus unterstellt, dass dies der **einfachste Baum** ist. D.h. wir möchten einen möglichst kompakten Baum haben. Der Algorithmus ist in Abb. 20 auf der nächsten Seite dargestellt.
Wie wählt der ID3-Algorithmus ein geeignetes Attribut aus? Das hier vorgestellte Verfahren wurde 1979 von Quinlan entwickelt und benutzt den Entropiebegriff von Shannon. Unter Entropie wird die Unreinheit, die Unordnung verstanden. Je höher die Entropie einer Datentabelle ist, desto größer ist deren Unordnung, desto mehr Fragen muss man also noch stellen. Grundlage ist die Informationstheorie. Gewählt wird das Attribut, welches den größten Informationsgewinn liefert.
Der **Informationsgehalt** eines Attributs B wird gemessen als:

$$I(B) = \sum_{i=1}^{k} -p(b_i) \times \log_2(p(b_i))$$

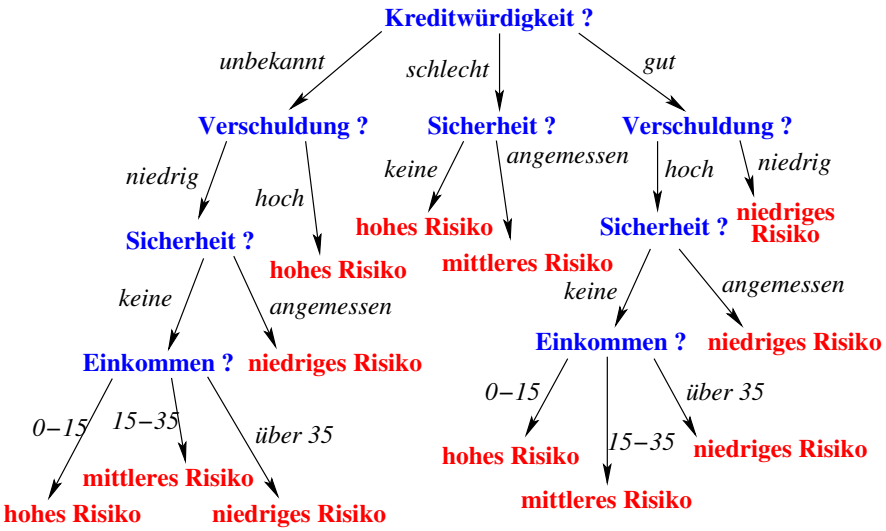


Abbildung 18: Entscheidungsbaum

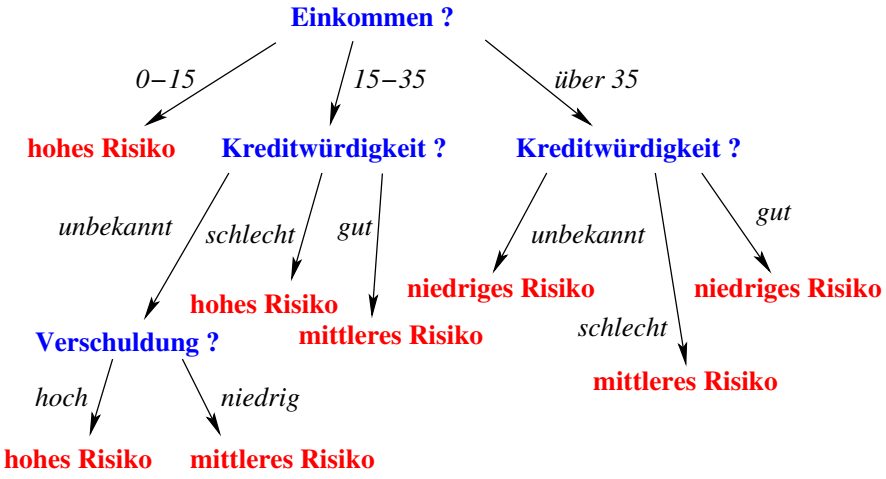


Abbildung 19: Entscheidungsbaum 2

```
FUNCTION induce_tree(Beispielmenge Ex, Attribute Attr)
 IF alle Eintraege aus Ex gehoeren zur gleichen Klasse
   THEN RETURN Blattknoten mit Beschriftung dieser Klasse
   ELSE
     Waehle ein Attribut A aus Attr;
     Setze A als Wurzel fuer den aktuellen Baum;
     Loesche A aus Attr;
     FOREACH Wert AV von A
       Erstelle Kante im Baum mit Kantenbeschriftung AV;
       Seien Ex_AV alle Elemente von Beispielmenge Ex,
               die als Wert fuer A gerade AV haben;
       Ergebnis der Kante AV := induce_tree(Ex_AV,Attr);
     END FOREACH;
 END IF;
END.
```

Abbildung 20: Algorithmus Entscheidungsbaum

Dabei stellen die b_i die möglichen Werte des Attributs B dar (k Werte). p ist die Wahrscheinlichkeit (besser: relative Häufigkeit) für das Eintreffen von b_i.
Wie groß ist der Informationsgehalt der Tabelle 6 auf Seite 39? D.h. wie groß ist der Informationsgehalt der Tabelle bezüglich des Zielattributs `Risiko`? `Risiko` hat 3 Ausprägungen mit folgenden relativen Häufigkeiten:

- $p(\text{Risiko hoch}) = \frac{6}{14}$
- $p(\text{Risiko mittel}) = \frac{3}{14}$
- $p(\text{Risiko niedrig}) = \frac{5}{14}$

Folglich ist

$$I(\text{Tabelle}) = I(\text{Risiko}) = -\frac{6}{14} \times \log_2(\frac{6}{14}) - \frac{3}{14} \times \log_2(\frac{3}{14}) - \frac{5}{14} \times \log_2(\frac{5}{14}) = 1,531$$

Tipp: Wenn Sie einen Taschenrechner haben, auf dem kein $\log_2$ verfügbar ist, dann kann man sich wie folgt behelfen:

$$\log_2(x) = \frac{\log_{10}(x)}{\log_{10}(2)} = \frac{\ln(x)}{\ln(2)}$$

Nun wählt man das Attribut aus, das den **maximalen Informationsgewinn** erzielt. D.h. man berechnet die Differenz aus $I(\text{Tabelle})$ und der Information, die sich unter Beachtung der nächsten Ebene ergibt (Abb. 21).

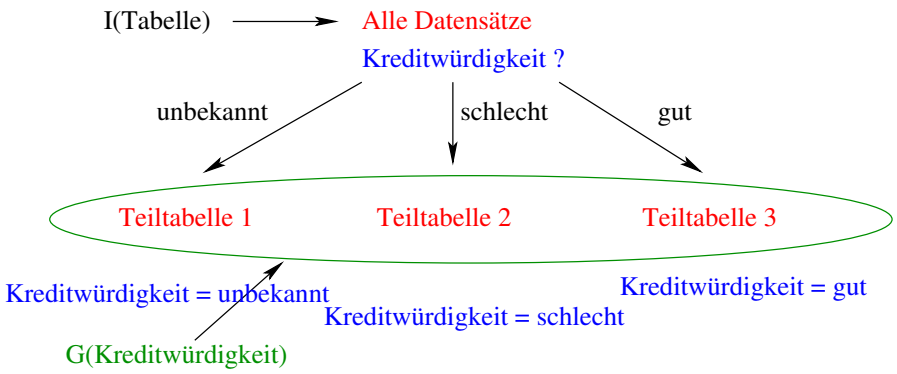


Abbildung 21: Informationsgewinn

Sei eine Beispielmenge E (die komplette DB) gegeben. Wählt man nun z.B. das Attribut B mit n Werten aus, so wird E in n Teilmengen (Teildatenbanken) zerlegt: $\{E_1, \ldots, E_n\}$. Nachdem B als Wurzel des Baums festgesetzt wurde, ist die zur Fertigstellung des Baums voraussichtlich erforderliche Informationsmenge:

$$G(B) = \sum_{j=1}^{n} \frac{|E_j|}{|E|} I(E_j)$$

G (gain) ist die (entsprechend der relativen Häufigkeit) gewichtete Summe der Einzelinformationen. Der Gewinn an Information wird dann berechnet als:

$$\text{gewinn}(B) = I(E) - G(B)$$

Es gilt, `gewinn` zu maximieren. Dazu geht man alle Attribute durch und wählt jenes aus, das den **maximalen Gewinn** liefert.
Wählen wir zunächst **Kreditwürdigkeit** als Attribut. Kreditwürdigkeit hat 3 Ausprägungen: **unbekannt, schlecht, gut**. Für jeden dieser Werte zählen wir, wie oft welches Risiko vorkommt:

Wert	hohes Risiko	mittleres Risiko	niedriges Risiko
unbekannt	2	1	2
schlecht	3	1	0
gut	1	1	3

Es ergibt sich:

$$I(\text{Kreditwürdigkeit\_unbekannt}) = -\frac{2}{5} \times \log_2(\frac{2}{5}) - \frac{1}{5} \times \log_2(\frac{1}{5}) - \frac{2}{5} \times \log_2(\frac{2}{5}) = 1,52$$

$$I(\text{Kreditwürdigkeit\_schlecht}) = -\frac{3}{4} \times \log_2(\frac{3}{4}) - \frac{1}{4} \times \log_2(\frac{1}{4}) = 0,81$$

$$I(\text{Kreditwürdigkeit\_gut}) = -\frac{1}{5} \times \log_2(\frac{1}{5}) - \frac{1}{5} \times \log_2(\frac{1}{5}) - \frac{3}{5} \times \log_2(\frac{3}{5}) = 1,37$$

Also ist

$$G(\text{Kreditwürdigkeit}) = \sum_{j=1}^{n} \frac{|E_j|}{|E|} I(E_j) = \frac{5}{14} \times 1,52 + \frac{4}{14} \times 0,8113 + \frac{5}{14} \times 1,37095 = 1,265$$

In Abb. 22 ist das Vorgehen dargestellt.

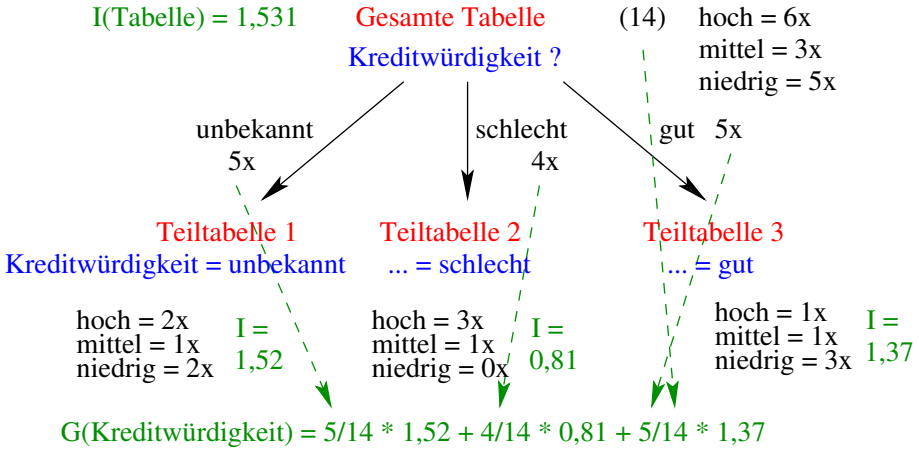


Abbildung 22: Gain-Berechnung

Diese Berechnung führen wir für alle 4 Attribute durch und erhalten:

- gewinn(kreditwuerdigkeit) $= 1,531 - 1,265 = 0,266$
- gewinn(einkommen) $= 1,531 - 0,564 = 0,967$
- gewinn(verschuldung) $= 1,531 - 1,468 = 0,063$
- gewinn(sicherheiten) $= 1,531 - 1,325 = 0,206$

Man wählt nun **einkommen** als obersten Knoten, da der Gewinn dort am größten ist, und setzt das Verfahren für **jeden Teilbaum rekursiv** fort.
Wir tun dies exemplarisch für den Zweig: *einkommen=15-35*. Wichtig ist jetzt, dass wir nicht mehr über die gesamte Tabelle nachdenken müssen, sondern nur noch über die Datensätze, wo *einkommen=15-35* gilt.
Wir können also in unserer ursprünglichen Tabelle 6 auf Seite 39 etliche Datensätze löschen: Tabelle 7. Wir könnten sogar die komplette Spalte für Einkommen herausstreichen.

Nr.	Zielattr. Risiko	frühere Kredit-würdigkeit	Verschul-dung	Sicher-heiten	Ein-kommen
2	hoch	unbekannt	hoch	keine	15 bis 35
3	mittel	unbekannt	niedrig	keine	15 bis 35
12	mittel	gut	hoch	keine	15 bis 35
14	hoch	schlecht	hoch	keine	15 bis 35

Tabelle 7: Kreditrisiko

Welches Attribut wählen wir als nächstes (nur für den Zweig *einkommen=15-35*)? Dazu berechnen wir zunächst wieder den Informationsgehalt der Tabelle 7.

- $p(\text{Risiko hoch}) = \frac{2}{4}$
- $p(\text{Risiko mittel}) = \frac{2}{4}$
- $p(\text{Risiko niedrig}) = \frac{0}{4}$

Folglich ist

$$I(\text{Tabelle2}) = I(\text{Risiko}) = -\frac{2}{4} \times \log_2(\frac{2}{4}) - \frac{2}{4} \times \log_2(\frac{2}{4}) - \frac{0}{4} \times \log_2(\frac{0}{4}) = 1$$

Nun wählt man wieder das Attribut aus, das den **maximalen Informationsgewinn** erzielt. D.h. man berechnet die Differenz aus $I(\text{Tabelle2})$ und der Information, die sich unter Beachtung der nächsten Ebene ergibt.
Wählen wir zunächst **Kreditwürdigkeit** als Attribut. Kreditwürdigkeit hat 3 Ausprägungen: **unbekannt, schlecht, gut**. Für jeden dieser Werte zählen wir, wie oft welches Risiko vorkommt:

Wert	hohes Risiko	mittleres Risiko	niedriges Risiko
unbekannt	1	1	0
schlecht	1	0	0
gut	0	1	0

Es ergibt sich:

$$I(\text{Kreditwürdigkeit\_unbekannt}) = -\frac{1}{2} \times \log_2(\frac{1}{2}) - \frac{1}{2} \times \log_2(\frac{1}{2}) - \frac{0}{2} \times \log_2(\frac{0}{2}) = 1$$

$$I(\text{Kreditwürdigkeit\_schlecht}) = -\frac{1}{1} \times \log_2(\frac{1}{1}) - \frac{0}{1} \times \log_2(\frac{0}{1}) - \frac{0}{1} \times \log_2(\frac{0}{1}) = 0$$

$$I(\text{Kreditwürdigkeit\_gut}) = -\frac{0}{1} \times \log_2(\frac{0}{1}) - \frac{1}{1} \times \log_2(\frac{1}{1}) - \frac{0}{1} \times \log_2(\frac{0}{1}) = 0$$

Also ist

$$G(\text{kreditwuerdigkeit}) = \sum_{j=1}^{n} \frac{|E_j|}{|E|} I(E_j) = \frac{2}{4} \times 1 + \frac{1}{4} \times 0 + \frac{1}{4} \times 0 = 0,5$$

Diese Berechnung führen wir auch für die anderen beiden Attribute durch und erhalten:

- $\text{gewinn}(\text{kreditwuerdigkeit}) = 1 - 0,5 = 0,5$
- $\text{gewinn}(\text{verschuldung}) = 1 - 0,6887 = 0,3113$
- $\text{gewinn}(\text{sicherheiten}) = 1 - 1 = 0$

Man wählt folglich **kreditwuerdigkeit** als nächsten Knoten, da der Gewinn dort am größten ist.

Der Gini-Index Alternativ kann man anstelle des Informationsgehalts und des Gains auch den Gini-Index verwenden. Der Gini-Index geht auf den italienischen Statistiker Corrado Gini (1884-1965) zurück.
Der **Gini-Index** ist das Äquivalent zum Informationsgehalt einer Tabelle bezüglich eines Zielattributs B; er wird gemessen als:

$$\text{gini}(B) = 1 - \sum_{i=1}^{k} p(b_i)^2$$

Die b_i stellen die möglichen Werte des Attributs B dar. p ist die Wahrscheinlichkeit (besser: relative Häufigkeit) für das Eintreffen von b_i.
Analog zum Gain definiert man dann

$$\text{GINI}(B) = \sum_{j=1}^{n} \frac{|E_j|}{|E|} \text{gini}(E_j)$$

5.2.5 C4.5-Algorithmus

Ein Nachteil des ID3-Algorithmus ist, dass er nicht mit numerischen Attributen umgehen kann. Dies kann der Nachfolger des ID3, der C4.5-Algorithmus. Und zwar werden numerische Attribute in **Intervalle** unterteilt und somit zu ordinalen Attributen. Hat ein Attribut A n Ausprägungen $A_1, \ldots, A_n$, so werden für jedes i die Intervalle $[a|a \leq A_i]$ und $[a|a > A_i]$ gebildet. Diese 2 Intervalle werden als neue Ausprägungen des Attributs A betrachtet. Es wird *die* Intervallbildung gewählt, die den größten Gewinn liefert.

Bemerkung 5.1 (ISplit)**.** Der ID3-Algorithmus hat einen weiteren Nachteil: Die Sortierung der Attribute (wie oben dargestellt) favorisiert Attribute mit *vielen* verschiedenen Ausprägungen (Attribute mit vielen *unterschiedlichen* Werten). Deshalb normalisiert C4.5 den Informationsgewinn. Sei:

$$ISplit(B) = -\sum_{j=1}^{n} \frac{|E_j|}{|E|} \times \log_2(\frac{|E_j|}{|E|})$$

Der **Gewinn an Information** wird dann normalisiert:

$$\text{gewinn'}(B) = \frac{gewinn(B)}{ISplit(B)}$$

Beispiel 5.3 (ISplit)**.** Betrachten wir einen kleinen Ausschnitt aus einer Kino-Besuch-Datenbank. Wir konzentrieren uns dabei auf nur *ein* Attribut, den *Kartenpreis* und das Zielattribut *Kino besucht*. Normalerweise müsste man *verschiedene* Attribute mit unterschiedlicher Werteanzahl – also z.B. Attribut 1 mit 3 Ausprägungen und Attribut 2 mit 8 Ausprägungen – betrachten. Um aber den Effekt zu veranschaulichen, betrachten wir nur *ein* Attribut mit zwei unterschiedlichen ordinalen Kodierungen.

Preis	8	5	5	9	8	8	4	5	8	8	9	8
Kino besucht	j	n	j	j	n	j	j	n	n	j	j	n

Wir betrachten 2 mögliche Umwandlungen des Preisattributs in Ordinalattribute. Variante 1:

- 4 und 5 werden zu *billig*: 4 Ausprägungen.
- 8 und 9 werden zu *teuer*: 8 Ausprägungen.

Variante 2:

- 4 wird zu *billig*: 1 Ausprägung.
- 5 wird zu *moderat*: 3 Ausprägungen.
- 8 wird zu *teuer*: 6 Ausprägungen.
- 9 wird zu *sehr teuer*: 2 Ausprägungen.

Für Variante 1 ergibt sich ein Gain von 0,97, für Variante 2 nur 0,73. Damit ist klar, dass der Gewinn bei Variante 2 größer sein wird.
Dies ist ein typischer Effekt, Attribute mit vielen Ausprägungen werden bevorzugt. Deshalb teilt man den Informationsgewinn durch den ISplit, um dies auszugleichen. Man erhält für Variante 1 einen ISplit von 0,92, für Variante 2 1,73. Der größere ISplit reduziert den Gewinn für Variante 2 stärker als den Gewinn bei Variante 1, womit der Bevorzugungseffekt ausgeglichen werden soll.

5.2.6 Probleme

Man kann beim Entscheidungsbaumlernen auf ein Problem stoßen. Es kann uns passieren, dass der Entscheidungsbaum die Trainingsdaten auswendig lernt, d.h. er wird dann alle Trainingsdaten korrekt klassifizieren. Allerdings hat sich gezeigt, dass ein solcher Entscheidungsbaum manchmal auf den Testdaten nicht gut funktioniert, da er ja alle Trainingsdaten nur auswendig gelernt hat. Man bezeichnet dies als **Overfitting**. Man versucht folglich, dies zu verhindern, und zwar durch ein Verkürzen der Bäume. Es gibt 2 Ansätze:

- Keine weiteren Unterbäume, wenn eine bestimmte Anzahl von Trainingsdaten unterschritten wird.
- Ersetzen bereits generierter Unterbäume durch *ein* Blatt.

Man beachte, dass wir damit den Wunsch aufgeben, dass unser Entscheidungsbaum die Trainingsdaten zu 100% korrekt vorhersagt. Dies kann aber ohnehin passieren, falls wir widersprüchliche Daten haben.

5.2.7 Ergänzungen

Das ID3-Verfahren ist ein Verfahren der Klasse **Top down induction of decision trees** (TDIDT-Verfahren). Es handelt sich um ein Verfahren, welches einen **univariaten** Baum erzeugt. D.h. an jedem Knoten wird exakt **ein** Attribut abgefragt. Es gibt auch Verfahren, die **multivariate** Entscheidungsbäume generieren. In diesen Bäumen können in einem Knoten mehrere Attribute benutzt werden, z.B. als Linearkombination von Attributen: Gewicht + 2 * Größe $<$ 70. Damit sind die Schnitte im Merkmalsraum zwar linear, aber nicht mehr achsenparallel. Man kann in den Knoten auch nichtlineare Ausdrücke abfragen: Gewicht / (Größe*Größe) $<$ 25. Nun sind die Schnitte im Merkmalsraum beliebig kurvig. Multivariate Bäume haben den Vorteil, dass sie meist genauer und kleiner sind. Allerdings ist ein Nachteil, dass sie schwieriger zu bauen und auch schwerer lesbar sind.

5.2.8 Aufgaben

Aufgabe 5.2 (Golfspiel)**.** Bestimmen Sie aus den in Tabelle 8 gegebenen Daten einen Entscheidungsbaum für das Attribut 'Play?', welches angibt, ob unter den gegebenen Witterungsbedingungen Golf gespielt wird. Wählen Sie bei gleicher Güte zweier Attribute das in der Tabelle weiter links stehende. Wie gehen Sie mit den numerischen Werten um?

Outlook	Temp (°F)	Humidity (%)	Windy?	Play?
sunny	85	85	false	no
sunny	80	90	true	no
overcast	83	78	false	yes
rain	70	96	false	yes
rain	68	80	false	yes
rain	65	70	true	no
overcast	64	65	true	yes
sunny	72	95	false	no
sunny	69	70	false	yes
rain	75	80	false	yes
sunny	75	70	true	yes
overcast	72	90	true	yes
overcast	81	75	false	yes
rain	71	80	true	no

Tabelle 8: Daten Golfspiel

Aufgabe 5.3 (Restaurant)**.** Die Tabelle 9 auf der nächsten Seite (vgl. Aufgabe 5.1 auf Seite 36) enthält 12 Datensätze mit diesen Attributen:

- Alternative: Gibt es in der Nähe ein anderes Restaurant? (ja/nein)
- Fr/Sa: Ist Freitag oder Samstag? (ja/nein)
- Hungrig: Bin ich hungrig? (ja/nein)
- Gäste: Wieviele Leute sind im Restaurant? (keine/einige/voll)
- Reservierung: Habe ich reserviert? (ja/nein)

- Typ: Um welche Art von Restaurant handelt es sich? (Franz./Chin./Ital./Burger)
- Wartezeit: Welche voraussichtliche Wartezeit wird vom Restaurant geschätzt? (0-10/10-30/30-60/>60)
- Warten (**Zielattribut**): Warte ich, wenn alle Tische besetzt sind? (ja/nein)

Generieren Sie einen Entscheidungsbaum und klassifizieren Sie nachfolgende Datensätze.

Alt.	Fr/Sa	Hung.	Gäste	Reserv.	Typ	Zeit	Warten
ja	nein	ja	einige	nein	Franz.	30-60	
ja	ja	ja	voll	ja	Chin.	10-30	
nein	nein	nein	keine	nein	Burger	0-10	

Alt.	Fr/Sa	Hung.	Gäste	Reserv.	Typ	Zeit	Warten
ja	nein	ja	einige	ja	Franz.	0-10	ja
ja	nein	ja	voll	nein	Chin.	30-60	nein
nein	nein	nein	einige	nein	Burger	0-10	ja
ja	ja	ja	voll	nein	Chin.	10-30	ja
ja	ja	nein	voll	ja	Franz.	>60	nein
nein	nein	ja	einige	ja	Ital.	0-10	ja
nein	nein	nein	keine	nein	Burger	0-10	nein
nein	nein	ja	einige	ja	Chin.	0-10	ja
nein	ja	nein	voll	nein	Burger	>60	nein
ja	ja	ja	voll	ja	Ital.	10-30	nein
nein	nein	nein	keine	nein	Chin.	0-10	nein
ja	ja	ja	voll	nein	Burger	30-60	ja

Tabelle 9: Daten Restaurantbeispiel

5.3 Verfahren zur Assoziationsanalyse

Wir wenden uns nun der Aufgabe zu, Assoziationsregeln zu finden. Der Standard-Algorithmus ist der **A-Priori-Algorithmus**.

5.3.1 Der A-Priori-Algorithmus

Der A-Priori-Algorithmus ist eines der wichtigsten iterativen Verfahren zur Erzeugung von *Assoziationsregeln* (vgl. Abschnitt 3.4). Er stellt eine Weiterentwicklung des AIS-Algorithmus dar, welcher 1993 veröffentlicht wurde. Ziel des A-Priori-Algorithmus ist es, *Frequent Itemsets* in der Menge aller Items zu finden. Mit *Item* bezeichnet man beliebige Objekte. *Frequent Itemsets* sind Itemmengen (Mengen von Objekten), deren relative Häufigkeit (ihr Support) den durch den Analysten festgelegten minimalen Schwellwert überschreiten.
Sei der minimale Support mit 2% festgelegt. Ein Frequent Itemset ist also (z.B.) bei einer Einkaufsdatenbank die Menge {Bier, Brot, Margarine}, falls diese 3er Kombination in mindestens 2% aller Einkäufe vorkommt.
Wieso reicht es, bei der Suche nach Assoziationsregeln sich auf Frequent Itemsets zu beschränken? Nun, eine Regel „Wer A und B kauft, kauft auch C“, erfüllt den geforderten, minimalen Support genau dann, wenn die Menge {A, B, C} ein Frequent Itemset ist.
Der A-Priori-Algorithmus wird in zwei Schritten vollzogen:

1. Finden von Frequent Itemsets (Kandidaten) mit ausreichendem Support
2. Erzeugen von **Assoziationsregeln** aus allen Frequent Itemsets

Der A-Priori-Algorithmus beschränkt sich bei der Generierung der Assoziationsregeln auf die Verwendung gefundener Frequent Itemsets, aus denen neue Itemsets zusammengesetzt werden. Begonnen wird mit **einelementigen Frequent Itemsets**. Im nächsten Durchlauf werden aus den

gefundenen einelementigen Frequent Itemsets **zweielementige**, im darauffolgenden Durchlauf aus den zweielementigen Itemsets **dreielementige** Itemsets (usw.) erzeugt. Der Vorgang wird solange wiederholt, bis keine Frequent Itemsets mehr gefunden werden. Die gefundenen Frequent Itemsets werden auch als *Kandidaten* bezeichnet.
Für jeden Kandidaten wird der Support berechnet. Ist er kleiner als der Schwellwert, wird der Kandidat verworfen.
Damit die Anzahl der auftretenden Kandidaten weiter reduziert und das zeitliche Laufzeitverhalten des Algorithmus verbessert werden kann, wird die sogenannte **Monotonieeigenschaft** der Itemsets verwendet. Sie besagt, dass jede nichtleere Teilmenge eines Frequent Itemsets wiederum ein Frequent Itemset sein muss. Der Support kann also nicht *kleiner* werden, wenn man aus einem Kandidaten für ein Frequent Itemset ein Item herausnimmt. Und nimmt man umgekehrt zu einem Frequent Itemset ein neues Item hinzu, so kann der Support *nicht* größer werden.
Was heißt das? Wir müssen – z.B. beim Erzeugen von 4er Kandidaten – nicht jede beliebige 4er Kombination von Items untersuchen, sondern nur solche 4er Kandidaten, die aus zwei 3er Kandidaten entstehen. Haben wir z.B. {A,B,C} und {A,C,D}, so könnte {A,B,C,D} ein Frequent Itemset sein. Diesen Kandidaten müssen wir also in Erwägung ziehen. Wissen wir aber, dass z.B. {B,C,D} *kein* 3er Frequent Itemset war, dann kann {A,B,C,D} wegen der Monotonie **kein** Frequent Itemset sein. Wir müssen also nicht in unserer Datenbank zählen, ob unser Kandidat den Support erfüllt. Wir wissen *vorher*, dass der Support *nicht* erfüllt wird. Das spart enorm Rechenzeit.

Generierung der Kandidaten Das Generieren der Kandidaten wird nun nochmal detailliert betrachtet. Diese Phase läuft in zwei Teilschritten ab:

- Join-Phase
- Pruning-Phase

In der Join-Phase werden alle $(k-1)$-langen Itemsets paarweise miteinander verbunden, die sich nur in einem Element unterscheiden. Es entstehen also k-elementige Itemmengen, bei denen sichergestellt ist, dass mindestens zwei Teilmengen der Länge $k-1$ ebenfalls Frequent Itemsets sind.
Nun wird in der Pruning-Phase für alle k-elementigen Kandidaten geprüft, ob *jede* $(k-1)$-elementige Teilmenge Frequent Itemset war (Monotonieeigenschaft).
Bei der Berechnung der Teilmengen ist es sinnvoll, Teilergebnisse wie Support und Konfidenz zwischenzuspeichern, um häufige Zugriffe auf die Datenbasis zu vermeiden.

Beispiel 5.4 (A priori)**.** Wir betrachten Kinobesuche und untersuchen, wer gern mit wem zusammen in's Kino geht.

Kinobesuch-ID	Kinobesucher
k_1	Anne, Claudia, Ernst
k_2	Anne, Ernst, Gudrun
k_3	Anne, Claudia, Ernst, Franz, Gudrun
k_4	Anne, Claudia, Horst
k_5	Bernd, Claudia, Ernst, Franz, Gudrun
k_6	Bernd, Claudia, Ernst, Gudrun, Horst

Wir fordern als minimalen Support: 50%.
Wir beginnen mit dem Zählen des Supports für die einzelnen Personen. Wir zählen also die relative Häufigkeit der Personen in unserer Kino-Datenbank mit 6 Datensätzen:

Anne	4	66%
Bernd	2	33%
Claudia	5	83%

Ernst	5	83%
Franz	2	33%
Gudrun	4	66%
Horst	2	33%

Was sehen wir? Bernd, Franz und Horst erfüllen **nicht** den minimalen Support. Bei der Bildung von 2er Frequent Itemsets können wir diese Drei ignorieren, denn jedes 2er FIS mit einem dieser Drei hat garantiert einen Support, der maximal 33% beträgt.
Nun bilden wir 2er FIS:

Anne, Claudia	50%	(1)
Anne, Ernst	50%	(2)
Anne, Gudrun	33%	(3)
Claudia, Ernst	66%	(4)
Claudia, Gudrun	50%	(5)
Ernst, Gudrun	66%	(6)

Einer dieser 6 Kandidaten erfüllt den Support nicht.
Jetzt bilden wir 3er Kandidaten. Dies tun wir aber nicht durch blindes Ausprobieren aller Kombinationen. Vielmehr bilden wir Kombinationen von 2er Frequent Itemsets, so dass ein 3er Kandidat entsteht.
Welche Kombinationen können wir bilden?

1+4	Anne, Claudia, Ernst	?%
1+5	Anne, Claudia, Gudrun	?%
2+6	Anne, Ernst, Gudrun	?%
4+5	Claudia, Ernst, Gudrun	?%

Müssen wir jetzt wieder zählen, um den Support zu bestimmen? **NEIN**, die beiden mittleren Kandidaten können wir sofort ausschließen, da Anne/Gudrun **kein** 2er FIS ist und somit die Kandidaten ACG und AEG aufgrund der Monotonie des Supports den geforderten Support nie und nimmer haben können.
Unser Kandidat ACE – hier müssen wir wieder zählen – erfüllt den Support nicht, CEG erfüllt mit 50% den geforderten Support.
Vierer-Kandidaten kann es nicht geben, da wir nur ein 3er FIS haben. Wir haben also 5 2er und 1 3er FIS.

Erzeugen der Regeln Wir zerlegen jedes Frequent Itemset X in 2 nichtleere Teilmengen A und $X \setminus A$ und betrachten die Regel:

$$A \to (X \setminus A)$$

Man kann z.B. für $X = \{a, b, c, d\}$ die Regel $(a, b) \to (c, d)$ betrachten. Wir müssen nun natürlich noch prüfen, ob die Regel die minimale Konfidenz erfüllt. Dazu müssen wir aber nicht noch einmal in der Datenbank zählen, da wir uns natürlich den Support für jedes Frequent Itemset gemerkt haben.
Wir müssen aber *jede* mögliche Zerlegung von X prüfen.
Für jedes häufig auftretende Itemset X müssen also Assoziationsregeln der Form $A \to (X \setminus A)$ – mit $A \subset X$ und $A \neq \emptyset$ – der Bedingung

$$\text{conf}(A \to (X \setminus A)) = \frac{\text{supp}(X)}{\text{supp}(A)} \geq \text{conf}_{\text{min}}$$

genügen. Es ist dabei nicht nötig, alle möglichen Assoziationsregeln auf ihre Konfidenz zu prüfen. Für ein Itemset $X = \{a, b, c, d\}$ und $A = \{a, b, c\}$ ist der Support von $A' = \{a, b\}$ größer (oder gleich) als der von A. Durch das Ersetzen von A mit A' kann die Konfidenz nur sinken. Wenn die Regel $A \to (X \setminus A)$ nicht die minimale Konfidenz erreicht, brauchen also alle Regeln der Form $A' \to (X \setminus A)$ mit $A' \subset A$ erst gar nicht betrachtet zu werden.
Das Ergebnis enthält alle Assoziationsregeln, die sowohl den minimalen Support als auch die minimale Konfidenz besitzen.
Wo liegt der **Vorteil** des Verfahrens? Es handelt sich um einfache Mengenoperationen. Und die Komplexität wird stark gesenkt. Überlegen Sie, wieviel Regelkandidaten beim naiven Ausprobieren aller Kombinationsmöglichkeiten (z.B. bei 1 Mio Verkaufsprodukten) entstünden.
Nachteilig ist aber, dass die Komplexität zwar gesenkt wird, man aber dennoch häufig bei großen Datenmengen Laufzeitprobleme durch die häufigen Datenbankabfragen (das für die Berechnung des Supports nötige Zählen) hat.

Beispiel 5.5 (A priori)**.** Wir setzen unser obiges Beispiel fort und betrachten den einzigen 3er Kandidaten: {Claudia, Ernst, Gudrun}. Wir greifen uns nur eine Variante heraus, um die Konfidenz zu berechnen:

$$\text{Claudia, Ernst} \rightarrow \text{Gudrun}$$

Den Support hatten wir mit 50% bereits berechnet. Wie hoch ist die Konfidenz?

$$\text{conf}(A \rightarrow B) = P(B|A) = \frac{\text{supp}(A \cup B)}{\text{supp}(A)}$$

Also:

$$\text{conf}(\text{Claudia, Ernst} \rightarrow \text{Gudrun}) = \frac{50}{66} = 0,75$$

Beachten Sie, dass wir die Support-Werte alle schon berechnet haben. Wir müssen also nicht erneut auf unsere Datenbank zugreifen. Das reduziert den Aufwand erheblich.

Beispiel 5.6 (A priori)**.** Seien folgende Frequent Itemsets der Länge **3** bereits berechnet (wir überspringen also die ersten 3 Schritte der Erzeugung der 1er, 2er, 3er Frequent Itemsets):

$$\{a, b, c\}, \{a, b, d\}, \{a, c, d\}, \{a, c, e\}, \{b, c, d\}$$

Daraus lassen sich folgende **4er-Kandidaten** erzeugen (Join):

$$\{a, b, c, d\} \quad (1{+}2)$$

$$\{a, c, d, e\} \quad (3{+}4)$$

$$\{a, b, c, e\} \quad (1{+}4)$$

Durch Pruning fallen $\{a, c, d, e\}$ und $\{a, b, c, e\}$ weg, da diese Mengen je eine (k-1)-Teilmenge enthalten ($\{c, d, e\}$ bzw. $\{b, c, e\}$), die *keine* 3-Frequent-Itemsets waren.

5.3.2 Modifikationen des A-Priori-Algorithmus

Hierarchische Assoziationsregeln Häufig hat man Begriffshierarchien, z.B. ist *Besteck* ein Oberbegriff für die Items *Gabel, Löffel, Messer*. Es ist meist eine gute Idee, Assoziationsregeln für *Besteck* zu suchen, als dies mit den Unterbegriffen *Gabel, Löffel, Messer* zu tun. Der Vorteil liegt auf der Hand: *Besteck* hat garantiert einen höheren Support als die Unterbegriffe.
Um hierarchische Regeln durch den A-Priori-Algorithmus finden zu lassen, werden die neuen Begriffe einfach als Items aufgenommen. Der Support eines Oberbegriffs ist gleich dem Support des Itemsets seiner Komponenten.
Zur Generierung der Regeln kann auch der *Basic-Algorithmus* verwendet werden. Dieser ist vergleichbar mit dem apriori-Algorithmus. Die Unterschiede sind, dass in den Transaktionen die Begriffe um die Oberbegriffe erweitert werden und dass alle Regeln, bei denen sich im Regelrumpf ein Nachkomme eines im Regelkopf enthaltenen Begriffs befindet, gelöscht werden.

Quantitative Assoziationsregeln Der A-Priori-Algorithmus kann analog angewandt werden. Die Idee ist, für alle Ausprägungen eines numerischen Attributs neue Items in die Menge aller Items einzufügen, indem man die numerischen Attribute diskretisiert, z.B. durch Intervalle.

Erzeugung von temporalen Assoziationsregeln Das Generieren von Assoziationsregeln über zeitliche Abläufe (Sequenzen) erfolgt in 5 Phasen:

1. Sortierphase: Während der Sortierphase werden die Transaktionen den einzelnen Kunden zugeordnet und in die chronologische Reihenfolge gebracht. Dadurch entstehen die sogenannten Kundensequenzen.
   Bei der Erstellung der Sequenzen ist es möglich, Zeitrahmen zu beachten. Das heißt, dass bei sehr großen Zeitabständen die Transaktionen nicht als eine Sequenz gewertet werden, sondern mehrere Sequenzen gebildet werden. Bei sehr kleinen Zeitabständen werden Transaktionen zusammengefasst. Wenn zum Beispiel ein Kunde innerhalb von 10 Minuten zweimal einkauft, werden die 2 Transaktionen als eine gewertet, da der Kunde wahrscheinlich nur etwas vergessen hat und nochmal zurückkehrte.

2. Generierungsphase: Hier werden die häufigen Itemmengen gesucht. Der Support bestimmt sich durch den Anteil der Kunden, die den Begriff in irgendeiner ihrer Transaktionen enthalten. Der Support wird nur einmal pro Kunde gezählt.
3. Transformationsphase: Die Sequenzen werden durch die in ihnen enthaltenen häufigen Itemmengen ersetzt.
4. Sequenzphase: Mit Hilfe der häufigen Itemmengen werden die gewünschten Sequenzen ermittelt. Der Ablauf ist wie beim apriori-Algorithmus.
5. Maximalphase: Aus der Menge der generierten Itemmengen werden die maximalen Sequenzen ermittelt.

Aufgabe 5.4 (Frequent Itemsets).
Seien diese Einkäufe gegeben:

ID	gekaufte Artikel
t_1	Brot, Saft, Cola, Bier
t_2	Saft, Cola, Wein
t_3	Brot, Saft, Wasser
t_4	Cola, Bier, Saft
t_5	Brot, Saft, Cola, Bier, Wein
t_6	Wasser

Ergänzen Sie zunächst für die gegebenen Assoziationsregeln folgende Tabelle:

Regel	Support	Konfidenz
Saft → Cola		
Cola → Saft		
Cola → Bier		
Bier → Cola		

Finden Sie dann alle Frequent Itemsets mit einem minimalen Support von 0,4.

5.4 Lineare Regression

Eine Regressionsfunktion beschreibt den Trend bzw. den durchschnittlichen Zusammenhang zwischen metrischen Attributen.
Gegeben sind dabei die Instanzenbeschreibung X und der davon abhängige Zielwert Y anhand einer Beispielmenge $E = X \times Y$. Der Zusammenhang zwischen diesen lässt sich nun mit Hilfe der Regressionsfunktion $\hat{y} = f(x)$ beschreiben. Ziel ist dabei, den quadratischen Fehler zwischen tatsächlichem und berechnetem Zielwert zu minimieren.

$$\text{error} = \sum (y_i - \hat{y}_i)^2 \rightarrow min$$

Seien diese x/y-Werte gegeben: (2,3), (3,7), (4,6), (5,10). Mittels linearer Regression erhält man:

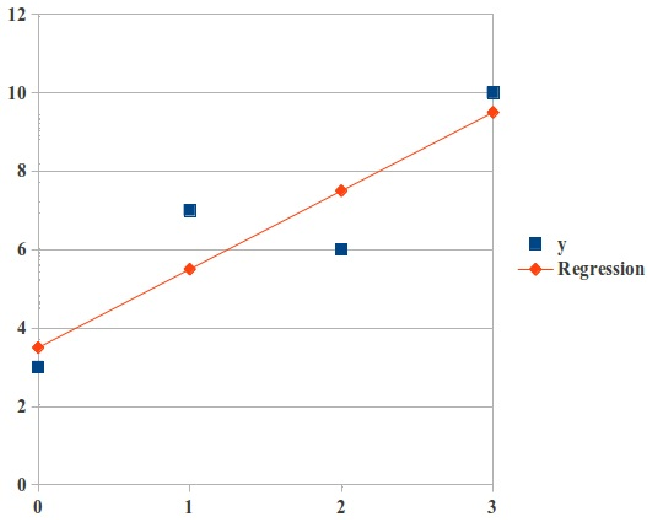


Abbildung 23: Beispiel Regression

Aufgabe 5.5 (Ausfuhränderung). Berechnen Sie aus den nachfolgend gegebenen Daten die durchschnittliche jährliche Änderung der Ausfuhr der BRD für die Periode 1980/81. Folgende Entwicklung der Ausfuhr der BRD (jeweils gegenüber dem Vorjahr) wurde beobachtet:

Periode 19..	73/74	74/75	75/76	76/77	77/78	78/79	79/80
Ausfuhränderung	+29 %	-4 %	+16 %	+7 %	+4 %	+10 %	+11 %

5.5 Überwachte und selbstorganisierende unüberwachte neuronale Netze

Neuronale Netze werden im Data Mining sehr häufig eingesetzt. Für tiefergehende Informationen sei hier auf die Vorlesungsreihe „Neuronale Netze“ sowie die einschlägige Literatur (z.B. [LC12]) verwiesen.

5.6 Verfahren zur Clusterbildung

Wir befassen uns nun mit Verfahren, die Objekte zu geeigneten Mengen (**Clustern**) zusammenfassen (vgl. Abschnitte 3.2 auf Seite 20, 4.6 auf Seite 30).

5.6.1 Partitionierendes Clustering

Man wählt bei diesen Verfahren eine (beliebige) Anfangspartitionierung von k Clustern. Man kann diese beispielsweise durch den Centroid, vgl. Abschnitt 4.6 auf Seite 30, repräsentieren. Aus dieser Anfangspartitionierung werden nun die Objekte schrittweise so zwischen den Clustern geordnet, dass die Güte der Gruppierung sich immer weiter verbessert. Dazu wird in jedem Rechendurchlauf für jeden Cluster der Centroid berechnet. Anschließend werden alle Objekte demjenigen Cluster zugeordnet, dessen Centroid sie am nächsten sind. Das Verfahren endet, wenn kein Objekt mehr einem anderen Cluster zugeordnet wird, d.h. die Güte der Partitionierung sich nicht weiter verbessert.
Partitionierende Clusterverfahren teilen die Eingabedaten in disjunkte Cluster ein, so dass gilt:

- Jeder Cluster besteht aus mindestens einem Objekt.
- Jedes Objekt ist höchstens in einem Cluster enthalten.

Der Algorithmus sieht so aus:

1. Erzeuge (zufällig) k Anfangscluster C_i. Alle Objekte x aus den Eingabedaten werden (zufällig) einem der Cluster zugeordnet.
2. Bestimme die Mittelpunkte $\overline{x_1}$, $\overline{x_2}$, ..., $\overline{x_k}$ der Cluster.
3. Für alle x aus den Eingabedaten: Weise x demjenigen Cluster C_i zu, von dessen Mittelpunkt $\overline{x_i}$ es am wenigsten entfernt ist.
4. Gehe zu 2, falls mindestens ein x einem anderen Cluster zugewiesen wurde.

Dazu werden zwei Verfahren (k-Means und k-Medoid) vorgestellt. Beide gehören zur Klasse der Mittelpunkt-Verfahren. Jedes Individuum gehört dabei immer nur zu **einer** Klasse. Ergänzend sei darauf hingewiesen, dass es Verfahren gibt, die wahrscheinlichkeitsbasiert arbeiten. Jedes Individuum kann dabei verschiedenen Clustern zugeordnet sein, jeweils mit einer gewissen Wahrscheinlichkeit.

k-Means Der k-Means-Algorithmus ist ein einfaches und populäres Verfahren zur Clusteranalyse. Dabei wird die Anzahl der gesuchten Cluster vorgegeben, deren Zentren zunächst zufällig festgelegt und iterativ adaptiert werden. Die Cluster werden durch den *Centroid* (Schwerpunkt) repräsentiert. In Abbildung 24 auf der nächsten Seite ist das Vorgehen dargestellt.

1. Eingabedaten
2. initiales Clustering
3. Centroidbestimmung
4. Neueinteilung der Cluster
5. Gehe zu 3.

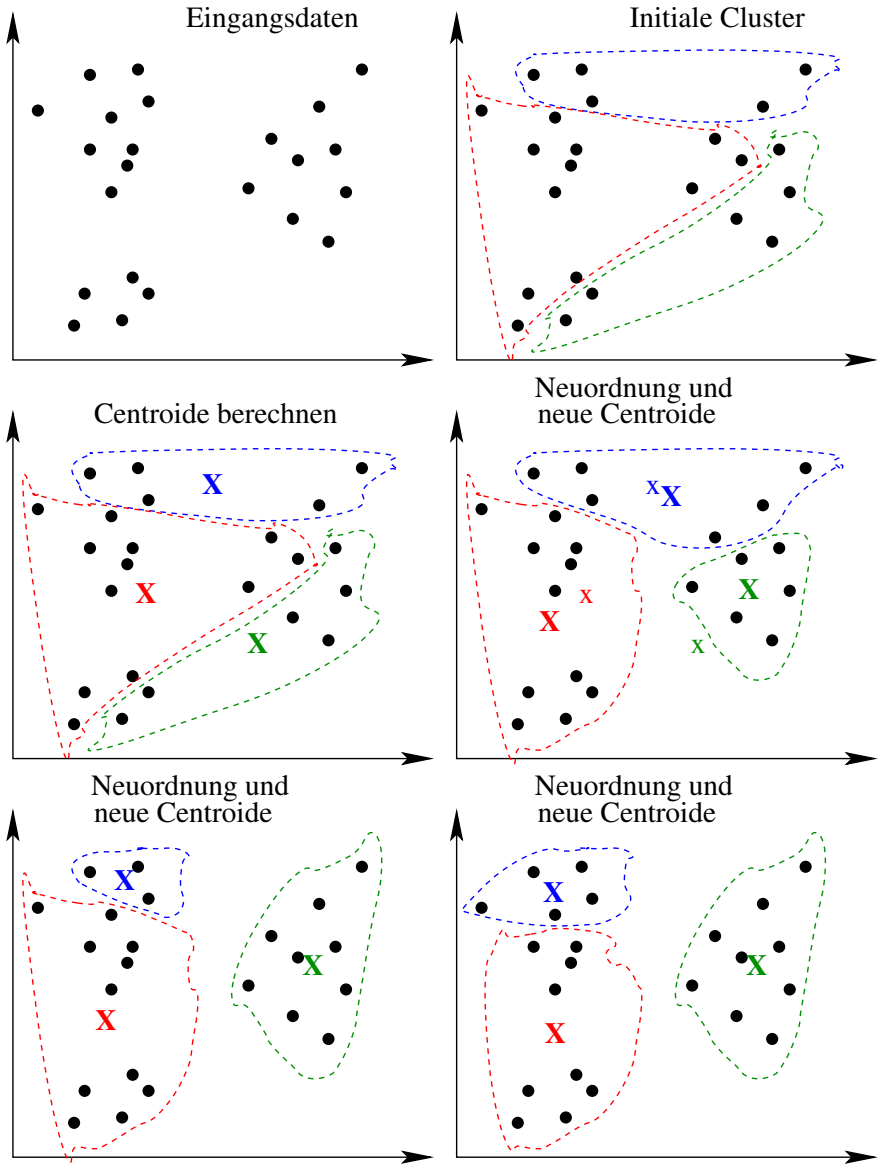


Abbildung 24: Clustering mit k-Means

Die Abbildung zeigt den initialen Schritt der (zufälligen) Zuweisung der Objekte zu einem Cluster, die Berechnung der Centroide und die Neuberechnung der Cluster. Manche k-Means-Verfahren weichen vom vorgestellten Algorithmus in dem Punkt ab, dass die Centroide der neuen Cluster nicht erst nach der Umordnung aller Punkte, sondern bei jeder Umordnung angepasst werden.

Vorteile von k-Means

- Anzahl der Iterationen vergleichsweise klein.
- Einfache Implementierung, deswegen ist es ein populäres Clusteringverfahren.

Nachteile von k-Means

- Anfällig bzgl. Rauschen und Ausreißern, da alle Objekte in die Berechnung des Centroiden eingehen.
- Gute initiale Cluster sind oft schwer zu bestimmen.
- Ergebnis und Laufzeit sind abhängig von der initialen Zerlegung.

Betrachten wir ein Beispiel genauer. In Abb. 25 sind die Daten gegeben, die wir clustern möchten. Wir benutzen die Euklidische Distanz. Wir möchten 3 Cluster bilden und wählen die Punkte (1, 5),

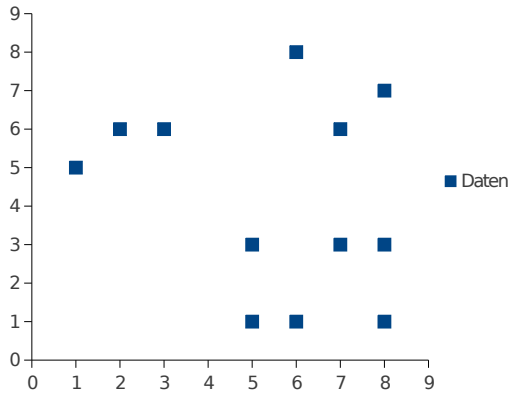


Abbildung 25: k-Means – Ausgangssituation

(2, 6), (3, 6) als Clustermittelpunkte unserer initialen Cluster.
Nun starten wir das k-Means-Verfahren. Die initialen Cluster sind relativ unglücklich gewählt, so dass die Punkte (1, 5) und (2, 6) zunächst „allein“ bleiben. Alle anderen Punkte werden dem Cluster (3, 6) zugeordnet, da ihre Euklidische Distanz zu (3, 6) geringer ist als zu (1, 5) und (2, 6). Im nächsten Schritt wandert der Punkt (3, 6) zum Cluster (2, 6), da sein Abstand zu (2, 6) geringer als zum Clustermittelpunkt (6.3, 3.9) ist. Der Clustermittelpunkt des Clusters wird zu (2.5, 6) (Abb. 26).
Im 3. und 4. Schritt setzt sich der Trend fort, dass unser Cluster (2.5, 6) weitere Punkte vom rechten Cluster stiehlt. Die Mittelpunkte verschieben sich dadurch. Im 4. Schritt sieht man, dass nun auch der Cluster (1,5) aktiv wird und der Punkt (2,6) ihm zugeordnet wird (Abb. 27 auf der nächsten Seite).
Im 5. Schritt erhalten wir nun endlich die Cluster, die wir erwartet haben. Im Schritt 6 ändert sich nichts mehr, so dass das Verfahren beendet werden kann (Abb. 28).

k-Medoid Ein Verfahren, das dem k-Means-Verfahren sehr ähnlich ist, ist das k-Medoid-Verfahren. Der wesentliche Unterschied zu k-Means ist, dass als Repräsentant des Clusters nicht der Schwerpunkt, sondern der sogenannte Medoid verwendet wird. Dieser muss ein Objekt aus den Eingabedaten sein.
Das k-Medoid-Verfahren versucht ständig, einen Medoid und einen Nichtmedoid zu tauschen. Der Nichtmedoid wird zum neuen Medoid, der vorherige Medoid zu einem Nichtmedoid. Dabei muss sich natürlich die Qualität der Cluster verbessern.

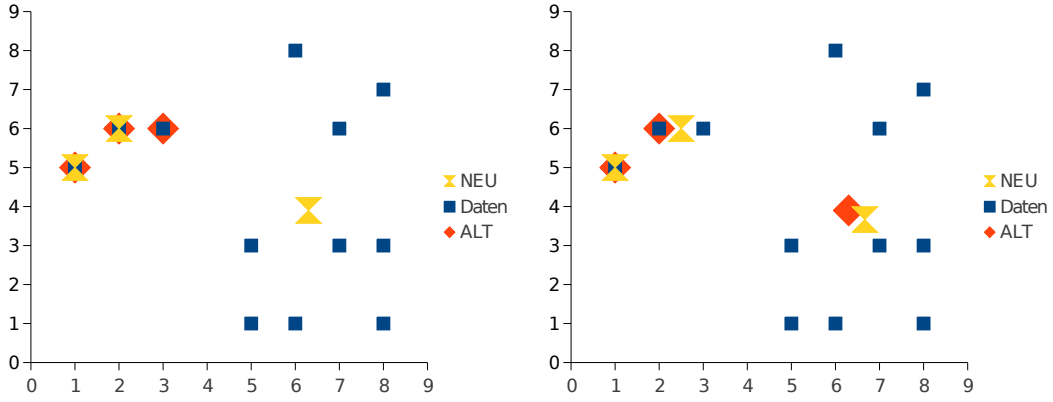


Abbildung 26: k-Means – 1./2. Schritt

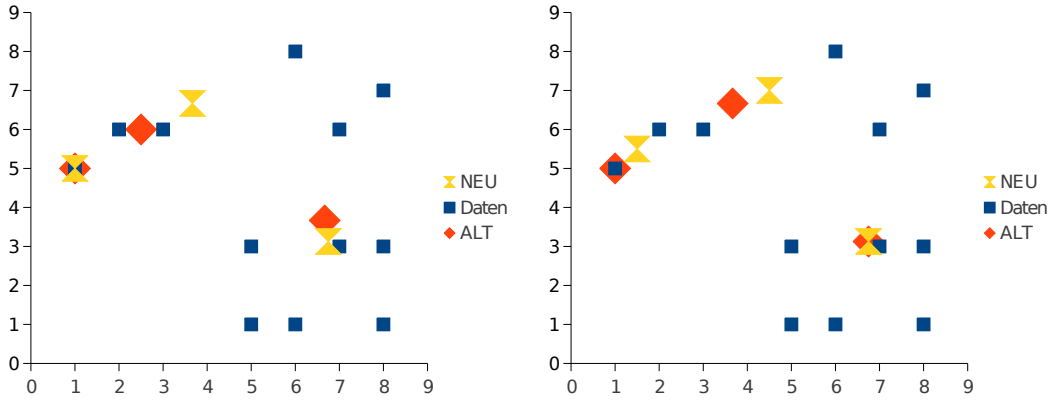


Abbildung 27: k-Means – 3./4. Schritt

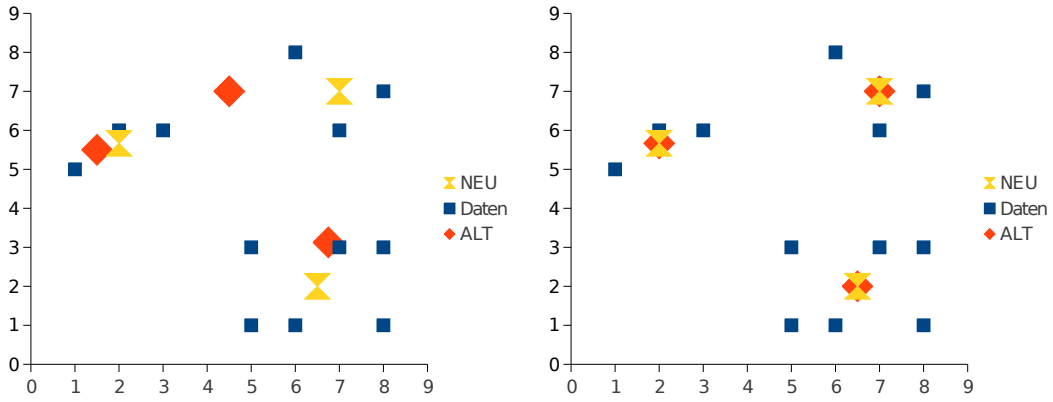


Abbildung 28: k-Means – 5./6. Schritt

Ein bekannter Algorithmus ist **Partitioning Around Medoids (PAM)**, der eine sehr gründliche Suche in den Eingabedaten durchführt. Dadurch bewertet er die Güte des Clusterings für nahezu alle möglichen Kombinationen von Medoiden und ist deshalb in der Lage, gute Cluster zu finden. Der Nachteil ist allerdings, dass die Laufzeit mit der Anzahl der Eingabeobjekte sehr stark ansteigt und PAM daher nur für kleine Mengen zu clusternder Objekte geeignet ist. Eine weniger gründliche Suche führt **Clustering Large Applications based on RANdomized Search (CLARANS)** durch, der nicht den kompletten Eingaberaum absucht, sondern zufallsbasiert nur einen Teil der Daten berücksichtigt. Dadurch ist er viel effzienter, ohne dabei wesentlich schlechtere Ergebnisse als PAM zu liefern.

Aufgabe 5.6 (k-Means)**.** Gegeben seien die folgenden (zweidimensionalen) Datensätze:

x	3	6	8	1	2	2	6	6	7	7	8	8
y	5	2	3	5	4	6	1	8	3	6	1	7

Finden Sie für diese Datensätze 3 Cluster mittels k-Means (d.h. bilden Sie drei Cluster)! Verwenden Sie die ersten drei Datentupel als Anfangszentren der Cluster. Berechnen Sie dann die Zugehörigkeit aller Datensätze zu einem der 3 Cluster. Nun berechnen Sie auf Basis der gefunden Cluster den Medoid/Centroid neu. Dann ordnen Sie wieder alle Datensätze erneut den Clustern zu usw. Verfolgen Sie die Wanderung der Zentren.

Aufgabe 5.7 (k-Means)**.** Erläutern Sie, wie man bei dem k-Means-Verfahren den Centroid berechnet.

5.6.2 Hierarchisches Clustering

Beim hierarchischen Clustering wird eine Hierarchie von Clustern derart aufgebaut, dass immer die Cluster mit minimaler Distanz (größter Ähnlichkeit) verschmolzen werden. Es können z.B. übergeordnete Cluster aus der Vereinigung bereits bestehender Cluster gebildet werden. Eine solche Clusterhierarchie ist gerade dann von Vorteil, wenn Daten zu analysieren sind, die natürlicherweise eine Hierarchie darstellen. Der Ablauf dieses Verfahrens lässt sich in einer Baumstruktur (Dendrogramm) darstellen (Abb. 29).
Ein **Dendrogramm** ist ein Baum, dessen Knoten jeweils einen Cluster repräsentieren und der folgende Eigenschaften besitzt:

- Die Wurzel repräsentiert die gesamte Datenmenge.
- Ein innerer Knoten repräsentiert die Vereinigung aller Objekte, die in den darunterliegenden Teilbäumen enthalten sind.
- Die Blätter repräsentieren einzelne Objekte.

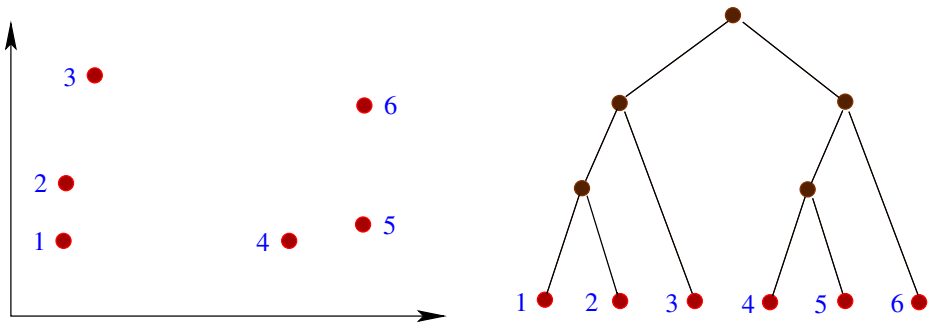


Abbildung 29: Hierarchisches Clustering

Man unterscheidet zwei Verfahren:

Agglomeratives Clustering Beim agglomerativen Clustering wird mit der kleinsten Auflösung von Clustern angefangen, den einzelnen Objekten selbst, d.h. anfangs beinhaltet jeder Cluster genau *ein* Objekt. Die zwei jeweils ähnlichsten Cluster werden dann zu einem hierarchisch höheren Cluster verschmolzen. Dieser Vorgang wird wiederholt, bis nur noch ein Cluster vorhanden ist.
In unserem Beispiel befinden sich die einzelnen Punkte (1,2,3,4,5,6) aus dem Diagramm auf der untersten Ebene (Blätter) des Dendrogramms. Dies ist Schritt eins. Im zweiten Schritt werden nun zwei Cluster zu einem zusammengefügt , die von ihrer Distanz her am nächsten sind . Für unser Beispiel bedeutet dies, dass nun Cluster (Punkt) 1 und 2 sowie 4 und 5 zu einem Cluster verschmelzen, da sie die geringste Distanz zueinander haben. Alle anderen Kombinationen der Punkte ergeben eine größere Distanz. Aus diesem Grund werden auch zunächst nur diese Cluster zu jeweils *einem* Cluster verschmolzen. Im nächsten Schritt wird dann der Cluster 1/2 mit dem Cluster 3 verschmolzen sowie Cluster 4/5 mit Cluster 6.
Nochmal: Ziel ist es, genau die Cluster zusammenzufügen, die die geringste Distanz / größte Ähnlichkeit zueinander haben.
Im letzten Schritt werden nun die beiden Cluster zu einem verschmolzen, so dass nur noch ein Cluster vorhanden ist.

Divisives Clustering Das divisive Clustering baut eine Hierarchie in umgekehrter Reihenfolge auf. Es behandelt die gesamte Datenmenge anfangs als einen großen Cluster und teilt sie in kleinere, hierarchisch tiefere Cluster ein. Dies erfolgt solange, bis in jedem Cluster genau ein Objekt vorhanden ist.
Nachteilig kann bei beiden Vorgehensmodellen sein, dass einmal gebildete Cluster (frühe Verschmelzungs- oder Aufteilungsentscheidungen) nicht wieder aufgelöst werden können. Der **Vorteil** dieser Verfahren ist, dass die Anzahl der Cluster nicht vorgegeben werden muss.
Hierarchische Verfahren eignen sich besonders dann gut, wenn man an den Verbindungen zwischen den Clustern interessiert ist. Implementierungen hierfür sind weit verbreitet. Aufgrund der nötigen paarweisen Distanzbildung für alle Objekte sind hierarchische Verfahren schlecht skalierbar und in der Praxis auf wenige tausend Elemente beschränkt. Weiterhin haben hierarchische Verfahren Probleme mit Ausreißern.

5.6.3 Erwartungsmaximierung

Im Gegensatz zu den vorherigen Methoden werden beim Clustering nach Erwartungsmaximierung (EM) die Objekte nicht bestimmten Clustern eindeutig zugeordnet. Objekte können zu mehreren Clustern gehören, jeweils mit einer bestimmten Wahrscheinlichkeit.
Eine Wahrscheinlichkeitsverteilung ist eine Abbildung, die jedem möglichen Ausgang eines Zufallsexperiments eine Wahrscheinlichkeit zuordnet. Beim Clustering durch Erwartungsmaximierung nimmt man also an, dass die Daten aus einem Zufallsexperiment entstanden sind, und approximiert die Cluster durch z.B. Gaußverteilungen. Eine Gaußverteilung ist eine symmetrische Wahrscheinlichkeitsverteilung, die einen Graphen in Glockenform induziert. Gaußverteilungen werden benutzt, weil sich – durch eine Mischung dieser – beliebige andere Verteilungen approximieren lassen. Man nimmt nun an, dass die Eingabedaten aus einer Mischung von **k Gaußverteilungen** entstanden sind. Das Ziel des EM-Algorithmus ist es daher, die k Gaußverteilungen zu finden, für die die Wahrscheinlichkeit, dass die gegebenen Daten aus ihnen entstanden sind, maximal ist. Dazu beginnt man ähnlich wie beim k-Means- oder k-Medoid-Verfahren mit beliebigen Startwerten und verbessert das Clustering iterativ.
Man beginnt also mit k zufällig gewählten Gauß-Verteilungen. Nun berechnet man die **Wahrscheinlichkeiten**, mit denen

- ein Punkt x (Objekt) aus einer
- der k Gaußverteilungen $C_i (i = 1, \ldots, k)$ entstanden ist.

$$P(x|C_i) = \frac{1}{\sqrt{(2\pi)^k |\sum_{C_i}|}} \times e^{-\frac{1}{2}(x-\mu_{C_i})^T (\sum_{C_i})^{-1} (x-\mu_{C_i})}$$

Dabei ist:

- k die **Anzahl der Cluster**
- $\sum_{C_i}$ die **Kovarianzmatrix** für die Punkte im Cluster C_i
- μ_{C_i} der Vektor des **Mittelpunkts** des Clusters i

Jetzt berechnet man die **Gesamt-Wahrscheinlichkeitsdichte** für x:

$$P(x) = \sum_{i=1}^{k} W_i \times P(x|C_i)$$

W_i ist dabei die Anzahl der Objekte im Cluster C_i, geteilt durch die Anzahl aller Objekte, also eine Gewichtung entsprechend der Größe des Clusters (relative Häufigkeit).
Die Wahrscheinlichkeit, mit der ein bestimmtes Objekt x zu einem Cluster C_i gehört, ist (Satz von Bayes):

$$P(C_i|x) = W_i \times \frac{P(x|C_i)}{P(x)}$$

Um zu prüfen, ob die gegebenen Daten mit maximaler Wahrscheinlichkeit aus den errechneten Gaußverteilungen entstanden sind, werden alle (für alle Objekte) Wahrscheinlichkeitsdichten zu *einer* **Gesamt-Wahrscheinlichkeitsdichte** summiert:

$$E = \sum_{x \in D} \log(P(x))$$

E soll **maximiert** werden.
Der Algorithmus arbeitet iterativ. Er berechnet zu den initialen Belegungen die Wahrscheinlichkeiten $P(x_i), P(x_j|C_i)$ und $P(C_i|x_j)$, um dann aus diesen neue Mittelwerte der k Cluster zu errechnen, aus denen sich dann wiederum neue Wahrscheinlichkeiten ergeben. Dies wird solange wiederholt, bis E nicht mehr erhöht werden kann.
Rechnerisch kann nun ein Objekt mehreren Clustern angehören, nämlich gerade denen, für die $P(C_i|x_j) > 0$ ist. Um der Einschränkung des partitionierenden Clustering zu genügen, dass ein Objekt höchstens einem Cluster angehört, kann man die Objekte demjenigen Cluster C_i zuordnen, für den $P(C_i|x_j)$ maximal ist.
Clustering nach Erwartungsmaximierung hat ebenso wie die Verfahren nach Mittelpunktbestimmung Probleme, Cluster zu finden, die bestimmte Eigenschaften aufweisen. So können beispielsweise Cluster, die stark unterschiedliche räumliche Strukturen besitzen, schlecht von solchen Verfahren erkannt werden. Für solche Cluster bietet sich das dichtebasierte Clustering an.

5.6.4 Dichtebasiertes Clustering

Statt einen Cluster als eine Menge von Objekten, die möglichst nah an einem gewissen Mittelpunkt liegen, kann man ein Cluster auch als eine Menge von Objekten ansehen, die in einer bestimmten Dichte zueinander stehen und von anderen Clustern durch Regionen geringerer Dichte getrennt werden. Die lokale Punktdichte für jeden Punkt im Cluster muss eine vorher festgelegte Mindestdichte erreichen. Die Menge von Objekten, die einen Cluster ausmacht, ist räumlich zusammenhängend. Cluster, die stark unterschiedliche räumliche Strukturen besitzen, werden schlecht von anderen Verfahren erkannt. In solchen Fällen ist das dichtebasierte Clustering von Vorteil.

5.7 Naive Bayes

Wir kehren zur Klassifikation zurück, bleiben aber beim wahrscheinlichkeitsbasierten Ansatz. Das Ziel ist nun die Vorhersage der wahrscheinlichsten Klasse. Dabei findet kein Training eines Modells statt. Die Vorhersage wird direkt aus den Trainingsdaten berechnet. Dabei gehen wir von einer wichtigen **Grundannahme** aus: Alle Attribute sind voneinander unabhängig.

5.7.1 Bayessche Formel

Sei Y ein Ereignis mit $P(Y) > 0$. Dann heißt

$$P(X|Y) = \frac{P(X \wedge Y)}{P(Y)}$$

bedingte Wahrscheinlichkeit von X unter der Bedingung Y. Falls $P(Y) = 0$, so ist die bedingte Wahrscheinlichkeit nicht definiert.
Stellt man die Formel für die bedingte Wahrscheinlichkeit um, so erhält man:

$$P(X \wedge Y) = P(Y) * P(X|Y)$$

Betrachtet man die bedingte Wahrscheinlichkeit von Y bezüglich X, so ergibt sich analog:

$$P(Y \wedge X) = P(X) * P(Y|X)$$

Da das logische Und kommutativ ist, gilt:

$$P(Y) * P(X|Y) = P(X) * P(Y|X)$$

Somit erhält man die *Bayessche Formel*:

$$P(X|Y) = \frac{P(Y|X) * P(X)}{P(Y)}$$

5.7.2 Berechnungsgrundlagen

Sei $A = \{a_1, \ldots, a_n\}$ eine Menge von Attributwerten, z.B. $\{sunny, hot\}$. Wir möchten berechnen, ob (nicht) gespielt wird (yes/no), wenn es *sunny* und *hot* ist.
Mit der Bayesschen Formel erhalten wir:

$$P(yes|[sunny, hot]) = \frac{P([sunny, hot]|yes) * P(yes)}{P([sunny, hot])}$$

Analog berechnen wir

$$P(no|[sunny, hot]) = \frac{P([sunny, hot]|no) * P(no)}{P([sunny, hot])}$$

Wir sagen dann *yes* vorher, wenn $P(yes|[sunny, hot])$ größer als $P(no|[sunny, hot])$ ist, sonst *no*.
Wir sehen, dass in beiden Formeln im Nenner der Term $P([sunny, hot])$ vorkommt. Da dieser für beide Ausdrücke gleich ist und wir ja nur bezüglich der Größe vergleichen, können wir diesen weglassen. Wir reden aber nun nicht mehr von P, sondern von **Likelihood** L:

$$L(yes|[sunny, hot]) = P([sunny, hot]|yes) * P(yes)$$
$$L(no|[sunny, hot]) = P([sunny, hot]|no) * P(no)$$

Der Naive-Bayes-Algorithmus geht von der Unabhängigkeit der Attribute aus und ersetzt:

$$P([sunny, hot]|yes) = P(sunny|yes) * P(hot|yes)$$

Nun nochmal in der Übersicht: Die Vorhersage der Klasse erfolgt über folgende Werte.

Relative Häufigkeit $h[a, k]$ des Attributs a in der Klasse k

Likelihood $L[k](A) = \prod_{i=1}^{n} h[a_i, k] \quad \times h[k]$

Wahrscheinlichkeit $P[k_j](A) = \frac{L[k_j](A)}{\sum_i L[k_i](A)}$

Vorhersage $k_m : P_m(A) = \max_j(P[k_j](A))$

Die relative Häufigkeit $h[a, k]$ ist exakt unser $P(a|k)$. Wir ersetzen aber $P(a|k)$ lieber durch die relative Häufigkeit $h[a, k]$, da wir die Information über die *Wahrscheinlichkeit* nicht wirklich haben, sondern nur eine kleine Menge von gegebenen Beispieldaten. Die Likelihood hatten wir oben bereits eingeführt.
Wir vergleichen die Likelihoods miteinander und nehmen den größten Wert. Um wieder Wahrscheinlichkeitswerte zu erhalten, normalisieren wir unsere Likelihoods, so dass deren Summe gerade 1 ergibt. Dies erreichen wir gerade dadurch, dass wir alle Likelihoods durch die Gesamtsumme *aller* Likelihoods dividieren: $\sum_i L[k_i](A)$.
Wir erläutern das Vorgehen nochmal an einem Beispiel.

5.7.3 Beispiel Wetterdaten

Wir beziehen uns wieder auf das Wetter-Beispiel (Tabelle 10 auf der nächsten Seite).

Tag	outlook	temperature	humidity	windy	play
1	sunny	hot	high	false	no
2	sunny	hot	high	true	no
3	overcast	hot	high	false	yes
4	rainy	mild	high	false	yes
5	rainy	cool	normal	false	yes
6	rainy	cool	normal	true	no
7	overcast	cool	normal	true	yes
8	sunny	mild	high	false	no
9	sunny	cool	normal	false	yes
10	rainy	mild	normal	false	yes
11	sunny	mild	normal	true	yes
12	overcast	mild	high	true	yes
13	overcast	hot	normal	false	yes
14	rainy	mild	high	true	no

Tabelle 10: Wetter-Daten

Wird an einem Tag mit folgenden Eigenschaften gespielt?

outlook = sunny
temperature = hot
humidity = normal
windy = true

5.7.4 Naive-Bayes-Algorithmus

Schritt 1 Ermittlung der **relativen Häufigkeiten**

$$h[sunny, yes] = \frac{2}{9}$$

weil

- es 9 Tage gibt, an denen gespielt wurde (*play=yes*), und
- an 2 dieser Tage *outlook=sunny* war.

		play	
		yes	no
outlook	sunny	**2/9**	**3/5**
	overcast	4/9	0/5
	rainy	3/9	2/5
temperature	hot	**2/9**	**2/5**
	mild	4/9	2/5
	cool	3/9	1/5
humidity	high	3/9	4/5
	normal	**6/9**	**1/5**
windy	true	**3/9**	**3/5**
	false	6/9	2/5

Schritt 2 Berechnung der **Likelihoods**

$$
\begin{aligned}
&L[yes](sunny, hot, normal, true)\\
&\quad = h[sunny, yes] * h[hot, yes] * h[normal, yes] * h[true, yes] * h[yes]\\
&\quad = \quad 2/9 \quad * \quad 2/9 \quad * \quad 6/9 \quad * \quad 3/9 \quad * 9/14\\
&\quad \approx 0{,}007055\\
&L[no](sunny, hot, normal, true)\\
&\quad = h[sunny, no] * h[hot, no] * h[normal, no] * h[true, no] * h[no]\\
&\quad = \quad 3/5 \quad * \quad 2/5 \quad * \quad 1/5 \quad * \quad 3/5 \quad * 5/14\\
&\quad \approx 0{,}010286
\end{aligned}
$$

Schritt 3 Berechnung der **Wahrscheinlichkeiten** (wir kürzen *sunny,hot,normal,true* mit A ab).

$$
\begin{aligned}
P[yes](A) &= L[yes](A) / (L[yes](A) + L[no](A))\\
&= 0{,}007055 \,/\, (0{,}007055 + 0{,}010286)\\
&\approx 0{,}406835\\
P[no](A) &= L[no](A) \,/\, (L[yes](A) + L[no](A))\\
&= 0{,}010286 \,/\, (0{,}007055 + 0{,}010286)\\
&\approx 0{,}593165
\end{aligned}
$$

Schritt 4 **Vorhersage** Wir sagen *die* Klasse vorher, die die größte Wahrscheinlichkeit liefert.

$$(P[yes](A) = 40,68\%) < (P[no](A) = 59,31\%) \quad \Rightarrow NO$$

5.7.5 Aufgaben

Aufgabe 5.8 (Naive Bayes). Tabelle 11 (vgl. Aufgabe 5.3 auf Seite 45) enthält 12 Datensätze mit folgenden Attributen:

- Alternative: Gibt es in der Nähe ein geeignetes anderes Restaurant? (ja/nein)
- Fr/Sa: Ist Freitag oder Samstag? (ja/nein)
- Hungrig: Bin ich hungrig? (ja/nein)
- Gäste: Wieviele Leute sind im Restaurant? (keine/einige/voll)
- Reservierung: Habe ich reserviert? (ja/nein)
- Typ: Um welche Art von Restaurant handelt es sich? (Franz./Chin./Ital./Burger)
- Wartezeit: Welche voraussichtliche Wartezeit wird vom Restaurant geschätzt? (0-10/10-30/30-60/>60)
- Warten (**Zielattribut**): Warte ich, wenn alle Tische besetzt sind? (ja/nein)

Berechnen Sie mit Naive Bayes, ob wir in folgenden Fällen warten oder nicht.

Alt.	Fr/Sa	Hung.	Gäste	Reserv.	Typ	Zeit	Warten
ja	nein	ja	einige	nein	Franz.	30-60	
ja	ja	ja	voll	ja	Chin.	10-30	
nein	nein	nein	keine	nein	Burger	0-10	

Alt.	Fr/Sa	Hung.	Gäste	Reserv.	Typ	Zeit	Warten
ja	nein	ja	einige	ja	Franz.	0-10	ja
ja	nein	ja	voll	nein	Chin.	30-60	nein
nein	nein	nein	einige	nein	Burger	0-10	ja
ja	ja	ja	voll	nein	Chin.	10-30	ja
ja	ja	nein	voll	ja	Franz.	>60	nein
nein	nein	ja	einige	ja	Ital.	0-10	ja
nein	nein	nein	keine	nein	Burger	0-10	nein
nein	nein	ja	einige	ja	Chin.	0-10	ja
nein	ja	nein	voll	nein	Burger	>60	nein
ja	ja	ja	voll	ja	Ital.	10-30	nein
nein	nein	nein	keine	nein	Chin.	0-10	nein
ja	ja	ja	voll	nein	Burger	30-60	ja

Tabelle 11: Restaurant-Daten

Aufgabe 5.9 (Naive Bayes)**.** Seien folgende Datensätze gegeben (a=angest, s=selbst, v=verh, l=ledig, M=Miete, E=Eigentum)

Beruf	a	a	a	a	s	s	s	s
Fam.st.	v	l	v	v	l	l	v	v
Kinder	j	n	n	j	j	n	j	n
Wohnung	M	E	M	E	E	M	E	E

Wo lebt ein lediger Angestellter mit Kindern?

6 Datenvorbereitung

Experience is something you don't get until just after you need it.
Olivier's Law

Bisher haben wir uns um die in der Praxis wohl schwierigste Aufgabe gedrückt: die Datenvorbereitung. Unter Vorbereitung fassen wir die 3 ersten Teilprozesse des KDD-Prozesses (vgl. Abbildung 2 auf Seite 7) zusammen.

6.1 Motivation

Nicht alle Daten können direkt aus der Datenbank in ein Data-Mining-Verfahren übertragen werden. Sie müssen vielfach erst aufbereitet werden. Dazu ist es nötig, sich mit den konkreten Daten auseinanderzusetzen. Welche Probleme können auftauchen?

- **Wertebereiche** und **Ausreißer**. Eine genauere Betrachtung der Wertebereiche der Attribute ist insbesondere bei metrischen Daten von Bedeutung. So ist es häufig sinnvoll, nicht jede einzelne Ausprägung eines Attributs zu berücksichtigen. Alternativ könnten Ausprägungen gruppiert werden (z.B. Altersgruppen). Bei eingehender Betrachtung der Wertebereiche fallen meist auch sogenannte *Ausreißer* auf – einzelne Ausprägungen, die sehr stark von den anderen abweichen (z.B. ein 75- und ein 90-jähriger in einer Gruppe von 30- bis 50-jährigen). Ob solche Ausreißer für das Data Mining ausgeblendet oder adaptiert werden sollten oder lieber doch im Originalzustand zu verwenden sind, hängt vom konkreten Kontext ab.
- **Fehlende, ungenaue, falsche, widersprüchliche Werte**. Verschiedene Faktoren können dazu führen, dass ein Datensatz *fehlende*, *ungenaue* oder gar *falsche Attributwerte* enthält. Die Ursachen können z.B. in einer Befragung selbst begründet sein. So können befragte Personen die Beantwortung einzelner Fragen aus persönlichen Gründen verweigern, oder gar falsche Angaben machen. Ein weiterer Grund für fehlende Attribute kann die Tatsache sein, dass die Struktur der Datenbank verändert wurde und bereits erfasste Datensätze nicht überarbeitet wurden. Und natürlich muss auch immer mit der Möglichkeit von Flüchtigkeitsfehlern gerechnet werden.
  Wie geht man nun mit solchen Daten um? Beim Umgang mit fehlenden Daten stellt sich die Frage nach der Bedeutung des Fehlens. Ist die Tatsache des Fehlens selbst eine Information, die sich auf das Ergebnis des Data Minings auswirkt? Oder kann die Ursache des Fehlens vielleicht einen Einfluss auf das Ergebnis haben? In vielen Fällen hilft es, fehlende Werte durch spezielle Ausprägungen zu ersetzen, die nicht auftreten können (z.B. negative Zahlen bei Stückzahlen).
- **Dimensionsreduktion**. Meist liegen in einer Datenbank zu einem Datensatz viele Attribute vor. Dies führt beim Data Mining zu teilweise hochdimensionalen Problemstellungen. Dabei treten zwei wesentliche Probleme auf:
  - Die visuelle Wahrnehmungsfähigkeit des Menschen beschränkt sich auf den 3-dimensionalen Raum bzw. den 4-dimensionalen Raum-Zeit-Eindruck.
  - Eine hohe Dimensionalität bedeutet auch immer einen hohen Rechenaufwand und somit erhöhte Programmlaufzeiten. Diese sind aber für etliche Anwendungsfälle inakzeptabel.

  Die Reduktion einer zu hohen Dimensionalität kann auf zweierlei Weise erfolgen. Zum einen können einzelne Attribute einfach **ausgeblendet** werden. Zum anderen kann versucht werden, **abhängige Komponenten zusammenzufassen**.
- **Datentransformation, Skalierung etc.** Daten können in Formaten vorliegen, die für das jeweilige Data-Mining-Verfahren unpassend sind. Z.B. können bestimmte Angaben in mm, andere in km angegeben sein.

Die Datenvorbereitung spielt beim Data Mining eine wichtige Rolle, da die Qualität der Daten nicht nur für das Laufzeitverhalten des eigentlichen Data-Mining-Prozesses, sondern auch für die Qualität

des Resultats wichtig ist. Data Mining ist ein typisch iterativer Prozess: Man probiert etwas aus, berechnet ein Resultat und prüft dieses. Häufig ist man natürlich nicht gleich im *ersten* Durchlauf erfolgreich. Nach allgemeiner Erfahrung liegt der Aufwand in diesem iterativen Prozess mit etwa 80% bei der Datenvorbereitung. Die Datenvorbereitung ist aus einem ganz einfachen Grund für die Qualität der Resultate essentiell. Wenn man schlechte (bzw. schlecht vorbereitete) Daten hat, so kann man nicht erwarten, dass die Verfahren dies ausbügeln. Es gibt ein altes Prinzip, welches auch hier zutrifft, GIGO: garbage in, garbage out.

6.2 Arten der Datenvorbereitung

Folgende Klassen der Vorbereitung unterscheidet man:

Datenselektion und -integration Daten auswählen und zusammenfügen. Daten aus unterschiedlichen Quellen werden zu einer großen Datenbank vereinigt.

Datensäuberung Daten werden bereinigt.

Datenreduktion Daten werden reduziert, z.B. bezüglich der Dimension.

Datentransformation Daten werden umgewandelt, um so adäquate Darstellungsformen (in Abhängigkeit vom jeweiligen Verfahren) für die Daten zu bekommen.

Man beachte, dass Datensäuberung und -reduktion gemäß dem KDD-Bild (Abb. 2 auf Seite 7) zum 2. Schritt (Datenvorverarbeitung) gehören.

6.2.1 Datenselektion und -integration

Meistens hat man es mit dezentralen Datenbanken zu tun. Daraus ergeben sich natürlich Probleme:

- Jede Filiale hat unter Umständen ihre eigene Datenbank.
- Es ist eine unterschiedliche Semantik und Syntax der Attribute möglich.

Die Datenintegration fügt Daten mehrerer Datensätze unterschiedlicher Quellen (verschiedene Datenbanken) zusammen. Ergebnis sollte ein schlüssiger Datensatz sein. Folgende Probleme können hierbei auftreten:

Entitätenidentifikationsproblem Welche Merkmale besitzen dieselbe Semantik? Hat man z.B. 2 Attribute `Kunden_ID` und `Kundennummer`, so stellt sich die Frage, ob beide Attribute das gleiche bedeuten. Eine Lösung dieses Problems bietet oftmals die Nutzung von Metadaten (Daten, die die Daten beschreiben).

Redundanzen Redundanzen können durch Inkonsistenzen in der Nomenklatur von Attributen oder Dimensionen auftreten. Betrachtet man z.B. die Attribute `Name` und `name`, so sind beide Attribute syntaktisch unterschiedlich, besitzen aber wohl dieselbe Semantik.

Widersprüche Beim Zusammenfügen von unterschiedlichen Datenquellen können Widersprüche in den Datenbanken auftreten, z.B. unterschiedliche Adressen für die gleiche Person.

Datenwertkonflikte Es kann vorkommen, dass in den Datensätzen eines Attributs unterschiedliche Werte auftreten. Beispiel: Entfernung in Meilen und Kilometern.

6.2.2 Datensäuberung

Warum Datensäuberung? Daten in der realen Welt sind unvollständig, mit Fehlern oder Ausreißern behaftet oder inkonsistent. Dies kann in Vorbereitungsschritten beseitigt werden (data cleaning), was oftmals von großer Wichtigkeit ist, weil unvollständige Daten zu Konfusion (und damit Fehlern) beim DM-Prozess führen können. Daraus können sich dann unzuverlässige Resultate ergeben.
Es ist allerdings wichtig, beim Bereinigen darauf zu achten, dass möglichst keine neue Information hinzugefügt wird. D.h. eingefügte Werte/Daten sollten informationsneutral sein, um die vorhandenen Informationen, die aus der realen Welt stammen, nicht zu verzerren oder zu verfälschen. Dies kann man in der Praxis häufig nicht absichern, man sollte es aber als Ziel im Blick behalten.

Was muss bereinigt werden?

- Fehlende Daten
- Verrauschte Daten
- Falsche Daten
- Inkonsistente Daten

Fehlende Daten Bei fehlenden Werten muss man beachten: Sind es Fehler im eigentlichen Sinn, oder lässt sich aus dem Fehlen eines oder mehrerer Werte eine Information ableiten (z.B. wenn jemand bei einem Kreditantrag Felder nicht ausfüllt)? Dies muss man für den jeweiligen Sachverhalt entscheiden. Folgende Möglichkeiten hat man, um auf fehlende Daten zu reagieren:

- Attribut ignorieren
  Aber seien Sie vorsichtig (s.o.): Die leeren Felder könnten als Information gewertet werden. Und beachten Sie, dass man dies nur dann machen kann, wenn man genügend *vollständige* Attribute hat.
- Fehlende Werte manuell einfügen
  Das manuelle Einfügen ist gerade bei großen Datenmengen sehr zeitintensiv. Außerdem kann es praktisch undurchführbar sein. Beispielsweise fehlten beim Data Mining Cup 2001 [DMC] für etwa 500 bzw. 1000 Kunden (Lern- bzw. Testdaten) die Werte für `Kunde_seit`.
- Globale Konstante zum Ausfüllen des fehlenden Werts (z.B.: `unbekannt`, `minus unendlich`)
  Diese Methode ist sinnvoll, wenn ein leeres Feld als Information angesehen wird oder viele Werte fehlen.
- Durchschnittswert aller Einträge derselben Klasse
  Dies ist eine recht einfache Möglichkeit der Datensäuberung. Sie bietet sich an, wenn die (numerischen) Werte der Klasse dicht beieinander liegen. Denn dann ist die Annahme gerechtfertigt, dass die fehlenden Werte auch in diesem Bereich liegen. Dies geht i.allg. auch sehr schnell.
- Wahrscheinlichsten Wert eintragen (anhand statistischer Methoden ermitteln)
  Hierfür sollte man sich entscheiden, wenn bei einem Fall oder einigen wenigen Fällen kein Wert vorhanden ist und man genügend Anhaltspunkte für einen sinnvollen/begründeten Wert hat. Das Finden des wahrscheinlichsten Werts kann aber sehr aufwändig sein.
- Datensatz als fehlerhaft kennzeichnen und von Weiterverarbeitung ausnehmen
  So kann man vorgehen, wenn man genug vollständige Daten hat (DMC 2001: bei Lerndaten insg. 10.000 und bei Testdaten 17.128) und Einfügen mit unverhältnismäßig hohem Aufwand verbunden ist oder mit unbefriedigenden Ergebnissen (Veränderung der Semantik der Daten) zu rechnen ist.

Achtung: Das Einfügen von Werten (nach welcher Methode auch immer) führt zur Veränderung des Datenbestands und kann die Qualität der Daten beeinflussen. Ebenso kann die Semantik der Daten verfälscht werden. Ignorieren/Weglassen verfälscht die Daten bezüglich der Wahrscheinlichkeitsverteilung der Attributwerte. Wir verletzen damit unser Ziel, dass jegliche Veränderung unseres Datenbestands **informationsneutral** sein sollte. Wir haben aber bei fehlenden Daten keine andere Chance.

Verrauschte Daten sind fehlerhafte Werte, die meist durch ungenaue Messwerte oder Schätzungen entstehen. Man kann diese Daten glätten (angleichen) durch:

- Klasseneinteilung (binning): Man gruppiert die Daten und ersetzt sie durch Mittelwerte oder Grenzwerte.
- Verbundbildung (clustering): Hier werden die Ausreißer durch Verbundbildung erkannt. Man steckt ähnliche Werte in Gruppen. Die Ausreißer liegen dann außerhalb dieser Gruppen.

- Kombinierte Maschine/Mensch-Untersuchung (combined computer and human inspection): Der Computer erstellt Liste (scheinbar) befremdlicher Werte, danach filtert der Mensch die Ausreißer aufgrund von Erfahrungswerten heraus.
- Regression: Man beschreibt die Daten als mathematische Funktion. Dann ersetzt man die realen Datenwerte durch die berechneten Funktionswerte der gefundenen Funktion.

Inkonsistente und falsche Daten können z.B. durch Verletzen der referenziellen Integrität (z.B.: Verweis eines Fremdschlüssels auf ein nicht existierendes Schlüsselattribut) auftreten.

Beispiel 6.1 (Inkonsistente Daten)**.** Seien die folgenden Tabellen gegeben:

Kundendaten

Ku-Nr.	Name	Ort
1	Meier	Wismar
2	Schulze	Schwerin
3	Lehmann	Rostock

Bestellungen

Bestell-Nr.	Ku-Nr.
1	2
3	1
2	1
4	4

In der Bestellungen-Tabelle wird auf den Kunden 4 verwiesen, den es aber gar nicht gibt.

Folgende Korrekturen sind möglich:

- Zuhilfenahme anderer Datensätze
- Künstliches Anlegen eines fehlenden Schlüsselattributs
- Falls nicht möglich: betreffenden Datensatz löschen oder als fehlerhaft markieren.

Beispiel 6.2 (Fehlerhafte, unzulässige Werte)**.**

- Verletzter Wertebereich: Wenn auf einstellige natürliche Zahlen beschränkt, dann dürfen keine Zahlen $x < 0$ oder $x > 9$ auftauchen.
- Verletzte Plausibilitätsbeziehungen: Sonst umsatzschwacher Kunde hat plötzlich sehr hohen Jahresumsatz.

6.2.3 Datenreduktion

Motivation: Bei vielen Data-Mining-Aufgaben sind die Datensätze sehr umfangreich. Dies kann dazu führen, dass Data Mining mit diesen Datenmengen sehr aufwändig bzw. unmöglich ist. Durch Datenreduktion versucht man, dieses Problem zu mindern oder zu lösen.

Strategien Folgende Techniken können angewendet werden.

1. Aggregation
2. Dimensionsreduktion
3. Datenkompression
4. Numerische Datenreduktion

Aggregation Unter **Aggregation** (auch Verdichtung) versteht man das Zusammenfassen von Fakten zu *einem Fakt* oder das Generalisieren der Daten. So kann man z.B. Daten durch ihre Mittelwerte ersetzen oder Teilwerte zu einer Gesamtsumme zusammenfassen (s. auch Abschnitt 6.2.4 auf Seite 67).

Dimensionsreduktion Bei der Dimensionsreduktion werden irrelevante Daten (Attribute) vernachlässigt und relevante Daten einbezogen.

- Schrittweise Vorwärtsauswahl: Gute Attribute werden schrittweise in die Zielmenge eingegliedert.
- Schrittweise Rückwärtseliminierung: Ausgehend von der Gesamtmenge werden schlechte Attribute schrittweise eliminiert.
- Kombination aus beiden

Datenkompression Die Daten werden entweder transformiert oder kodiert, um eine Reduktion der Datenmenge und damit eine Reduktion der Komplexität des Data Mining zu erhalten. Z.B. kann man mehrere Binärattribute zu einem Byte zusammenfassen. Ebenso kann man die einzelnen, separaten Attribute für die Produkte *Löffel, Gabel, Messer ...* zu **einem** Attribut *Besteck* zusammenfassen.

Numerische Datenreduktion Eine numerische Datenreduktion kann man auf der Basis von Stichproben realisieren. Man nimmt also nicht die gesamte Datenmenge, sondern nur eine – natürlich viel kleinere – Stichprobe. Zu klären ist natürlich die Frage, wie man die Stichprobe bildet.

Zufällige Stichprobe Es werden zufällig Datensätze aus der Quelldatenmenge ausgewählt.

Repräsentative Stichprobe Es werden zufällig Datensätze aus der Quelldatenmenge ausgewählt, allerdings so, dass die Stichprobe repräsentativ ist. Man kann die Auswahl unter Beachtung der Häufigkeitsverteilungen bestimmter Attribute treffen. Ebenso sollte man absichern, dass jede Attributausprägung – bei nominalen und ordinalen Attributen – vorkommt. Bei Klassifikationsproblemen sollte jede Klasse vorkommen.

Geschichtete Stichprobe Wie bei der repräsentativen Stichprobe werden Datensätze zufällig ausgewählt. Es wird aber darauf geachtet, dass wichtige Attribute auch einen Wert im Datensatz besitzen.

Inkrementelle Stichproben Eine erste Stichprobe wird nach der Datenanalyse Schritt für Schritt erweitert. Dies kann nach einem der oberen Verfahren geschehen.

Average Sampling Die Quelldatenmenge wird in Teile gespalten und jeder Teil unabhängig von den anderen einer Analyse unterzogen. Die Ergebnisse werden im Anschluss daran gemittelt und so zu einem Gesamtergebnis vereint.

Selektive Stichprobe Aus der Quelldatenmenge werden unergiebige Datensätze herausgefiltert. Anschließend wird eine Stichprobenziehung durchgeführt.

Windowing Ähnlich wie bei der inkrementellen Stichprobe wird hier von einer Basisstichprobe ausgegangen. Diese wird nach der Analyse aber nur um ergiebige Datensätze (von denen Data Mining Methoden viel lernen können) erweitert. Sprich: Zu Beginn wird nur ein Teil der Trainingsdaten (Datenfenster) verwendet. Dann werden nur noch Trainings-Datensätze in die Trainingsmenge aufgenommen, bei denen sich das erlernte Verfahren (z.B. ein Entscheidungsbaum) falsch verhält.

Clustergestützte Stichprobe Bei dieser Methode werden ähnliche Datensätze in Clustern zusammengefasst und ein Repräsentant gewählt, der im Anschluss für die Analyse herangezogen wird.

Alternativ kann man eine numerische Datenreduktion auch über lineare Regression erreichen. Dabei ersetzt man einfach die Daten durch die Koeffizienten einer linearen Regressionsfunktion. D.h. die Daten werden durch eine Funktion ersetzt, die mittels linearer Regression ermittelt wurde.

6.2.4 Datentransformation

Oftmals sind die Daten in ihrer ursprünglichen Form nicht besonders gut zum Data Mining geeignet. Die Datentransformation hat die Aufgabe, die Daten in eine Form umzuwandeln, die für das jeweilige Data-Mining-Verfahren geeignet ist. Folgende Transformationen können erforderlich sein:
Anpassung:

1. von unterschiedlichen Datenstrukturen
2. der Datentypen
3. von Konvertierungen oder Kodierungen
4. von Zeichenketten
5. von Datumsangaben
6. von Maßeinheiten und Skalierungen

Weitere Transformationen können in Betracht kommen:

1. Kombination oder Separierung von Attributen
2. ggf. Berechnung abgeleiteter Werte
3. Datenaggregation
4. Datenglättung

Anpassung unterschiedlicher Datenstrukturen Innerhalb der Datenbank kann es Attribute mit nicht kompatiblen Datenstrukturen geben. Diese müssen gegebenenfalls angepasst werden.

Anpassung der Datentypen Betrachtet man die 2 als Datum, so stellt sich die Frage: Ist diese 2 vom Datentyp `Char` oder `Integer`, und sollte eine Umwandlung stattfinden?

Anpassung von Konvertierungen oder Kodierungen Datenquelle und Zielmenge benutzen evtl. unterschiedliche Kodierungen. Diese muss man anpassen, um die Korrektheit der Semantik sicherzustellen.
Z.B. erzeugt die **Binärkodierung** aus Attributen mit einer bestimmten Anzahl Merkmalsausprägungen eine Menge binärer Attribute. Jeder Merkmalsausprägung wird ein neues, binäres Attribut zugeordnet, das den Wert 1 annimmt, wenn die Ausprägung in einem einzelnen Datensatz vorkommt und sonst den Wert 0 besitzt (vgl. Beispiel 4.8 auf Seite 29). Dieses Verfahren kann z.B. das Attribut Kaufverhalten mit den Ausprägungen `Vielkäufer`, `Seltenkäufer` und `Nichtkäufer` so transformieren, dass man das Attribut Kaufverhalten durch 3 Binärattribute Vielkäufer, Seltenkäufer, Nichtkäufer ersetzt. Auf diese Weise kann ein qualitatives Attribut in mehrere binärkodierte Attribute überführt werden. Das Binärkodierungsverfahren bereitet qualitative Attribute für Algorithmen vor, die numerische Attribute erfordern. Bei der Anwendung der Binärkodierung ist zu beachten, dass die Performanz der Mustererkennung durch die steigende Attributanzahl beeinträchtigt werden kann.
Diskretisierung wird angewendet, um den Wertebereich von quantitativen Attributausprägungen in endlich vielen Teilmengen zusammenzufassen. Die Diskretisierung kann z.B. bei der Verallgemeinerung des Alters sinnvoll sein, da auf diese Weise die Altersinformationen zu Altersgruppen {jung, mittel, alt} zusammengefasst werden können und so eine Reduzierung der Attributausprägungen erreicht wird.

Anpassung von Zeichenketten Kann das Data-Mining-Programm mit Umlauten, Groß- und Kleinschreibung sowie Leerzeichen in den Datensätzen umgehen, oder sollte hier eine Umwandlung erfolgen?

Anpassung von Datumsangaben Datumsangaben sind oft unterschiedlich kodiert (auf Grund von unterschiedlichen Formaten in verschiedenen Ländern, 12.03.2003; 12-03-03; 03-12-03) und erfordern daher in den meisten Fällen eine Anpassung. Es können auch Daten aus unterschiedlichen Zeitzonen vorhanden sein, die auf jeden Fall angepasst werden müssen, da sonst das Ergebnis verfälscht werden könnte.

Anpassung von Maßeinheiten und Skalierungen Wie bei den Datumsangaben sind bei Maßeinheiten oft nationale Standards Grund für die Notwendigkeit von Anpassungen (z.B. Inch; Zentimeter; Yard; Meter). Die **Normalisierung** transformiert sämtliche Merkmalsausprägungen eines Attributs auf die Werte einer stetigen, numerischen Skala (z.B. [0,1]: $x_{neu} = \frac{x - \min(x_i)}{\max(x_i) - \min(x_i)}$). Dabei werden alle Werte durch den ermittelten Maximalwert dividiert oder zunächst um den Minimalwert subtrahiert (dann ist das Minimum 0) und anschließend durch den Maximalwert dividiert.
Eine andere Normalisierungstechnik berechnet den statistischen Mittelwert und die Standardabweichung der Attributwerte, subtrahiert den Mittelwert von jedem Wert und dividiert dann das Ergebnis durch die Standardabweichung. Normalisierung kann dann angewendet werden, wenn Minimum und Maximum eines Attributes gegeben sind. Die Normalisierung kann z.B. zur Kodierung des Alters eingesetzt werden. Der Minimalwert hierbei sind 0 Jahre und der Maximalwert z.B. 100 Jahre. Ein Alter von 40 Jahren würden dann – auf einer Skala von 0 bis 1 – mit 0,4 kodiert werden.

Kombination oder Separierung von Attributen Es kann nötig sein, verschiedene Attribute zu einem neuen zusammenzufügen wie z.B. Tag, Monat und Jahr zu Datum. Umgekehrt kann es aber auch erforderlich sein, das Datum in seine Bestandteile zu zerlegen, um so z.B. den Monatsanfang erkennen zu können.

Berechnung abgeleiteter Werte Durch das Berechnen abgeleiteter Werte können ganz neue Attribute aufgenommen werden. So kann das Attribut *Gewinn* durch die Differenz der Attributwerte *Ausgaben* und *Einnahmen* berechnet werden.

Datenaggregation Oft liegen die Daten in einer zu feinen Aggregationsebene vor. Wird zum Beispiel die Einwohnerzahl von Berlin gesucht und liegen nur die Einwohnerzahlen der einzelnen Stadtteile vor, so kann man durch Aggregation (in diesem Fall Summenbildung) die Daten in eine höhere Aggregationsebene überführen.

Datenglättung Die Hauptidee der Datenglättung (s. auch verrauschte Daten) ist, dass jeder (numerische) Wert aus der gegebenen Datenmenge durch die Gleichung

$$\text{Wert}(i) = \text{Mittelwert}(i) + \text{Rauschen}$$

dargestellt werden kann, wobei i der i-ten Instanz entspricht. Das Ziel der Methode ist, die ursprüngliche Wertmenge durch eine reduzierte Menge zu ersetzen, da man hofft, dass solche Werte zu besseren Lösungen führen. Darüber hinaus wird das Rauschen in den Daten reduziert.
Man kennt vier Techniken, mit denen Daten geglättet werden können:

- Eingruppierung (*binning*)
- Clustering
- kombinierte menschliche und maschinelle Kontrolle
- Regression

Fazit Datentransformation dient dazu, Daten zwischen verschiedenen Formen zu transformieren, um eine konsistente Semantik der Attributwerte zu garantieren oder neue Attribute aus bestehenden Attributen zusammenzusetzen. Es können auch bestehende Attribute in neue Attribute aufgeteilt oder Daten zwischen verschiedenen Ebenen aggregiert werden, um eine geänderte Sicht auf die Daten zu erhalten.

6.3 Datenvorbereitung – Beispiel

Wir erläutern einige Aspekte dieses Kapitels an einem kleinen Beispiel. Dazu nehmen wir an, wir hätten unsere Daten bereits aus den unterschiedlichsten Quellen gesammelt und diese in einer Datentabelle (Tabelle 12) vereinigt.
Was fällt uns als erstes auf?

- Der Wohnort ist bei allen Personen gleich. Dieses Attribut enthält die Information, dass alle in Wismar wohnen. Dies wird uns aber bei Analysen nicht helfen, denn wir benötigen ja gerade unterschiedliche Werte der Attribute, um Unterschiede oder Ähnlichkeiten zwischen den Objekten herstellen zu können. Dieses Attribut kann folglich gelöscht werden. Wir reduzieren damit die Dimension des Problems.
- Das Geschlecht ist mal als m/w angegeben, mal als male/fem. Das können wir schnell korrigieren.
- Beim Jahresgehalt ist in Datensatz 2 wohl das Gehalt nicht in Tausend angegeben, sondern absolut. Auch das können wir problemlos ändern.
- Im Datensatz 12 fehlen etliche Werte. Diesen sollten wir folglich weglassen.
- Im Datensatz 6 steht bei Betriebszugehörigkeit eine 96. Es gibt 2 Interpretationen: Die 96 könnte als 1996, also Zugehörigkeit seit 1996 interpretiert werden. Es könnte aber auch sein, dass die Betriebszugehörigkeit in Monaten angegeben wurde. Wir entscheiden uns für die *Monate*.

Pers. nr.	Wohn-ort	Ge-schlecht	Alter	Jahres-gehalt	Betriebs-zugehörigkeit	Position	Bildungs-abschluss
1	23966	m	45	32	10	arb	Lehre
2	23966	w	57	35000	25	verw	Bachelor
3	23966	m	52	40	5	manager	Master
4	23966	m	28	27	6	arb	Lehre
5	23966	male	57	45	25	manager	Master
6	23966	fem	26	27	96	arb	Lehre
7	23966	m	39	39	4	manager	Master
8	23966	m	38	32	3	arb	Lehre
9	23966	m	42	31	15	arb	ohne
10	23966	w	37	30	10	verw	Abi
11	23966	m	45	32	8	arb	
12	23966	m	37		5		
13	23966	w	35	30	15	verw	Abi

Tabelle 12: Ausgangsdaten

- In Datensatz 11 fehlt der Bildungsabschluss. Das können wir z.B. korrigieren, indem wir dort den häufigsten Wert einsetzen, der für die gleiche Position (arb) vorkommt: Lehre.
  Aber Vorsicht: Mit dieser Aktion interpretieren wir schon etwas in die Daten hinein, und zwar einen möglichen Zusammenhang zwischen Bildungsabschluss und Position. Man könnte alternativ ein Verfahren wie *k nearest neighbour* (vgl. Abschnitt 5.1.1 auf Seite 33) einsetzen, um so über den/die nächsten Nachbarn einen Wert für den Bildungsabschluss zu finden.

Die bereinigten Daten sind in Tabelle 13 dargestellt.

Pers. nr.	Ge-schlecht	Alter	Jahres-gehalt	Betriebs-zugehörigkeit	Position	Bildungs-abschluss
1	m	45	32	10	arb	Lehre
2	w	57	35	25	verw	Bachelor
3	m	52	40	5	manager	Master
4	m	28	27	6	arb	Lehre
5	m	57	45	25	manager	Master
6	w	26	27	8	arb	Lehre
7	m	39	39	4	manager	Master
8	m	38	32	3	arb	Lehre
9	m	42	31	15	arb	ohne
10	w	37	30	10	verw	Abi
11	m	45	32	8	arb	Lehre
13	w	35	30	15	verw	Abi

Tabelle 13: Bereinigung der Daten

Nun müssen wir uns überlegen, was das Ziel unserer Analyse sein soll. Zunächst gehen wir davon aus, dass wir die Daten clustern wollen. Offensichtlich wird die Personalnummer keine Information enthalten. Diese können wir also ersatzlos streichen. Für das Clustern sind metrische Werte günstig, da man mit diesen besser rechnen kann und folglich bessere Abstandsmaße bekommt. Wir können die nominalen Attribute einfach durch Integer-Werte kodieren (Tabelle 14 auf der nächsten Seite). Wenn wir mit dieser Tabelle clustern (KNIME, 3 Cluster), dann erhalten wir die in Abbildung 30 auf der nächsten Seite dargestellten Cluster.
Auffällig ist, dass das Alter offensichtlich ein große Rolle spielt. Woran liegt das? Nun, der Abstand zwischen einer 26jährigen und einer 57jährigen Person ist so gravierend (31), dass die Unterschiede in den anderen Attributen im Prinzip keine Rolle spielen. Wie können wir das verhindern? Wir

Geschlecht	Alter	Jahresgehalt	Betriebszugehörigkeit	Position	Bildungsabschluss
0	45	32	10	0	1
1	57	35	25	1	3
0	52	40	5	2	4
0	28	27	6	0	1
0	57	45	25	2	4
1	26	27	8	0	1
0	39	39	4	2	4
0	38	32	3	0	1
0	42	31	15	0	0
1	37	30	10	1	2
0	45	32	8	0	1
1	35	30	15	1	2

Tabelle 14: Codierung Variante 1

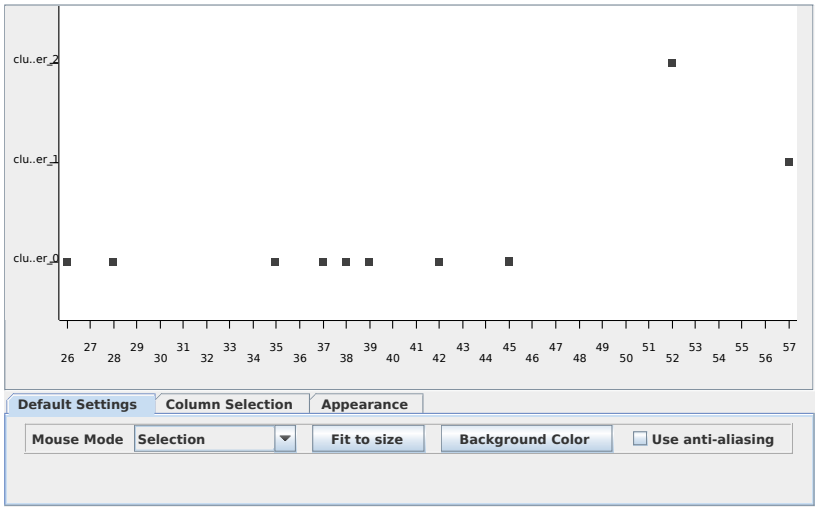


Abbildung 30: Clusterversuch 1

normalisieren alle Daten auf das Intervall [0,1] (Tabelle 15).

Geschlecht	Alter	Jahresgehalt	Betriebszugehörigkeit	Position	Bildungsabschluss
0	0,61	0,28	0,32	0	0,25
1	1	0,44	1	0,5	0,75
0	0,84	0,72	0,09	1	1
0	0,06	0	0,14	0	0,25
0	1	1	1	1	1
1	0	0	0,23	0	0,25
0	0,42	0,67	0,05	1	1
0	0,39	0,28	0	0	0,25
0	0,52	0,22	0,55	0	0
1	0,35	0,17	0,32	0,5	0,5
0	0,61	0,28	0,23	0	0,25
1	0,29	0,17	0,55	0,5	0,5

Tabelle 15: Codierung Variante 2

Nun erhalten wir die in Abbildung 31 dargestellten Cluster (wieder nur bezüglich des Alters).

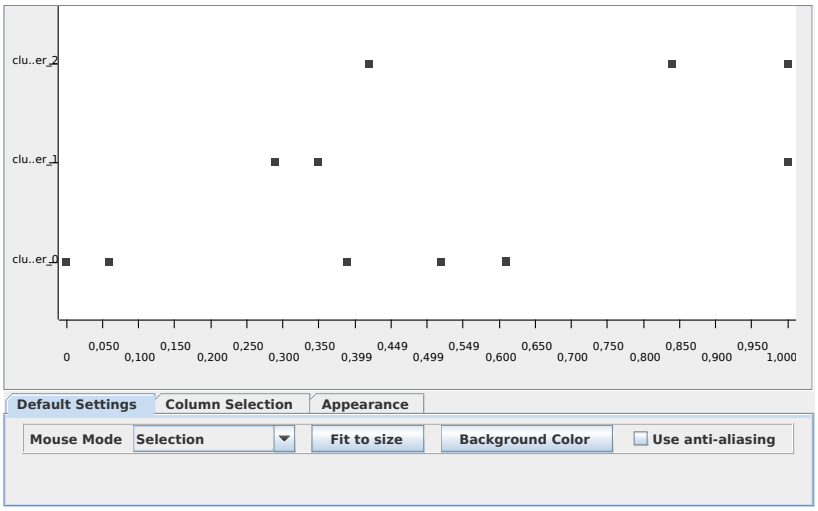


Abbildung 31: Clusterversuch 2 – Alter

Dass die jetzige Clusterbildung sinnvoll ist, zeigt die Darstellung bezüglich des Bildungsabschlusses (Abbildung 32 auf der nächsten Seite).
Zurück zu den Rohdaten. Nehmen wir an, wir wollen einen Entscheidungsbaum für diese Daten entwickeln, und zwar für das Zielattribut *Jahresgehalt*. Jetzt brauchen wir keine metrischen Attribute, sondern nominale/ordinale. Wir müssen also in unserer Ausgangstabelle die metrischen Attribute in ordinale oder nominale umwandeln. Dies kann man so bewerkstelligen, dass man auf allen Werten *eines* Attributs Clustering durchführt. Man erhält so zwei Intervalle, die man entsprechend benennt (Tabelle 16 auf der nächsten Seite).
Mit diesen Daten kann man nun z.B. den ID3-Algorithmus anwenden und erhält mit dem Zielattribut *Gehalt* den in Abbildung 33 auf der nächsten Seite dargestellten Entscheidungsbaum.

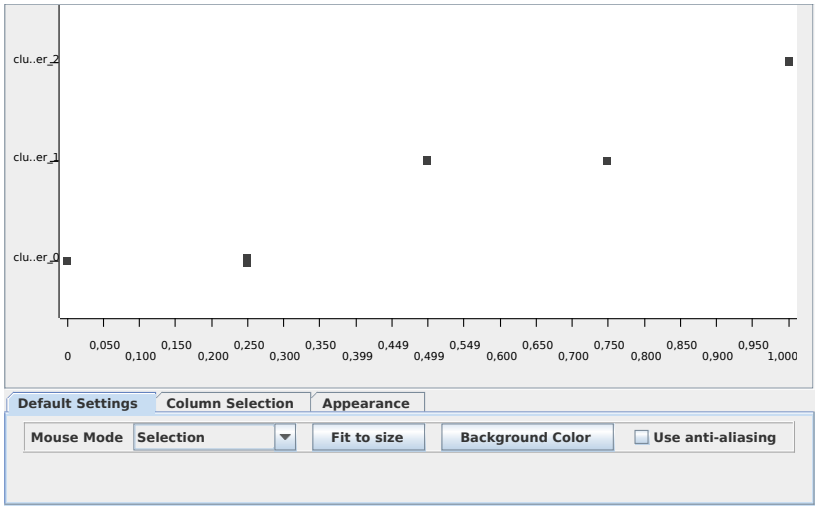


Abbildung 32: Clusterversuch 2 – Bildungsabschluss

Geschlecht	Alter	Jahresgehalt	Betriebszugehörigkeit	Position	Bildungsabschluss
m	alt	gering	kurz	arb	Lehre
w	alt	viel	lang	verw	Bachelor
m	alt	viel	kurz	manager	Master
m	jung	gering	kurz	arb	Lehre
m	alt	viel	lang	manager	Master
w	jung	gering	kurz	arb	Lehre
m	jung	viel	kurz	manager	Master
m	jung	gering	kurz	arb	Lehre
m	alt	gering	lang	arb	ohne
w	jung	gering	kurz	verw	Abi
m	alt	gering	kurz	arb	Lehre
w	jung	gering	lang	verw	Abi

Tabelle 16: Codierung Variante 3

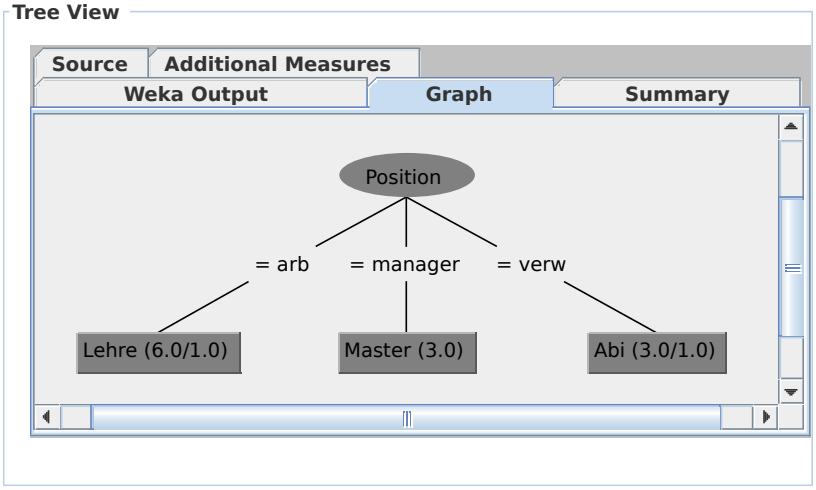


Abbildung 33: Entscheidungsbaum mit ID3

7 Bewertung

Negative expectations yield negative results. Positive expectations yield negative results.
Nonreciprocal Laws of expectations.

Man vergleiche hierzu auch Abschnitt 1.3.5 auf Seite 10.
In großen Datenmengen kann sich Wissen in verschiedensten Strukturtypen darstellen. Jedes Verfahren des Data Minings sucht nur nach einem spezifischen Strukturtyp. So kann es dazu kommen, dass eine Information nicht gefunden wird, die zwar vorhanden, nicht aber in dieser speziellen Struktur dargestellt ist. Um abschätzen zu können, welches Verfahren im konkreten Anwendungsfall die besseren Ergebnisse liefert, empfiehlt es sich, mehrere relevante Verfahren zu testen und zu vergleichen.
Zunächst ist anzumerken, dass ein Verfahren bzw. ein errechnetes Modell nur dann bewertet werden kann, wenn das erwartete Ergebnis zum Vergleich vorliegt oder anderweitig die Qualität der Lösung bewertet werden kann. So kann beispielsweise bei Clustering-Verfahren nur schwer eine Entscheidung getroffen werden, ob ein Verfahren bzw. eine Clusterbildung gut oder schlecht ist. (Es sei denn wir untersuchen Cluster-Verfahren, ob sie bei gegebenen Beispielaufgaben die von uns gewünschten Cluster erzeugen.) Geht man aber von realen Aufgaben aus (wo die gewünschten Zielcluster eben nicht bekannt sind), so werden zwei unterschiedliche Verfahren mit denselben Daten eventuell zu unterschiedlichen Clustern führen. Und dann kann man keine sichere mathematische Aussage darüber treffen, welches Ergebnis nun das bessere ist. Daher gehen wir insbesondere auf die Bewertung von Verfahren zur Klassifikation und Vorhersage ein.
Auf einen wichtigen Grundsatz sollten wir uns vorab verständigen: Werden mehrere Verfahren im Ergebnis als gleich gut bewertet, wird der Vorzug meistens dem Verfahren bzw. dem Modell gegeben, das die gesuchten Zusammenhänge am einfachsten und am leichtesten nachvollziehbar darstellt. Dieses Prinzip nennt man **Prinzip der minimalen Beschreibungslängen**, engl. *MDL – minimum description length*. Die Beschreibungslänge ist definiert als:

- Speicherplatz zur Beschreibung einer Theorie plus
- Speicherplatz zur Beschreibung der Fehler der Theorie

Dieses Konzept geht auf William of Ockham, geboren in Ockham (in Surrey, England) ca. 1285 zurück, der postulierte, dass ein Modell desto besser sei, je einfacher es sei (Occam's razor).

7.1 Interessantheitsmaße

Wir werden zunächst auf spezielle Aspekte der Bewertung von *Assoziationsregeln* (vgl. Abschnitt 4.4 auf Seite 26) eingehen.
Das Ergebnis einer Assoziationsanalyse ist oft eine große Menge an Regeln. Die problembezogene Interpretation und Bewertung der Regeln muss durch den Nutzer erfolgen. Dies kann ein ernstes Problem darstellen, da die Anzahl der Regeln oft kaum überschaubar ist und kein generelles Vorgehensmodell mit einheitlichen Bewertungsgrundlagen existiert. Dieses Problem wird auch als *post-mining rule analysis problem* bezeichnet.
Das Ziel einer sinnvollen Interpretation ist es also, die Fülle an Informationen adäquat zu reduzieren:

- Es ist möglich, die Regeln entsprechend bestimmter signifikanter Merkmale zu sortieren, beispielsweise Support oder Konfidenz. Danach lässt sich die Bedeutsamkeit einer Regeln besser einschätzen.
- Die Regelmenge kann gefiltert werden, d.h. Regeln, die bestimmten Kriterien entsprechen, werden ausgeblendet. Beispielsweise könnte man alle Regeln entfernen, deren Support geringer als 20% ist oder die bestimmte Begriffe aus der ursprünglichen Datenmenge enthalten. Nimmt man das Warenkorbbeispiel, so ließe sich nach Regeln filtern, die Kaffee und Milch enthalten.
- Sind die Regeln nicht sehr komplex, so lassen sich grafische Darstellungen heranziehen, die Korrelationen zwischen bestimmten Attributen erkennen lassen.

Wenn kein explizites Analyseziel verfolgt wird, kann das Durchführen dieser Maßnahmen schnell in ein Trial and Error verfallen. Es ist deshalb sinnvoll, sogenannte *Interessantheitsmaße* zur Bewertung von Regeln zu verwenden. Interessantheitsmaße stammen überwiegend aus der deskriptiven Statistik und erlauben eine Bewertung der Analyseergebnisse einer Assoziationsanalyse auf mathematischer Grundlage.

7.1.1 Support

Wie bereits beschrieben (vgl. Abschnitt 4.4.1 auf Seite 27) ist der Support ein Maß für die Häufigkeit, mit der eine bestimmte Regel in der Datenbasis auftritt. Er wird mit reellen Werten zwischen 0 und 1 gemessen und gibt an, welcher Anteil der Gesamtmenge von einer Regel erfasst wird.
Leider ist es nicht möglich, allgemein zu sagen, welche Supportwerte eine Regel aufweisen muss, um weitere Beachtung zu verdienen. Der Support sagt nur aus, welche Tatsachen oft und welche eher selten vorkommen. Gleichwohl können seltene Items auch interessante Zusammenhänge aufweisen.
Der Support einer Regel $A \rightarrow B$ ist definiert durch:

$$\text{supp}(A \rightarrow B) = P(A \cup B).$$

Wir berechnen also die relative Häufigkeit, in wievielen Datensätzen unserer Datenmenge sowohl A als auch B vorkommt.
Der Support lässt sich mittel eines Venn-Diagramms veranschaulichen (Abbildung 34). Der Support ergibt sich aus dem Flächeninhalt der schraffierten Fläche, geteilt durch die Fläche der gesamten Menge.

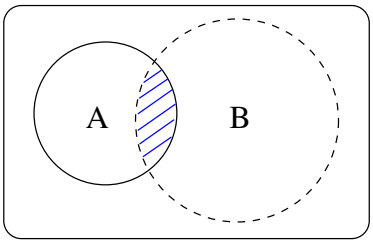


Abbildung 34: Support der Assoziationsregel $A \rightarrow B$

7.1.2 Konfidenz

Die Konfidenz gibt die bedingte Wahrscheinlichkeit eines Items B bei gegebenem Item A an. Sie stellt also die Stärke des Zusammenhangs dar. Analog zum Support wird auch die Konfidenz in Werten zwischen 0 und 1 gemessen.

$$\text{conf}(A \rightarrow B) = P(B|A) = \frac{\text{supp}(A \rightarrow B)}{\text{supp}(A)}$$

Die Konfidenz ergibt sich in Abbildung 35 aus dem Verhältnis der schraffierten Fläche, also aller Fälle, in denen sowohl A als auch B gilt, und der gestrichelt schraffierten Fläche, also A.

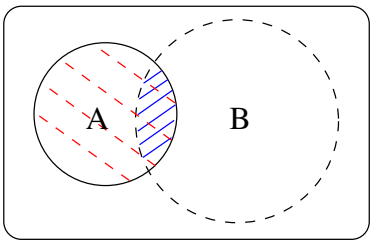


Abbildung 35: Konfidenz der Assoziationsregel $A \rightarrow B$

Eine Schwäche der Konfidenz ist, dass sie fragwürdige Zusammenhänge herstellt. Am Beispiel des Warenkorbs kann man argumentieren: Brot und Milch werden in absoluten Zahlen sehr häufig gekauft. Sie in einen Zusammenhang mit anderen Produkten zu setzen, ist daher wenig sinnvoll. Dennoch könnte man der Konfidenz nach behaupten, dass jeder, der beispielsweise Kaffee kauft, auch Brot kauft. Dieser statistische Zusammenhang ist jedoch sehr fragwürdig.
Es ist also sinnvoll, sich – neben Support und Konfidenz – weitere Interessantheitsmaße anzuschauen. Das obige Kaffee-Brot-Problem bekommt man durch ein weiteres Interessantheitsmaß in den Griff, die Vollständigkeit (completeness).

$$\text{completeness}(A \to B) = P(A \cup B|B)$$

Die Completeness ergibt sich aus dem Verhältnis der schraffierten Fläche, also aller Fälle, in denen sowohl A als auch B gilt, und der gepunktet schraffierten Fläche, also B.

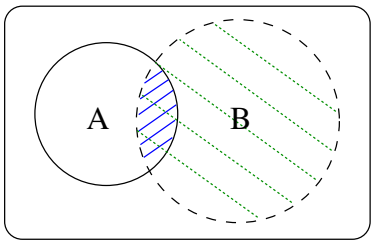


Abbildung 36: Completeness der Assoziationsregel $A \to B$

Beispiel 7.1 (Interessantheismaße für Assoziationsregeln)**.** Seien folgende Werte gegeben:

- $P(A) = 0,50$
- $P(B) = 0,40$
- $P(A \cup B) = 0,35$

Es ergeben sich folgende Werte:

$$\text{supp}(A \to B) = P(A \cup B) = 0,35$$

$$\text{conf}(A \to B) = P(B|A) = \frac{\text{supp}(A \to B)}{\text{supp}(A)} = \frac{0,35}{0,50} = 70\%$$

$$\text{completeness}(A \to B) = P(A \cup B|B) = \frac{\text{supp}(A \to B)}{\text{supp}(B)} = \frac{0,35}{0,40} = \frac{7}{8}$$

Die Konfidenz der Regel ist nicht sonderlich gut, allerdings wird das durch die relativ hohe Completeness ausgeglichen, so dass die Regel doch ein interessanter Kandidat ist.

7.1.3 Gain-Funktion

Es kann vorkommen, dass eine Regel, die knapp am Support gescheitert ist, eine sehr hohe Konfidenz hat. In dieser Situation ist das natürlich ärgerlich, dass wir eine starre Grenze durch den Support definiert haben. Ganz auf den Support zu verzichten geht aber auch nicht, da dann äußert seltene (geringer Support) Regeln akzeptiert würden.
Mit der Gain-Funktion soll dies genau verhindert werden. Es geht also darum, einen Mix aus Support und Konfidenz hinzubekommen, der diese obigen Effekte verhindert.

$$\text{gain}(A \to B) = \text{supp}(A \to B) - \theta \times \text{supp}(A).$$

Ob der Effekt der Reduktion des Interessantheitsmaßes erwünscht ist, hängt vom Ziel der Analyse ab. Der Parameter θ kann Werte zwischen 0 und 1 annehmen, wodurch der Wert der Gain-Funktion stets zwischen $-\theta$ und $1 - \theta$ liegt.
Ein Wert der Gain-Funktion gleich 0 besagt, dass in jedem $\frac{1}{\theta}$-ten Fall, in dem A enthalten ist, auch B vorkommt. Werte über bzw. unter 0 symbolisieren einen stärkeren bzw. schwächeren Zusammenhang.

Bemerkung 7.1 (Gain)**.** Woher kommt der *gain*? Nun, wir fordern, dass eine Regel $A \to B$, die zwar eine hohe Konfidenz hat, andererseits aber nicht so oft vorkommt (der Support von A ist gering), einen geringeren Interessantheitswert als die Konfidenz bekommt. Je geringer der Support von A, desto höher soll die Konfidenz der Regel sein:

$$\text{conf}(A \to B) \geq \frac{\min_{\text{conf}}}{\text{supp}(A)} + \theta$$

Die minimale Konfidenz ist also $\min_{\text{conf}} + \theta$, sie wird mit kleiner werdendem Support von A aber größer. Wir erlauben θ zwischen 0 und 1. Mit $\text{conf}(A \to B) = \frac{\text{supp}(A \to B)}{\text{supp}(A)}$ können wir die Ungleichung mit $\text{supp}(A)$ multiplizieren und erhalten:

$$\text{supp}(A \to B) \geq \min_{\text{conf}} + \text{supp}(A) \times \theta$$

$$\text{supp}(A \to B) - \theta \times \text{supp}(A) \geq \min_{\text{conf}}$$

7.1.4 p-s-Funktion

Mit den Unzulänglichkeiten einiger Interessantheitsmaße für Assoziationsregeln befasste sich Gregory Piatetsky-Shapiro[1].
Er forderte, dass Interessantheitsmaße für Assoziationsregeln (RI) folgende Kriterien erfüllen müssten:

1. $RI(A \to B) = 0$ genau dann, wenn $\text{supp}(A \to B) = \text{P}(A) \times \text{P}(B)$ ist. Das Interessantheitsmaß sollte 0 sein, wenn A und B statistisch unabhängig sind.
2. RI sollte mit $\text{supp}(A \to B)$ monoton wachsen.
3. RI sollte mit $\text{P}(A)$ und $\text{P}(B)$ monoton fallen.

Kriterium 1 befasst sich mit der Situation, dass A und B statistisch unabhängig voneinander sind. Was heißt das? Wie viele korrekte Vorhersagen für B würden wir per Zufall erwarten? Wenn wir dies ohne jede Restriktion tun, dann ist die Wahrscheinlichkeit für das zufällige Auswählen eines Einkaufs, der B erfüllt, gerade $\text{P}(B)$, also die relative Häufigkeit von B. Fordern wir nun zusätzlich, dass auch A erfüllt sein soll, dann sollte sich $\text{P}(A) \times \text{P}(B)$ ergeben. Kriterium 1 fordert also, dass RI 0 sein sollte, falls sich der Support der Regel $A \to B$ genaus so verhält wie bei der zufälligen Auswahl von Itemsets, die A und B erfüllen.
Kriterium 2 fordert, dass RI mit dem Support der Regel steigen sollte. Die Forderung 3, dass das Interessantheitsmaß RI mit der relativen Häufigkeit sowohl von A als auch B monoton fallen sollte, ist ebenso nachvollziehbar.
Die Konfidenz erfüllt nur die Bedingung 2. Insbesondere Kriterium 1 wird häufig verletzt, auch unabhängige Mengen A und B werden meistens mit einer hohen Konfidenz bewertet. Kriterium 3 erfüllt sie ebenso nicht, da die Konfidenz von $\text{P}(B)$ nicht abhängt.
Die p-s-Funktion erfüllt die obigen Restriktionen. Sie ist identisch mit der Gain-Funktion mit $\theta = \text{supp}(B)$, also:

$$ps(A \to B) = \text{supp}(A \to B) - \text{supp}(A) \times \text{supp}(B)$$

Bei dieser Funktion wird davon ausgegangen, dass der Support einer Regel höher sein sollte, als der bei statistischer Unabhängigkeit. Positive Werte spiegeln einen positiven Zusammenhang wider, negative Werte einen negativen Zusammenhang.

[1] Piatetsky-Shapiro, G. (1991). Discovery, analysis and presentation of strong rules. In G. Piatetsky-Shapiro & W. J. Frawley (Eds.), Knowledge discovery in databases (pp. 229-248). Menlo Park: AAAI Press.

7.1.5 Lift

Der Lift gibt an, inwiefern sich die Verteilung eines Elementes in einer Teilmenge von der in der Gesamtmenge unterscheidet. Werte größer als 1 entsprechen einer positiven Korrelation. Somit bedeutet ein Lift von Eins eine Korrelation von Null. Ein Lift von 4 der Regel $A \rightarrow B$ besagt beispielsweise, dass B innerhalb dieser Regel viermal häufiger als in der Gesamtmenge auftritt.

$$\text{lift}(A \rightarrow B) = \frac{\text{supp}(A \rightarrow B)}{\text{supp}(A) \times \text{supp}(B)} = \frac{\text{conf}(A \rightarrow B)}{\text{supp}(B)}$$

7.2 Gütemaße und Fehlerkosten

Wie kann man die Güte einer Vorhersage quantifizieren? Ein einfacher Ansatz ist das Berechnen des Fehlers einer Vorhersage (*Klassifikation* oder *numerische Vorhersage*).

7.2.1 Fehlerraten

Fehlerrate bei Klassifikationen Bei Klassifikationsproblemen ist unter der *Fehlerrate* der relative Anteil der falsch klassifizierten Beispiele einer Instanzenmenge zu verstehen:

$$\text{Fehlerrate} = \frac{\text{Fehler}}{\text{Alle}}$$

Fehlerrate bei numerischer Vorhersage Bei numerischer Vorhersage wird diese absolute Aussage durch den Abstand der vorhergesagten zu den erwarteten Ergebnissen ergänzt:

$$\text{Fehlerrate} = \frac{\sum_i (\text{Realwert}_i - \text{Vorhersagewert}_i)^2}{\sum_i \text{Realwert}_i^2}$$

Es gibt eine Reihe weiterer Maße für die Bewertung numerischer Vorhersagen, s. [WF01, S. 158].

Erfolgsrate Um einen subjektiv besseren Eindruck zu vermitteln, wird häufig die Erfolgsrate statt der Fehlerrate verwendet:

$$\text{Erfolgsrate} = 1 - \text{Fehlerrate}$$

7.2.2 Weitere Gütemaße für Klassifikatoren

Im folgenden werden wir uns weitere Gütemaße speziell für Klassifikatoren anschauen. Diese erweitern das Konzept der Fehlerraten. Wir betrachten dazu einen Klassifikator, der *gute* Kunden vorhersagen soll. Man unterscheidet 4 Fälle:

TP (richtig positiv) Ein guter Kunde wird als guter erkannt.

TN (richtig negativ) Ein nicht guter Kunde wird als nicht guter erkannt.

FP (falsch positiv) Ein nicht guter Kunde wird als guter erkannt.

FN (falsch negativ) Ein guter Kunde wird als nicht guter erkannt.

Darauf aufbauend definiert man eine Reihe von abgeleiteten Kenngrößen [Run10, S. 86ff.].

Korrekte Klassifikationen T=TP+TN

Falsche Klassifikationen F=FP+FN

Relevanz R=TP+FN (Anzahl der guten Kunden)

Irrelevanz I=FP+TN (Anzahl der nicht guten Kunden)

Positivität P=TP+FP (Anzahl der als gut klassifizierten Kunden)

Negativität N=TN+FN (Anzahl der als nicht gut klassifizierten Kunden)

Korrektheitsrate T/n (Anteil der korrekt klassifizierten Kunden)

Inkorrektheitsrate F/n (Anteil der nicht korrekt klassifizierten Kunden)

Richtig-positiv-Rate TPR=TP/R (Wie oft wurde ein guter Kunde auch als solcher klassifiziert.) (**Sensitivität, Trefferquote, Recall**)

Richtig-negativ-Rate TNR=TN/I (Wie oft wurde ein nicht guter Kunde auch als solcher klassifiziert.)

Falsch-positiv-Rate FPR=FP/I (Wie oft wurde ein nicht guter Kunde als guter klassifiziert.)

Falsch-negativ-Rate FNR=FN/R (Wie oft wurde ein guter Kunde als nicht guter klassifiziert.)

Positiver Vorhersagewert, Genauigkeit, Präzision TP/P (Wie oft ist ein als gut vorhergesagter Kunde ein guter Kunde?)

Negativer Vorhersagewert TN/N (Wie oft ist ein als nicht gut vorhergesagter Kunde ein nicht guter Kunde?)

Negative Falschklassifikationsrate FN/N (Wie oft ist ein als nicht gut vorhergesagter Kunde ein guter Kunde?)

Positive Falschklassifikationsrate FP/P (Wie oft ist ein als gut vorhergesagter Kunde ein nicht guter Kunde?)

Meist nimmt man mehrere Kenngrößen zur Bewertung eines Klassifikators, beispielsweise TPR und FPR. In einer *Receiver-Operating-Curve* (ROC-Diagramm) trägt man TPR und FPR gegeneinander auf. Häufig wird auch das PR-Diagramm verwendet, wo man Precision und Recall gegeneinander aufträgt.

7.2.3 Fehlerkosten

Unter betriebswirtschaftlichen Gesichtspunkten ist noch zu beachten, dass sich die aufgrund einer Klassifikation getroffenen Entscheidungen unterschiedlich auf Gewinn bzw. Verlust einer Firma auswirken können. Unter Berücksichtigung dieses Aspekts lassen sich für ein Verfahren auch die sogenannten *Fehlerkosten* berechnen. Wird beispielsweise einem Bankkunden aufgrund der Klassifikation des Verfahrens ein Kredit bewilligt, den dieser nicht zurückzahlt, entstehen zusätzliche Kosten – möglicherweise sogar in Höhe der vollen Kreditsumme. Wird aber jemandem ein Kredit verweigert, der ihn zurückgezahlt hätte, entstehen lediglich Zinseinbußen. Fehler ist also nicht gleich Fehler. Die Kosten eines Fehlers bei einer Entscheidung für den Kredit sind höher als bei einer Entscheidung gegen den Kredit. Wenn nun verschiedene Fehlerarten mit ihren spezifischen Kosten ungleich verteilt auftreten, ist meist nicht das Verfahren mit der minimalen, sondern mit der betriebswirtschaftlich optimalen Fehlerrate gesucht.
Man kann hierzu eine Kostenmatrix oder eine Bonusmatrix aufstellen, die je nach Vorhersagetreffer oder -fehler Plus- bzw. Minuspunkte vergibt.

	Vorhersage	
	0	1
0	10	-20
1	-30	20

Wird ein Datensatz des Typs 0 mit 1 vorhergesagt, so kostet uns das 20, eine korrekte Vorhersage für 0 bringt uns 10. Ziel ist nun natürlich nicht mehr, den Prozentsatz der korrekt vorhergesagten Datensätze, sondern den Gewinn zu maximieren.

7.3 Trainings- und Testmengen

Alle Data-Mining-Verfahren zur Vorhersage beruhen darauf, dass sie sich die ihnen präsentierten Muster in irgendeiner Weise merken, wenn auch nicht explizit jedes einzelne. So sollte jedes Verfahren die ihm häufig genug präsentierten Beispiele auch richtig bearbeiten. Was aber meistens mehr

interessiert, ist die Frage nach dem Verhalten des Verfahrens bei neuen Beispielen. Es ist also sinnvoll, neben den Trainingsdaten auch noch einen Vorrat an Beispielen zu halten, der ausschließlich zum Testen verwendet wird. Nun stehen Daten zwar häufig in großer, aber nicht unbeschränkter Menge zur Verfügung. Im folgenden werden Verfahren vorgestellt, die eine für Trainingszwecke vorliegende Instanzenmenge in Trainings- und Testmengen zerlegen.
Um verschiedene Data-Mining-Verfahren sinnvoll miteinander vergleichen zu können, sollte idealerweise für alle Verfahren dieselbe Zerlegung der Instanzenmenge verwendet werden. So werden beide mit denselben Daten trainiert bzw. getestet. Dies gewährleistet eine gute Vergleichbarkeit der Fehlerraten beider Verfahren.

7.3.1 Holdout

Beim Holdout wird die Instanzenmenge in zwei Teilmengen zerlegt, wobei die eine zum Trainieren und die andere zum Testen verwendet wird.

$$\text{Train} \cup \text{Test} = \text{Instance}$$

7.3.2 Stratifikation

Bei absolut zufälligem Holdout kann das Problem auftreten, dass die Testdaten Beispiele enthalten, zu denen keine ähnlichen in den Trainingsdaten vorkommen – oder andersherum. Dadurch würden die Fehlerraten verfälscht. Mit der Stratifikation wird nun versucht, die Teilung so vorzunehmen, dass sowohl in der Trainings- als auch in der Testmenge die Häufigkeitsverteilung h der Klassen K die gleiche wie in der gesamten Instanzenmenge ist.

$$\forall k \in K : h_k^{\text{Instance}} = h_k^{\text{Train}} = h_k^{\text{Test}}$$

7.3.3 Kreuzvalidierung

Bei der Kreuzvalidierung wird die zur Verfügung stehende Datenmenge in V gleich große Teilmengen zerlegt. Nun wird eine dieser Untermengen als Testmenge zurückbehalten, mit den anderen wird das Verfahren trainiert. Auf der zurückbehaltenen Testmenge wird anschließend die Fehlerrate ermittelt. Dies wird mit jeder anderen Teilmenge als Testmenge wiederholt. Die Fehlerrate des Verfahrens wird als Mittelwert der jeweiligen Fehlerraten berechnet. Wiederholt man die Validierung mehrfach, liefert der Mittelwert der Fehlerraten der einzelnen Validierungen eine genauere Fehlerrate für das Verfahren. Eine Modifikation der Kreuzvalidierung ist das *Bootstrapping*, wo man zulässt, dass sich die Stichproben überschneiden.

$$\text{Fehlerrate} = \frac{\sum_{i=1}^{V} \text{Fehlerrate}_i}{V}$$

7.3.4 Leave-one-out

Das Leave-one-out-Verfahren ist im wesentlichen identisch mit der Kreuzvalidierung. Hierbei wird jedoch die N-elementige Instanzenmenge in genau N Teilmengen zerlegt.

$$\text{Fehlerrate} = \frac{\sum_{i=1}^{|\text{Instance}|} \text{Fehlerrate}_i}{|\text{Instance}|}$$

7.4 Bewertung von Clustern

Eine schwierige Frage im Kontex *Clustering* ist die Bewertung der Qualität einer Clusterbildung. Oder anders formuliert: Wann ist eine Clusterbildung besser als eine andere?
Ansatz 1: Eine erste Forderung ist, dass die einzelnen Cluster möglichst kompakt sind. Dies kann man z.B. durch die Summe der Abweichungen der Objekte eines Clusters vom Clustermittelpunkt (oder deren Quadrate) messen. Diese Abweichungen summiert man nun über alle Cluster. Das

Resultat sollte möglichst minimal sein. Wir gehen hier zunächst davon aus, dass wir eine *feste* Clusteranzahl k vorgegeben haben. Man kann also die Güte eines Clusterings C wie folgt messen:

$$\mathrm{G}_1 = \sum_{i=1}^{k} \sum_{x \in C_i} \mathrm{dist}(x, m_i)^2$$

Hier ist m_i der Mittelpunkt von Cluster C_i. Beachten Sie, dass die Qualität der Clusterbildung umso besser ist, je kleiner der Wert ist. Günstiger ist es also, den Reziprokwert als Gütemaß zu nehmen:

$$\mathrm{Güte}_1 = \frac{1}{\sum_{i=1}^{k} \sum_{x \in C_i} \mathrm{dist}(x, m_i)^2}$$

Ansatz 2: Wir können ebenso fordern, dass die Cluster möglichst weit voneinander entfernt liegen. Man summiert die Quadrate der Distanzen zwischen allen Clustermittelpunkten.

$$\mathrm{Güte}_2 = \sum_{1 \le i \le j \le k} \mathrm{dist}(m_j, m_i)^2$$

Man kann die Ansätze kombinieren, indem man beide Gütemaße multipliziert:

$$\mathrm{Güte} = \mathrm{Güte}_1 \times \mathrm{Güte}_2$$

Beim ersten Gütemaß kann man alternativ auch den maximalen Abstand zweier Punkte in einem Cluster nehmen. Ebenso könnte man zu jedem Punkt die Distanz zu seinem *nächsten* Nachbarn (im gleichen Cluster) nehmen und davon das Maximum über alle Punkte bilden (*minimum distance, single linkage*).
Bisher sind wir von einer *festen* Clusteranzahl k ausgegangen. Ändert sich etwas, wenn wir k variabel lassen? Wir suchen also nach einem beliebigen Clustering, wo aber eben auch die Anzahl der Cluster nicht vorgegeben ist.
Sehen Sie das Problem? Nehmen wir unsere obigen Gütemaße, wäre das Resultat ein Clustering, wo alle Objekte separat einen Cluster bilden. Die Qualität nach Gütemaß 1 wäre dann maximal, da der Nenner 0 ist; besser geht das nicht. Auch nach Gütemaß 2 wäre die Güte maximal.
Wir diskutieren dieses Problem hier nicht weiter. Klar ist aber, dass mit steigender Anzahl der Cluster die Qualität auch künstlich bestraft werden muss.

A Anhang – KNIME

Experience varies directly with equipment ruined.
Horner's Five Thumb Postulate
Arthur Bloch, Murphy's Law

KNIME läuft unter allen Betriebssystemen, erforderlich ist JAVA. Im folgenden soll nur ein kurzer Einstieg in den Umgang mit KNIME gegeben werden.
Der Konstanz Infomation-Miner, kurz KNIME ist ein eigenständiges Projekt einer kostenlosen Data-Mining-Applikation der Universität Konstanz. Es zeichnet sich durch eine sehr einfache Drag&Drop-Bedienung sowie einer großen Anzahl an verschiedenen Algorithmen und Methoden aus. Der Aufbau ist modular und wird ständig um neue Inhalte erweitert, die leicht intern über das Programm geladen werden können. Neben eigenen Algorithmen lässt sich die Software so um eine Vielzahl weitere Inhalte erweitern, so sind z.B. nahezu alle Weka-Verfahren für KNIME umgesetzt worden. KNIME ist in seiner Leistungsfähigkeit durchaus vergleichbar mit einer Vielzahl von kommerziellen Data-Mining-Programmen. Der komplette Miningprozess vom Datenimport über Datenvorverarbeitung und Datenanalyse bis hin zur Darstellung der Ergebnisse lässt sich fast vollständig mit den bereitgestellten Methoden bewerkstelligen.
Nach dem Start von KNIME öffnet sich das in Abb. 37 dargestellte Fenster.

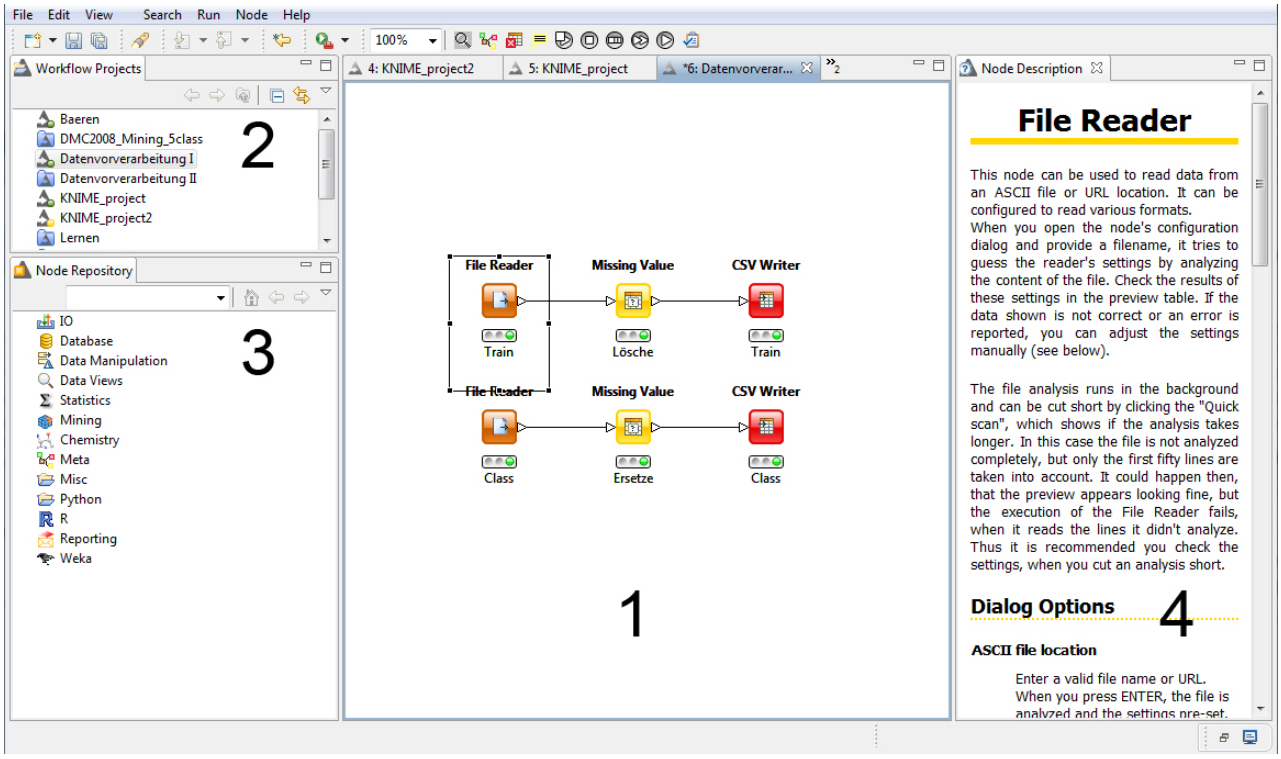


Abbildung 37: KNIME - Start-Fenster

Die Oberfläche enthält folgende Komponenten:

1. Workflow-Fenster
   - primäres Modellierungsfenster des Miningprozesses
   - Verwaltung und Konfiguration der einzelnen Nodes (Rechtsklick auf die jeweilige Node)
   - Verbindungspfeile signalisieren den Datenfluss
2. Projektfenster
   - Verwaltung, Erstellung, Import und Export von Projekten
3. Nodefenster Bereitstellung von Algorithmen und Methoden zum:
   - Datenimport/-export aus Dateien oder Datenbanken

- Datenvorverarbeitung und Datenmanipulation
- Grundlegende statistische Analysen
- Algorithmen und Methoden, getrennt nach KNIME eigene und Weka Methoden

4. Nodebeschreibung

- Detaillierte Beschreibung sämtlicher Nodes z.B.:
- Beschreibung des Algorithmus
- benötigte Inputformatierungen der Daten
- Art des erzeugten Outputs
- kurze Beschreibung der Konfigurationsmöglichkeiten

Abbildung 38 zeigt den Workflow für das Wetterbeispiel.

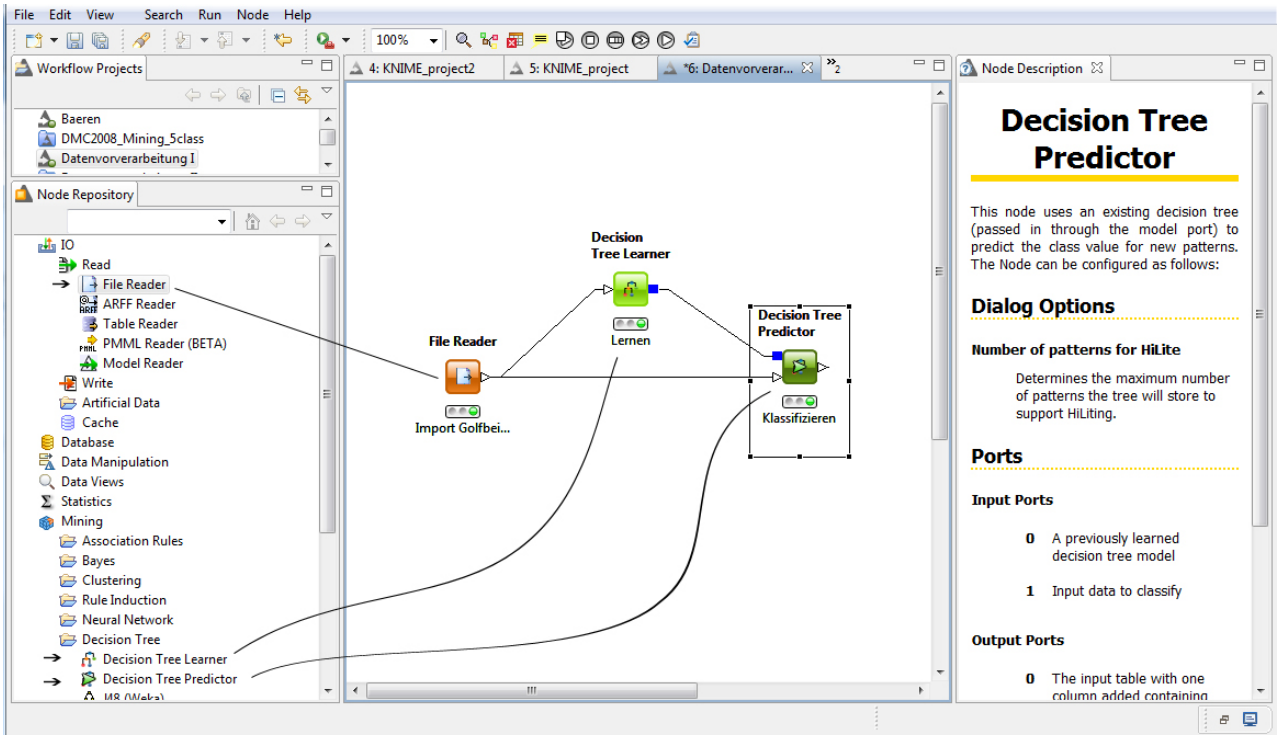


Abbildung 38: KNIME – Wetterbeispiel

Um eine Datei zu importieren, muss in der Nodeauswahl unter IO im Abschnitt *Read* der Filereader ausgewählt und auf die Arbeitsfläche gezogen werden. Gleichzeitig können zusätzlich aus dem Abschnitt *Mining* im Unterverzeichnis *Decision Tree* die Decision-Tree-Learner- und Decision-Tree-Predictor-Nodes in das Arbeitsfenster gezogen werden. Danach werden die Nodes entsprechend der jeweiligen In- und Outputpfade miteinander verbunden. Über einen Rechtsklick auf den Filereader wird im erscheinenden Menü der Eintrag *Configure* ausgewählt. Nun kann der Node konfiguriert werden. Zuerst wird die zu untersuchende Datei ausgewählt, der Filereadernode unterstützt eine Vielzahl von unterschiedlichen Formaten, insbesondere natürlich Dateien im TXT- und CSV-Format. In einigen Fällen müssen zusätzliche Einstellungen vorgenommen werden, um die Datei vollständig und fehlerfrei importieren zu können. Diese Einstellungen können unter *Basic Settings* im Konfigurationsmenü vorgenommen werden. Ist eine Datei ausgewählt, wird das Konfigurationsmenü mit einem Klick auf den OK-Button geschlossen. Bei richtiger Ausführung verfärbt sich die Ampel unter dem Node von rot zu gelb. Mit einem weiteren Rechtsklick auf den Node wird im Menü der Eintrag *Execute* ausgewählt. War der Vorgang fehlerfrei, schaltet die Ampel auf grün.
Jetzt muss der *Decision Tree Learner* konfiguriert werden. Im Konfigurationsmenü wählt man unter dem Eintrag *Class Column* das Zielmerkmal aus, die restlichen Einstellungen können i.allg. unverändert übernommen werden. Abschließend wird der Node ausgeführt.

Um diesen Entscheidungsbaum nun auch testen zu können, muss man vorab die gegebene Beispielmenge in Trainings- und Testmenge zerlegen. Dies macht der *Partitioning*-Node. Mit der Trainingsmenge wird ein Modell erstellt (Abb. 39).

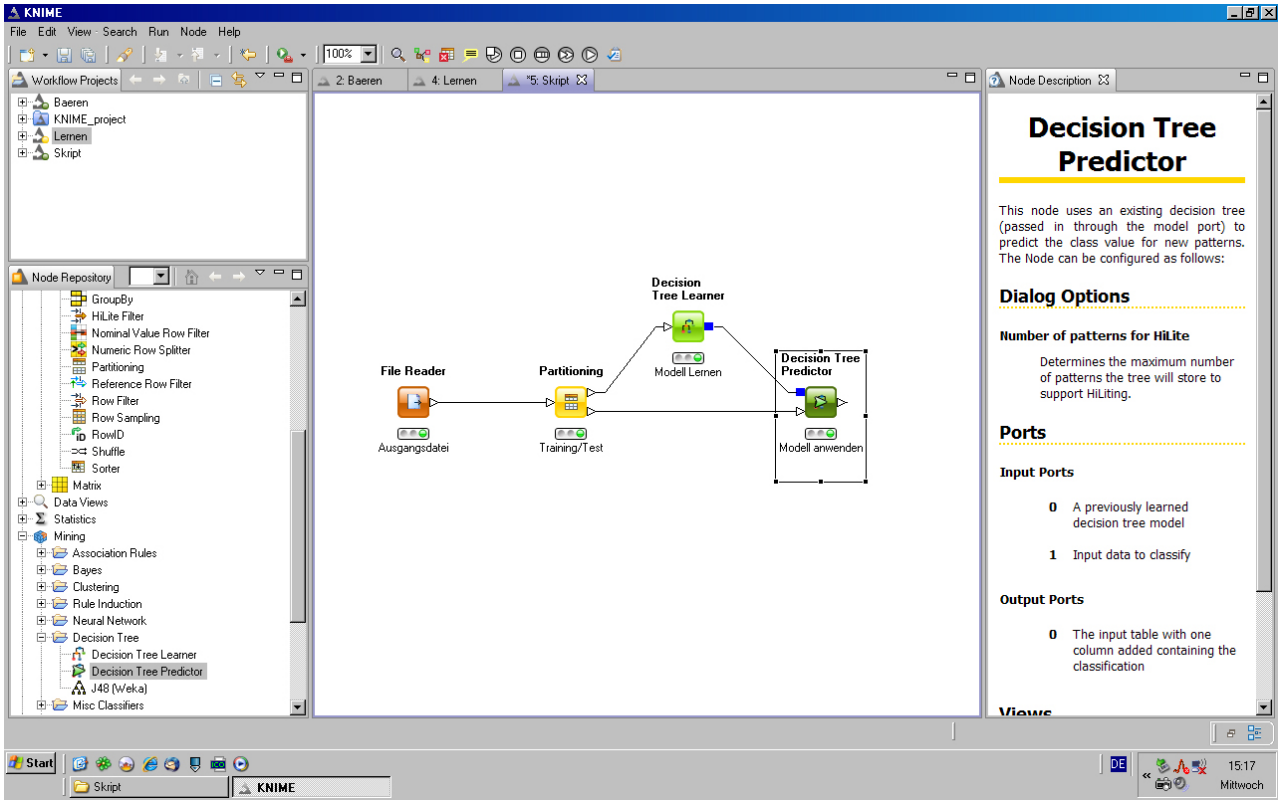


Abbildung 39: KNIME – Trainings- und Testmenge

Nun kann man mit Hilfe des *Decision Tree Predictor* den gelernten Entscheidungsbaum auf die Testmenge anwenden (Rechtsklick, *Classified Data*). Der Predictor fügt eine zusätzlich Spalte mit der vorhergesagten Klassifikation für den jeweiligen Datensatz ein (Abb. 40).

File

Table "data.trn+default" - Rows: 14 | Spec - Columns: 6 | Properties

Row ID	S PlayTennis	S Prediction (DecTree)	S Outlook	S Temperature
D1	no	yes	sunny	hot
D2	no	yes	sunny	hot
D3	yes	yes	overcast	hot
D4	yes	no	rain	mild
D5	yes	no	rain	cool
D6	no	no	rain	cool
D7	yes	yes	overcast	cool
D8	no	yes	sunny	mild
D9	yes	yes	sunny	cool
D10	yes	no	rain	mild
D11	yes	yes	sunny	mild
D12	yes	yes	overcast	mild
D13	yes	yes	overcast	hot
D14	no	no	rain	mild

Abbildung 40: KNIME – Wetterbeispiel

B Anhang – WEKA

When working on a project, if you put away a tool that you are certain you're finished with, you will need it instantly.
Law of Annoyance

Eine Alternative zu KNIME ist WEKA, allerdings ist die Oberfläche nicht so benutzerfreundlich. WEKA ist ein für Hochschulen frei verfügbares Data-Mining-Tool, WEKA steht für *Waikato Environment for Knowledge Analysis*. In WEKA sind viele Algorithmen implementiert, s. hierzu [WFH11]. WEKA läuft unter allen Betriebssystemen, wo Java läuft. Es gibt eine Reihe von WWW-Seiten, die beim Einstieg hilfreich sind. Man findet sowohl WEKA als auch etliche Infos unter http://www.cs.waikato.ac.nz/~ml/weka/index.html. Im folgenden soll nur ein kurzer Einstieg in den Umgang mit WEKA gegeben werden.
Man startet WEKA mit `java -jar weka.jar`. Es öffnet sich das in Abb. 41 dargestellte Fenster.

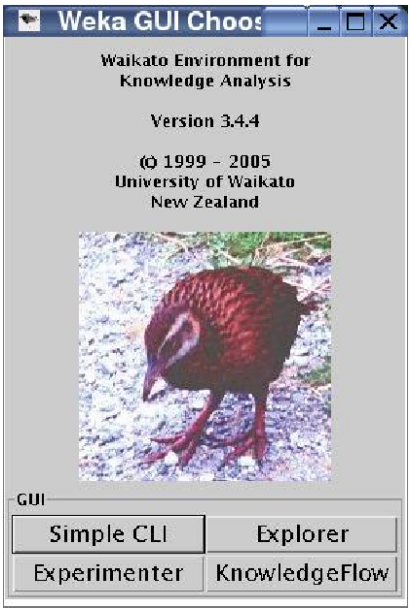


Abbildung 41: WEKA - Start-Fenster

Wir werden mit dem Explorer arbeiten. Es kann im Weka GUI Chooser aber auch die Experimentierumgebung (Experimenter) und ein Kommandozeilenfenster geöffnet werden.
In einem neuen Fenster öffnet sich der Weka Explorer (Abb. 42), welcher eine Art Registertabelle mit den Registern Preprocess, Classify, Cluster, Associate, Select attributes und Visualize enthält. Diese werden nun im einzelnen vorgestellt.
Im Register *Preprocess* können Einstellungen zu den zu verwendenden Datensätzen vorgenommen werden. Datensätze können aus einer Datei (1), über eine URL (2) oder aus einer Datenbank (3) in's Wekasystem geladen werden (Abb. 42).
Dateien werden im ARFF-Format erwartet. ARFF-Dateien bestehen aus einem Kopf und einem Körper. Im Kopf stehen die Schlüsselworte, die mit @ beginnen, wie @relation, @attribute, @data. Es werden nur nominale und numerische Daten erkannt. Bei nominalen Attributen folgt die Wertemenge in geschweiften Klammern ({...}), bei numerischen Attributen folgt das Schlüsselwort `numeric` oder `real`. Zeilen, die mit % beginnen, sind Kommentare. Im Körper stehen die eigentlichen Daten. Ein Datensatz bzw. eine Instanz steht in einer Zeile, die einzelnen Attribute werden durch Kommata getrennt, fehlende Werte werden durch ? ersetzt.

Beispiel B.1.

```
@relation weather.symbolic

@attribute outlook {sunny, overcast, rainy}
```

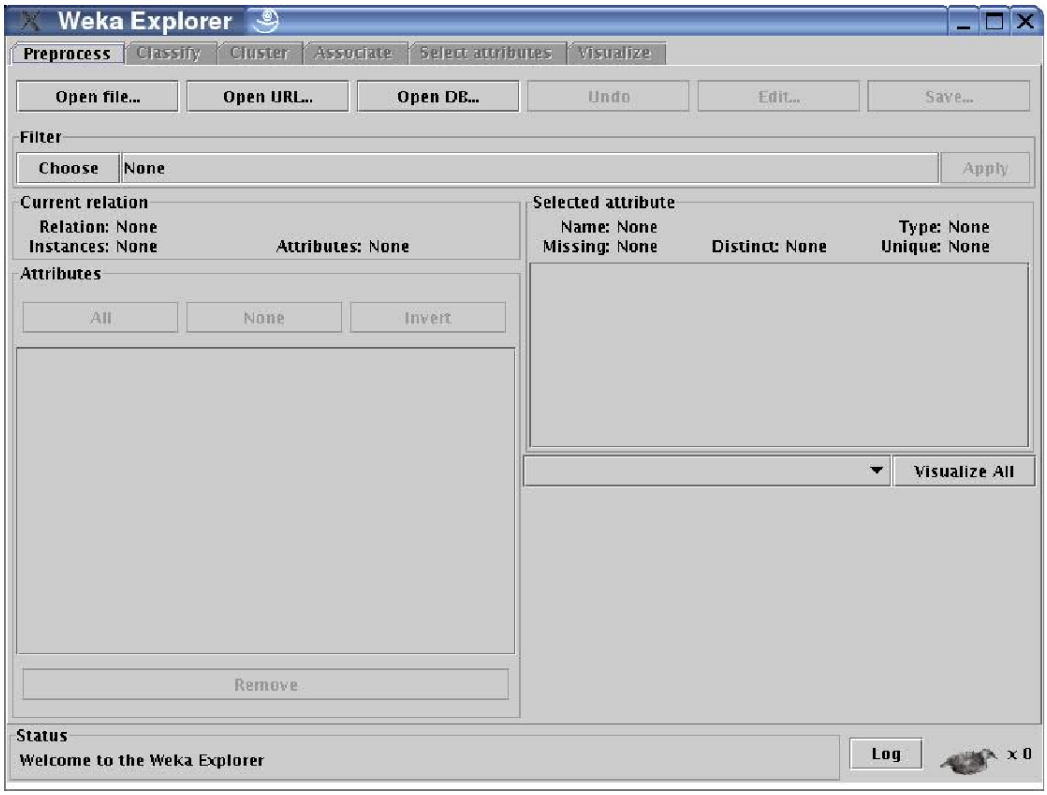


Abbildung 42: WEKA - Explorer

```
@attribute temperature {hot, mild, cool}
@attribute humidity {high, normal}
@attribute windy {TRUE, FALSE}
@attribute play {yes, no}

@data
sunny,hot,high,FALSE,no
sunny,hot,high,TRUE,no
overcast,hot,high,FALSE,yes
rainy,mild,high,FALSE,yes
rainy,cool,normal,FALSE,yes
rainy,cool,normal,TRUE,no
overcast,cool,normal,TRUE,yes
sunny,mild,high,FALSE,no
sunny,cool,normal,FALSE,yes
rainy,mild,normal,FALSE,yes
sunny,mild,normal,TRUE,yes
overcast,mild,high,TRUE,yes
overcast,hot,normal,FALSE,yes
rainy,mild,high,TRUE,no
```

Lädt man diese Datei in's Wekasystem (Abb. 43), erscheinen die Attributnamen. Hier kann ausgewählt werden, welche Attribute im weiteren Verlauf verwendet werden können.
WEKA bietet eine ganze Reihe von Filtern an, mit denen man die Daten vorverarbeiten kann.
Der Programmteil *Classify* dient (überraschenderweise ...) zum Klassifizieren. In WEKA sind etliche Klassifizierer implementiert. Wählt man jetzt z.B. den ID3-Algorithmus (unter Choose/Trees) und startet diesen, so bekommt man im rechten Teilfenster das Resultat der Berechnung (Abb. 44). Man kann auswählen, auf welchen Daten gelernt werden soll:

Use training set Gesamte Datenmenge wird zum Lernen benutzt.

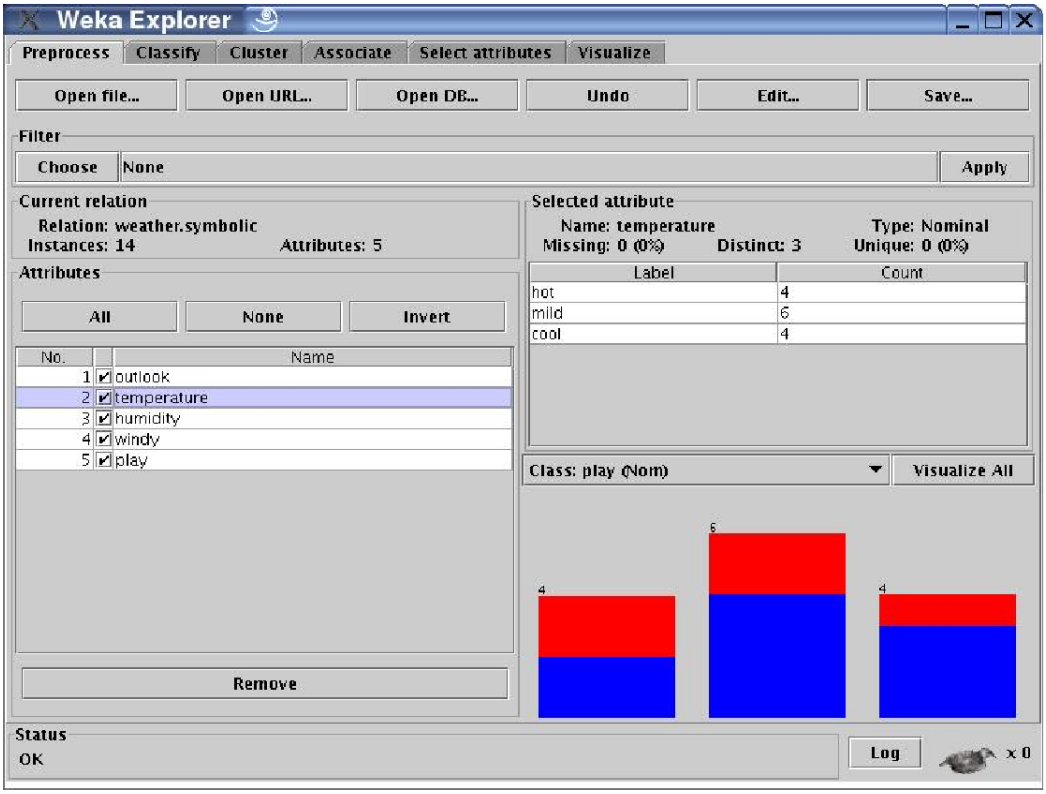


Abbildung 43: WEKA - Preprocessing

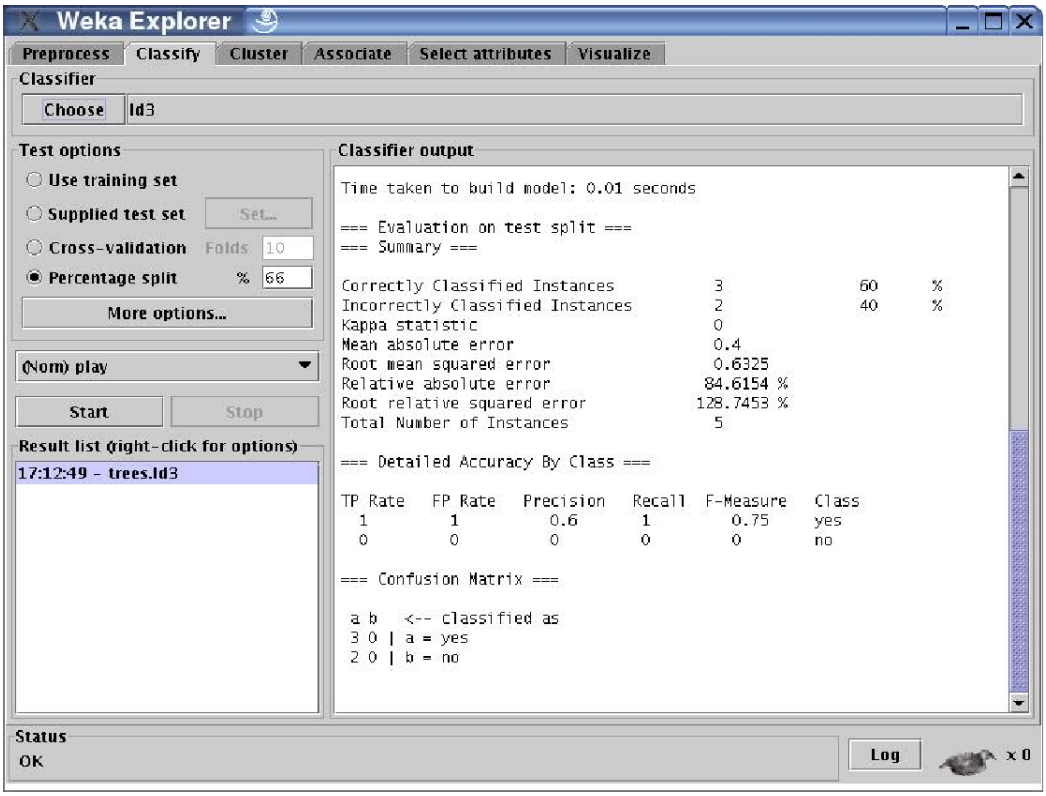


Abbildung 44: WEKA - ID3

Supplied test set Eine separate Datei kann zum Lernen angegeben werden.

Cross validation Gegebene Menge wird mittels Kreuzvalidierung gesplittet (vgl. Abschnitt 7.3.3).

Percentage split Prozentuale Aufteilung der gegebenen Menge in Traings- und Testmenge.

Analog sieht das Vorgehen beim Clustern aus:

- Man wählt einen Algorithmus aus.
- Die Trainingsmenge wird definiert.
- Start.

Ein Hinweis noch: Man kann sich, wenn man im linken unteren Fenster auf einen Test klickt, mit der rechten Maustaste eine Visualisierung generieren lassen.

C Anhang – Entropie

The probability of winning the lottery is slightly greater if you buy a ticket.
Yellin's Law

C.1 Variante 1

Sei E ein Ereignis, das mit der Wahrscheinlichkeit $p(E) > 0$ eintritt. Mit

$$I(E) = \log_2 \frac{1}{p(E)}$$

bezeichnen wir die *Informationseinheiten*, die das Eintreten des Ereignis E enthält. Ist $p(E) = 1$, d.h. wir haben es mit dem sicheren Ereignis zu tun, dann steckt im Eintreten des Ereignisses keinerlei Information, da es ja sicher ist. Es gilt $I(E) = 0$, was dies korrekt widerspiegelt. Wann erreichen wir $I(E) = 1$? Wenn $p(E) = \frac{1}{2}$ ist, d.h. wenn wir zwei gleichwahrscheinliche Ereignisse haben. Je unwahrscheinlicher das Ereignis wird, desto größer wird $I(E)$. Auch dies entspricht unserer Intuition:

Je unwahrscheinlicher ein Ereignis, desto größer der Informationsgehalt, dass es eintritt.

Nun betrachten wir ein Zufallsexperiment. n Ereignisse $a_1, a_2, a_3, \ldots, a_n$ sind möglich, und zwar mit den Wahrscheinlichkeiten $p_1, p_2, p_3, \ldots, p_n$. Dabei setzen wir natürlich voraus:

$$\sum_{i=1}^{n} p_i = 1$$

$$p_i \geq 0$$

Falls das Ereignis a_i eintritt, dann erhalten wir $I(a_i) = \log_2 \frac{1}{p_i}$ Bits an Informationen. Das Ereignis tritt mit einer Wahrscheinlichkeit von p_i ein. Folglich ist die mittlere Informationsgröße, die man erhält:

$$\sum_{i=1}^{n} p_i I(a_i) = \sum_{i=1}^{n} p_i \log_2 \frac{1}{p_i}$$

Auf dieser Basis können wir nun die **Entropie** definieren:

Definition C.1 (Entropie)**.**

$$H_n := H_n(p_1, p_2, \ldots, p_n) = -\sum_{i=1}^{n} p_i \log_2(p_i)$$

Natürlich setzen wir im Fall $p_i = 0$ den Summanden auf 0.
Wir können $I(a_i)$ als diejenige Information interpretieren, die wir benötigen, um zu spezifizieren, dass a_i eintritt.
H_n kann man als mittlere Informationsgröße pro Ausgang des Zufallsexperiments auffassen. Man kann H aber auch als mittlere Unsicherheit interpretieren, die ein Zuschauer *vor* dem Experiment hat.

C.2 Variante 2

Die 2. Variante der Veranschaulichung des Entropiebegriffs stammt aus der Kryptografie und entwickelt den Entropiebegriff auf der Basis von Kodierungen ([PM10, S. 233 ff.]).
Ein Bit ist eine binäre Zahl. Man kann mit ihr 2 Informationen darstellen, also 2 Dinge voneinander unterscheiden. Mit 2 Bits kann man 4 Dinge kodieren (00, 01, 10, 11), mit 3 analog 8. Allgemein kann man mit n Bits 2^n unterscheiden.
Umgekehrt können wir also n unterschiedliche Dinge mit $\log_2(n)$ darstellen.
Wenn wir aber die Wahrscheinlichkeiten für das Auftreten eines Ereignisses in Betracht ziehen, dann können wir dies sogar verbessern.
Wir wollen die Elemente der Menge {a, b, c, d, e} unterscheiden und betrachten dafür folgende Binärkodierung:

a	0	b	10	c	110
d	1110	e	1111		

Die Wahrscheinlichkeiten für das Auftreten der Elemente seien: $p(a) = \frac{1}{2}$, $p(b) = \frac{1}{4}$, $p(c) = \frac{1}{8}$, $p(d) = \frac{1}{16}$, $p(e) = \frac{1}{16}$.
Diese Kodierung nutzt also manchmal 1 Bit (a), manchmal 2, 3 oder 4 (d und e).
Im Mittel nutzt diese Kodierung – jetzt müssen wir die Wahrscheinlichkeiten für das Auftreten des jeweiligen Elements berücksichtigen – diese Bit-Anzahl:

$$p(a) \times 1 + p(b) \times 2 + p(c) \times 3 + p(d) \times 4 + p(e) \times 4 = \\ \frac{1}{2} + \frac{2}{4} + \frac{3}{8} + \frac{4}{16} + \frac{4}{16} = \frac{15}{8} = 1,875$$

Und dies ist natürlich besser als eine Kodierung mittels 3 Bit, die man gemäß aufgerundetem Logarithmus $\log_2(5)$ bräuchte.
Mit dieser Kodierung brauchen wir also 1 Bit, um a von den anderen zu unterscheiden. Das ist $-\log_2(p(a)) = 1$. Wir brauchen 4 Bits, um d und e von den anderen Elementen zu unterscheiden. Das ist $-\log_2(p(d)) = 4$.
Wir bauen nun eine Kodierung, die für die Kodierung von x genau $-\log_2(p(x))$ Bits (bzw. die nächstgrößere natürliche Zahl) benutzt. Und nun ist klar, dass man für eine Informationsübertragung im Durchschnitt

$$\sum_{i=1}^{n} -p_i \log_2(p_i)$$

Bits braucht. Und dies ist genau die Entropie.
Auf diesem Ansatz basiert die Huffman-Kodierung, die in der Kryptografie benutzt wird.

Abbildungsverzeichnis

Tabellenverzeichnis

Symbolverzeichnis

Verzeichnis der Abkürzungen

Verzeichnis der verwendeten Begriffe

Literatur

[BL00] Berry, Michael J. ; Linoff, Gordon S.: *Mastering Data Mining.* Wiley, 2000

[CL14] Cleve, Jürgen ; Lämmel, Uwe: *Data Mining.* 1. Auflage. Oldenbourg, 2014

[DMC] *Data Mining Cup.* www.data-mining-cup.de

[FPSS96] Fayyad, Usama M. ; Piatetsky-Shapiro, Gregory ; Smyth, Padhraic: From Data Mining to Knowledge Discovery: An Overview. In: Fayyad, Usama M. (Hrsg.) ; Piatetsky-Shapiro, Gregory (Hrsg.) ; Smyth, Padhraic (Hrsg.) ; Uthurusamy, Ramasamy (Hrsg.): *Advances in Knowledge Discovery and Data Mining.* Menlo Park, Cambridge, London : MIT Press, 1996, S. 1–34

[HK01] Han, Jiawei ; Kamber, Micheline: *Data Mining: Concepts and Techniques.* Morgan Kaufmann Publ., 2001

[KMU06] Kemper, Hans-Georg ; Mehanna, Walid ; Unger, Carsten: *Business Intelligence.* 2. Auflage. Vieweg Verlag, 2006

[LB01] Linoff, Gordon S. ; Berry, Michael J. A.: *Mining the Web.* Wiley, 2001

[LC12] Lämmel, Uwe ; Cleve, Jürgen: *Künstliche Intelligenz.* 4. Auflage. München, Wien : Hanser, 2012

[Lus99] Lusti, Markus: *Data Warehousing und Data Mining.* Springer, 1999

[ML00] Maimon, Oded ; Last, Mark: *Knowledge Discovery and Data Mining.* Kluwer Academic Publ., 2000

[ML13] Müller, Roland M. ; Lenz, Hans-Joachim: *Business Intelligence.* Heidelberg : Springer, 2013

[Pet05] Petersohn, Helge: *Data Mining.* Oldenbourg, 2005

[PM10] Poole, David ; Mackworth, Alan: *Artificial Intelligence.* Oxford University Press, 2010

[Pyl99] Pyle, Dorian: *Data Preparation for Data Mining.* Morgan Kaufmann, 1999

[Run00] Runkler, Thomas A.: *Information Mining.* Vieweg, 2000

[Run10] Runkler, Thomas A.: *Data Mining.* Vieweg, 2010

[TSDK11] Turban, Efraim ; Sharda, Ramesh ; Delen, Dursun ; King, David: *Business Intelligence.* 2nd edition. Pearson Education, 2011

[WF01] Witten, Ian H. ; Frank, Eibe: *Data Mining.* Hanser Verlag, 2001

[WFH11] Witten, Ian H. ; Frank, Eibe ; Hall, Mark A.: *Data Mining.* 3. Auflage. Morgan Kaufmann, 2011

Index