Optimierung des Demographic Clustering Algorithmus

Silke Wagner

Diplomarbeit

bei Prof. Dr. D. Wagner
Fachbereich für Informatik und Informationswissenschaften
Universität Konstanz

und Dr. C. Lingenfelder
SWSD Data Mining Development
IBM Deutschland Entwicklung GmbH

Konstanz, im Februar 2003

Inhaltsverzeichnis

Einleitung

Informationen sind in der heutigen Gesellschaft wichtiger denn je. Sowohl für Privatpersonen als auch für Unternehmen bedeuten sie oft den entscheidenden Wissensvorsprung. In Unternehmen und anderen Organisationen, die im Lauf der Jahre große Datenmengen gesammelt haben, erkannte man das Potential, das in diesen Daten steckt: es ist nicht nur eine Sammlung von Daten, die man speichert, um beispielsweise mit einem Kunden Kontakt aufnehmen zu können, sondern es ist auch eine Datenbank, in der wichtige Informationen, zum Beispiel über das Kaufverhalten der Kunden, enthalten sind. Natürlich sind diese Informationen nicht in einem einzelnen Tabelleneintrag zu finden, man muss sie vielmehr mit Hilfe verschiedener Verfahren aus den „Rohdaten“ extrahieren. Diese Verfahren fasst man unter dem Begriff *Data Mining* zusammen. Inzwischen gibt es viele Software-Pakete, die dem Benutzer diese Art der Informationsgewinnung erleichtern sollen. Eines davon ist der von IBM Deutschland Entwicklung GmbH in Böblingen entwickelte *IBM DB2 Intelligent Miner for Data*. Jedoch bedeutet Data Mining nicht, dass man ein bestimmtes Programm mit allen zur Verfügung stehenden Daten „füttert“, und als Ausgabe neue Erkenntnisse über die Daten erhält. Die Bearbeitung der Daten durch einen Algorithmus ist vielmehr nur ein Teilschritt eines Prozesses, dessen andere Schritte (wie beispielsweise die Vorverarbeitung der Daten oder die Wahl des Verfahrens) für das Gelingen ebenso wichtig sind. Im ersten Kapitel dieser Diplomarbeit wird der Data Mining-Prozess mit seinen einzelnen Schritten ausführlich beschrieben. Des weiteren werden häufig verwendete Data Mining-Verfahren vorgestellt. Im zweiten Kapitel wird dann ein wichtiges Verfahren genauer betrachtet: die Cluster-Analyse. Zunächst wird an einem Beispiel die Anwendung der Cluster-Analyse erläutert. Nach der Erklärung der grundlegenden Begriffe wird die Funktionsweise einzelner Algorithmen beschrieben.
Gegenstand des zweiten Teils dieser Diplomarbeit ist der *Demographic Clustering Algorithmus*, ein im *Intelligent Miner for Data* implementierter Cluster-Analyse-Algorithmus. Er wird im dritten Kapitel genau analysiert. Des weiteren wird erläutert, wie man durch Verwendung einer speziellen Datenstruktur eine lineare Laufzeit erzielen kann. Dies ist von großer Bedeutung, denn ohne diese Struktur wäre die Laufzeit quadratisch in der Anzahl der Datensätze, was den Algorithmus für die praktische Anwendung unbrauchbar machen würde. Im Rahmen dieser Diplomarbeit wurden verschiedene Ansätze zur weiteren Laufzeitoptimierung erarbeitet (und implementiert), welche im vierten Kapitel vorgestellt werden. Abschließend wird anhand verschiedener Testergebnisse diskutiert, mit welcher Laufzeit-Ersparnis zu rechnen ist und von welchen Parametern sie abhängt.

Kapitel 1

Data Mining

1.1 Definition

Bevor das Thema Data Mining mit seinen einzelnen Aspekten ausführlicher diskutiert wird, soll an dieser Stelle zunächst einmal geklärt werden, was denn genau unter Data Mining zu verstehen ist.
Gemeinhin erwartet man, auf der Suche nach einer solchen Definition in der Fachliteratur eine gemeingültige „offizielle“ Definition zu finden. Dies ist jedoch nicht der Fall: Man findet viele verschiedene Definitionen, die in der Kernaussage zwar ähnlich sind, den Schwerpunkt des Data Mining aber auf verschiedene Bereiche legen. Des weiteren unterscheiden sie sich dadurch, dass manche mit Data Mining einen ganzen Prozess bezeichnen (siehe Kap. 1.3), wohingegen andere Definitionen nur einen Teilschritt dieses Prozesses beschreiben. Im Allgemeinen wird jedoch auf den gesamten Prozess Bezug genommen. Einige typische Definitionen, wie man sie in leicht abgewandelter Version häufig findet, sind die folgenden:

> *Unter Data Mining versteht man die Sammlung von Methoden und Verfahren, mit denen man sehr große Datenmengen analysieren und in diesen Daten a priori unbekannte Strukturen und Beziehungen entdecken kann. Hierzu werden die Daten vorverarbeitet und gereinigt, so dass sie dazu geeignet sind, als nützliches Hilfsmittel für die Herleitung von Entscheidungen und Strategien zu dienen.*
>
> (J. Grabmaier, A. Rudolph)

> *Data Mining is the nontrivial extraction of implicit, previously unknown, and potentially useful information from data. This encompasses a number of different technical approaches, such as clustering, data summarization, learning classification rules, finding dependency networks, analyzing changes and detecting anomalies.*
>
> (W. J. Frawley, G. Piatetsky-Shapiro, C. J. Matheus)

> *Data Mining is the process of extracting previously unknown, valid, and actionable information from large databases and then using the information to make crucial business decisions.*
>
> (P. Cabena, P.O. Hadjinian, R. Stadler, J. Verhaas, A.Zanasi)

Bei der Definition von Grabmaier und Rudolph gehört die Vorverarbeitung der Daten und die Auswertung der Ergebnisse nicht zum eigentlichen Data Mining. Auch bei Frawley, Piatetsky-Shapiro und Matheus werden diese Schritte nicht erwähnt, hier liegt der Schwerpunkt auf den verschiedenen Verfahren, die beim Data Mining eingesetzt werden können. Auch wenn bei vielen der Definitionen Data Mining nicht explizit als ein aus mehreren Schritten bestehender Prozess beschrieben wird, so wird oft bei der weiteren Lektüre des entsprechenden Buches bzw. Aufsatzes deutlich, dass dies implizit gemeint ist. Die Existenz dieser unterschiedlichen Interpretationen (Prozess vs. Prozess-Schritt) hat ihre Ursache wohl unter anderem darin, dass in den 80er Jahren ein neuer Begriff kreiert wurde, *Knowledge Discovery in Databases (KDD)*, der alle alten Ausdrücke ersetzen sollte, die Verfahren zum Finden von Mustern und Ähnlichkeiten in Rohdaten bezeichneten. Vor allem im Bereich der künstlichen Intelligenz und des maschinellen Lernens wurde KDD schnell übernommen und bezeichnete fortan den gesamten Prozess des Extrahierens von Wissen aus Datenbanken, von der Festlegung der Ziele bis zur Analyse der Ergebnisse. In diesem Kontext bezeichnete *Data Mining* den Schritt im Prozess, in dem die Mining-Algorithmen angewendet werden. Diese Interpretation wurde dann auch auf der ersten internationalen KDD-Konferenz, die 1995 in Montreal stattfand, formalisiert [CHS+98]. Das wachsende Interesse an diesem Gebiet führte aber dazu, dass der Begriff *Data Mining* in der Presse und von den Verkäufern entfremdet wurde und als Ausdruck für den gesamten Prozess verwendet wurde. Inzwischen werden die beiden Begriffe meist als Synonyme betrachtet, und darum soll auch hier im weiteren Verlauf mit *Data Mining* der gesamte Prozess gemäss der Beschreibung in Kapitel 1.3 bezeichnet werden.

Nun könnte man meinen, dass sich Data Mining nicht allzu sehr von herkömmlichen Datenanalyse-Methoden unterscheidet. Vor allem als das Data Mining noch in den Kinderschuhen steckte, behaupteten Kritiker, dass man lediglich bereits bekannte Verfahren unter einem neuen Namen verkaufen würde. Dieser Verdacht ist sehr nahe liegend, da viele der traditionellen Datenanalyse-Verfahren – einschließlich Statistik – beim Data Mining mit eingehen. Ein weiterer Grund für diese nach wie vor bestehende Kritik liegt wohl darin, dass manche Software-Verkäufer das wachsende Interesse an Data Mining ausnutzen um ihre Produkte mit Hilfe dieses Schlagwortes am Markt zu positionieren [CHS+98]. Data Mining wird oft mit einem der folgenden Gebiete verwechselt:

- Datenabfragen auf Data Warehouses[1]

[1] Ein Data Warehouse führt Daten aus unterschiedlichen Quellsystemen zusammen und ermöglicht durch spezielle Maßnahmen der Datenstrukturierung und des Performancetunings übergreifende und/oder detaillierte Abfragen. Ziel eines Data Warehouse ist es, Daten in nutzbare Informationen umzuwandeln und für das Management und Analysten bereitzustellen. (Markus Bohner, *Data Warehouse - Begriffe und Fakten,* 1998, www.isb-ka.de/zeitung/01-98/artikel2.htm)

- Abfragen auf einer beliebigen Zahl ungleicher Datenbanken
- Abfragen in einer parallelen Umgebung
- Multidimensionale Datenanalyse (MDA)
- Online Analytical Processing (OLAP)
- Exploratory data analysis (EDA)
- moderne Visualisierungstechniken
- traditionelle statistische Methoden

All diese Verfahren sind kein Data Mining, da ihnen eine wesentliche Eigenschaft fehlt: Die *Entdeckung von Informationen ohne eine vorab formulierte Hypothese.* Als Faustregel lässt sich also sagen: wenn man genau den Umriss und mögliche Inhalte dessen kennt, wonach man sucht, hat man es sehr wahrscheinlich nicht mit Data Mining zu tun.
Ein gewisser Zusammenhang zu den oben genannten Verfahren ist jedoch durchaus gegeben. Vor allem mit der traditionellen Statistik ist Data Mining eng verwandt, weswegen die Beziehung der beiden kurz genauer betrachtet werden soll. Viele der Analysen, die heute mit Data Mining gemacht werden, wie z.B. die Erstellung von Vorhersagemodellen oder die Entdeckung von Zusammenhängen in Datenbanken, wurden früher mit Hilfe von Statistik gemacht. Man kann sogar sagen, dass es zu fast jedem der Haupteinsatzgebiete von Data Mining einen entsprechenden statistischen Ansatz gibt [CHS+98]. Der große Unterschied liegt darin, dass die traditionelle Statistik nur bis zu einem gewissen Datenvolumen sinnvoll eingesetzt werden kann. Ist die Datenmenge zu groß, ist es einerseits schwer, sinnvolle Hypothesen zu erstellen, andererseits ist der Aufwand, der zur Verifikation bzw. Falsifikation einer Hypothese benötigt wird so enorm groß, dass das Verfahren viel zu teuer und dadurch unattraktiv wird. Des weiteren ist es in Unternehmen oft wichtig, schnell zu handeln, um der Konkurrenz zuvorzukommen, sodass langwierige Verfahren die gewünschten Ergebnisse viel zu spät liefern. Das ist einer der großen Vorteile des Data Mining, denn von einem Data Mining-Verfahren wird verlangt, dass es skalierbar ist [Mai01].

Definition 1.1.1 (Skalierbarkeit) *Unter* Skalierbarkeit *(im Sinne der Informatik) von Data Mining-Verfahren versteht man die Eigenschaft, dass diese Verfahren eine Laufzeit von $\mathcal{O}(n)$ oder $\mathcal{O}(n \cdot \log n)$ besitzen, wobei $n \in \mathbb{N}$ die Zahl der Datensätze angibt, auf denen das Verfahren operiert.*

Data Mining-Verfahren setzen also da an, wo die klassischen Methoden der Statistik versagen: sie erzielen selbst auf riesigen Datenmengen noch gute Laufzeiten. Neben diesem wohl wichtigsten Unterschied, gibt es aber noch weitere Unterschiede zwischen den Verfahren, die darin bestehen, dass Data Mining-Verfahren

- meist keine Annahmen über die Verteilung der Daten machen, da (fast) alle vorhandenen Daten verwendet werden und somit also die wirkliche Verteilung in das Verfahren einfließt

- oft auf Heuristiken zurückgreifen
- oft keine eindeutigen Ergebnisse liefern, sondern vielmehr Ergebnisse, die der richtigen Interpretation bedürfen (und dann aber sehr wertvoll sein können (siehe Kap. 1.2))
- keine Hypothese über die Daten bestätigen oder verwerfen (*verification driven analysis*), sondern (halb-)automatisch nach Mustern und Strukturen in den Daten suchen (*discovery driven analysis*)
- mit Hilfe moderner Visualisierungswerkzeuge die Ergebnisse leichter interpretierbar machen

Abschließend sei noch erwähnt, dass Statistik natürlich nicht das einzige Verfahren ist, das beim Data Mining eine große Rolle spielt - wenn auch das wichtigste. Data Mining ist vielmehr ein interdisziplinäres Verfahren an der Schnittstelle von Statistik, maschinellem Lernen, künstlicher Intelligenz und Datenbanksystemen.

1.2 Was macht Data Mining so erfolgreich?

Data Mining hat sich zu einem der führenden Managementthemen entwickelt. Vorreiter sind – wie so oft – die amerikanischen Anwender und das in vielen Bereichen des Data Mining: im Retail Banking und im Kreditkartenwesen werden die Data Mining-Verfahren schon seit geraumer Zeit produktiv eingesetzt. Aber das Bankwesen ist nicht die einzige Branche. Anwender stammen aus sehr unterschiedlichen Bereichen, wie z.B. Einzelhandel, Telekommunikation, Versicherungen, Kommunen, soziale Bereiche, Gesundheitswesen und Finanzbehörden. Auch die Ziele, für die Data Mining eingesetzt wird, sind sehr unterschiedlich: Missbrauchserkennung beim Bezug von Sozialleistungen, Fördermitteln, Kreditkarten o.ä., Reduzierung von Risiken bei der Kreditvergabe oder bei der Entscheidung über neue Investitionen, Erschließung der am besten verkäuflichen Produktkombinationen oder Erkennen homogener Kundengruppen sind nur einige der vielen Einsatzgebiete des Data Mining [KWZ98]. In Europa zeichnet sich eine ähnlich erfolgreiche Verbreitung der Data Mining-Tools ab. Doch woher kommt dieser Erfolg?
Der Erfolg begründet sich im glücklichen Zusammenspiel mehrerer Faktoren, die man grob in zwei Bereiche zusammenfassen kann: zum einen die veränderten Marktverhältnisse, welche die Unternehmen zur Suche neuer Wettbewerbsstrategien zwingen und zum anderen die rasante technologische Entwicklung, welche die technische Realisierung der Data Mining-Verfahren überhaupt erst möglich machte. Diese beiden Bereiche sollen im Folgenden näher betrachtet werden.

1.2.1 Veränderte Marktverhältnisse

Unternehmen aller Branchen erfuhren in den letzten Jahren fundamentale Veränderungen der Märkte und der Wettbewerbsbedingungen. Die wichtigsten dieser Veränderungen sind [CHS+98] :

- Verhaltensmuster der Kunden

  Konsumenten sind heutzutage anspruchsvoller, denn sie können sich via Einkaufsführer, Katalog und Internet besser informieren und somit auch besser vergleichen. Viele Konsumenten sind jedoch auch mit den zu großen Auswahlmöglichkeiten überfordert und beschränken von vornherein die Anzahl an Geschäften/Unternehmen, die sie bereit sind zu besuchen.

- Marktsättigung

  Viele Märkte sind gesättigt. Fast jeder hat heutzutage ein Bankkonto, eine Kreditkarte, eine Hausratsversicherung usw. Die meisten haben im Supermarkt relativ feste Einkaufsgewohnheiten, von denen sie oft nur wenig abweichen. Will ein Unternehmen auf einem dieser Gebiete seinen Marktanteil vergrößern, bleiben ihm meist nur wenige Möglichkeiten: er muss Kunden von der Konkurrenz abwerben oder mit den eigenen Kunden mehr Umsatz erzielen. Hierzu ist es natürlich ein großer Vorteil, die Kunden und deren Verhalten genau zu kennen.

- Kürzere Produktlebensdauer

  Die heutigen Produkte werden schnell auf den Markt gebracht, haben oft aber auch eine kurze Lebensdauer, weil sie von neueren Produkten abgelöst werden. Bestes Beispiel hierfür ist der Computermarkt: Fast täglich erscheinen noch schnellere PCs mit noch mehr Speicherplatz, noch schnelleren Laufwerken usw.

Diese Veränderungen führten zusammen mit anderen Faktoren dazu, dass der Wettbewerb härter und risikoreicher wurde, was die Unternehmen dazu brachte, ihre traditionellen Ansätze neu zu bewerten und nach Wegen zu suchen, den veränderten Bedingungen gerecht zu werden. Man erkannte, dass der einzelne Kunde sehr wertvoll ist und es deshalb wichtig ist, ihn seinen Bedürfnissen entsprechend individuell zu betreuen und an das Unternehmen zu binden. Hierzu ist es hilfreich, wenn man Fragen beantworten kann wie

- Welchen Kunden sollte wann welches Angebot unterbreitet werden?
- Bei welchem Kundenprofil lohnt sich ein Außendienstbesuch?
- Welche Kunden sind dem Unternehmen treu, welche nicht?
- Welcher Lifetime-Profit lässt sich mit welchem Kunden erzielen?
- Wie lassen sich „gute“ Kunden mit hohen Lifetime-Values gewinnen?

Des weiteren erhöhte der verschärfte Konkurrenzdruck die Nachfrage nach besseren Prognosemethoden, um die Marktentwicklung abschätzen und schnell handeln zu können.
Aufgrund der technologischen Entwicklung (siehe Kap. 1.2.2) hatten viele Unternehmen inzwischen große Datenbanken aufgebaut, die detaillierte Informationen über Kunden und Interessenten enthalten. Neben der Adresse liegen oftmals soziodemographische Daten, Kaufinformationen, Potenzialdaten sowie Kommunikationsdaten vor. Diese Informationen werden in der Regel genutzt, um direkt mit dem einzelnen

Kunden zu kommunizieren. Man wurde sich darüber im klaren, dass in diesen Datenbanken „verborgene Schätze“, nämlich Antworten auf obige Fragen, liegen [Das00]. Das Problem dabei ist, dass die Antworten nicht in einem einzelnen Datenfeld oder einem Kundenmerkmal, sondern in der richtigen Kombination unterschiedlicher Kundeninformationen zu finden sind. So kann etwa die Angebotsaffinität eines Kunden von einer Vielzahl von Merkmalen wie Alter, Geschlecht, Familienstand, demographischen Typologien, bisher gekauften Produkten, gezeigtem Produktinteresse, Zahlungsmoral und einer Reihe weiterer Eigenschaften abhängen. Mit den traditionellen statistischen Methoden ist es aber nicht oder nur mit einem nicht vertretbaren Aufwand möglich, diese verborgenen Schätze ans Tageslicht zu befördern. Es mussten neue Verfahren her, die selbst auf großen Datenmengen gute Ergebnisse liefern.
Auf Anwenderseite war also eine große Nachfrage nach besseren Analyse- und Prognosetechniken vorhanden.

1.2.2 Technologische Entwicklung

Die Nachfrage nach Data Mining-Verfahren hat zwar erst in den letzten Jahren enorm zugenommen, doch auch wenn dies schon eher der Fall gewesen wäre, hätte sich Data Mining wahrscheinlich trotzdem nicht schneller zu dem entwickelt, was es heute ist. Grund dafür ist die technologische Entwicklung, die erst in den letzten Jahren die technischen Voraussetzungen für effektives und schnelles Data Mining geschaffen hat. Schaut man sich die Entwicklung der Datenhaltung und -nutzung seit den 60er Jahren an, so ist Data Mining eigentlich ein logischer Entwicklungsschritt, der zwangsläufig früher oder später kommen musste, sobald die technischen Voraussetzungen dafür geschaffen waren (siehe Abb. 1.1). Begünstigt wurde die Entwicklung durch die in Kapitel 1.2.1 erläuterte Marktsituation.
Die bisherige Entwicklung der Datenhaltung und -nutzung ist auch insofern wichtig, als dass dadurch in vielen Unternehmen einerseits bereits die nötige datentechnische Infrastruktur (z.B. Data Warehouses) vorhanden war/ist, um Data Mining ohne größeren Aufwand einführen zu können und andererseits im Lauf der Zeit große Datenmengen gesammelt werden konnten, so dass Data Mining überhaupt erst für die Unternehmen interessant wurde.
Auf technologischer Seite verdankt das Data Mining also seinen Erfolg im Wesentlichen folgenden Faktoren:

- Enorme Datenmengen

  Nach 40 Jahren Informationstechnologie haben sich enorme Datenmengen angesammelt, die inzwischen in Giga- und Terabytes[2] gemessen werden. Es wird geschätzt, dass sich die Menge der gesammelten Daten weltweit alle 20 Jahre verdoppelt und dass die Anzahl und Größe der Datenbanken sogar noch schneller wächst [WF01]. Jeder erzeugt täglich viele Einträge in diverse Datenbanken, z.B. beim Geldabheben am Automaten, beim Einkaufen im Supermarkt, beim Surfen im Internet oder auch einfach nur beim Telefonieren. Natürlich werden

[2]Ein Terabyte sind eine Trillion Bytes, das entspricht einer Datenmenge von 1 Milliarde DIN-A4-Seiten, was einem Papierstapel von 100 km Höhe gleichkommt.

Entwicklungsschritt	Geschäftsfragen	verfügbare Technologien	Eigenschaften
Data Collection (60er Jahre)	„Was war der Durchschnitt aller Einkünfte der letzten Jahre?“	Computer, Bänder, Platten	Retrospektivische, statische Datenbeschaffung
Data Access (80er Jahre)	„Was waren im letzten März in New England die Verkaufszahlen pro Einheit?“	Relationale Datenbanken, SQL, ODBC	Retrospektivische, dynamische Datenbeschaffung auf Record-Basis
Data Navigation (90er Jahre)	„Was waren im letzten März in New England die Verkaufszahlen pro Einheit? Verfeinern (drill down) bis Boston.“	multidimensionale Datenbanken, Data Warehouse, OLAP	Retrospektivische, dynamische Datenbeschaffung auf verschiedenen Grundlagen
Data Mining (ab 2000)	„Was für Verkaufszahlen werden wir voraussichtl. nächsten Monat in Boston haben? Warum?“	Erweiterte Algorithmen, Mehrprozessor-Rechner, parallele Datenbanken	Prospektivische, proaktive Informationsbeschaffung

Abbildung 1.1: Entwicklung der Datenhaltung und -nutzung [KWZ98]

viele dieser Daten gesammelt, damit die entsprechende Dienstleistung überhaupt erbracht werden kann (z.B. Reservierungen), doch sie werden mehr und mehr auch dazu verwendet, strategische Entscheidungen für das Unternehmen zu treffen. Hinzu kommt die wachsende Verfügbarkeit demographischer und psychographischer Daten, die von den Unternehmen hinzugekauft werden können. Solche Daten sind sehr wichtig, wenn man Aussagen über das Kundenverhalten machen will, das oft durch Präferenzen und Entscheidungen charakterisiert wird, die nicht aus den unternehmensinternen Datenbanken ersichtlich sind.

- Wachsende Verbreitung von Data Warehouses

  Viele Unternehmen haben für eine bessere Nutzung und Konsistenz ihrer Daten bereits Data Warehouses eingeführt. Dies verringert den Aufwand bei der Einführung von Data Mining, da Data Warehouses (in der Regel) saubere und gut dokumentierte Datenbanken enthalten, was die Vorverarbeitung der Daten im Data Mining-Prozess (siehe Kap. 1.3) enorm erleichtert.

- Neue IT-Lösungen

  Kostengünstigere und leistungsstärkere IT-Lösungen im Hinblick auf Speicherplatz und Prozessorleistung ermöglichen groß angelegte Data Mining-Projekte

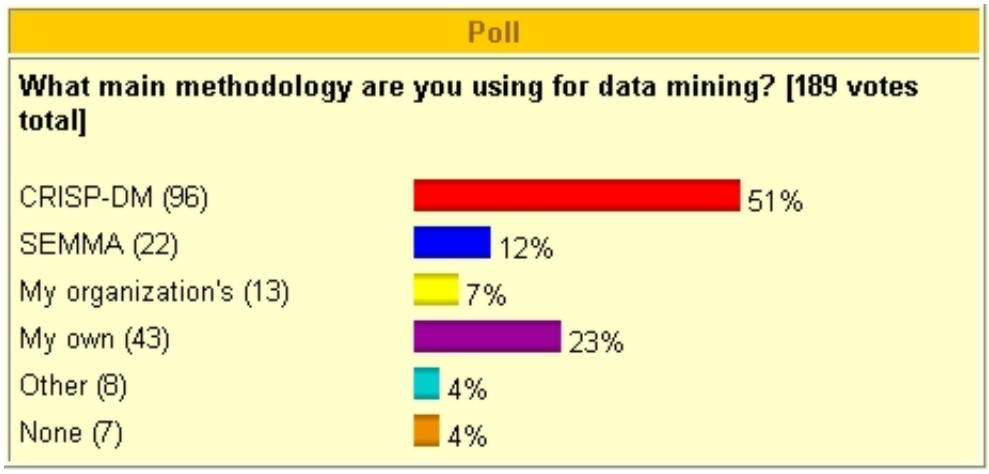


Abbildung 1.2: KDnuggets-Umfrage vom Juli 2002

auf riesigen Datenmengen. Speichermedien sind inzwischen so billig, dass nicht mehr mit Speicherplatz gehaushaltet werden muss und man also alle verfügbaren Daten speichern kann, auch solche, die für den eigentlichen Geschäftsprozess unwichtig sind, die aber vielleicht zusammen mit anderen Daten (und mit dem richtigen Data Mining-Verfahren) wertvolle Informationen liefern können. Auf Prozessorebene hat neben der ständigen Beschleunigung vor allem die parallele Technologie die Entwicklung des Data Mining vorangetrieben, da viele Data Mining-Algorithmen „von Natur aus“ parallel sind. Die verbesserte Rechnerleistung ermöglichte auch die Entwicklung hochwertiger Visualisierungswerkzeuge, welche eine der Stärken des Data Mining ausmachen.

- Zusammenarbeit von Forschern und Unternehmern

  Der Kontakt von Forschungszentren und Universitäten mit Industrie und Handel wächst ständig. Die Entwicklung neuer Algorithmen erfolgt (teilweise) im Hinblick auf ihren Nutzen in Unternehmen, also mit verstärktem Blick auf die Skalierbarkeit und Anwendbarkeit in den Unternehmen.

Wie man sieht, gibt es also auch auf technischer Seite viele Faktoren, die die rasche Entwicklung von Data Mining sehr begünstig(t)en und die zusammen mit den oben genannten markttechnischen Faktoren (siehe Kap. 1.2.1) den Erfolg von Data Mining bewirken.

1.3 Der Data Mining-Prozess

Sucht man nach einem Prozessmodell für den Data Mining-Prozess, so findet man wohl so viele verschiedene Modelle wie es Hersteller von Data Mining-Tools gibt. Bisher gibt es noch keine allgemeingültigen Standards. In der Literatur orientiert man sich jedoch mehr und mehr an CRISP-DM, einem von einem Konsortium großer

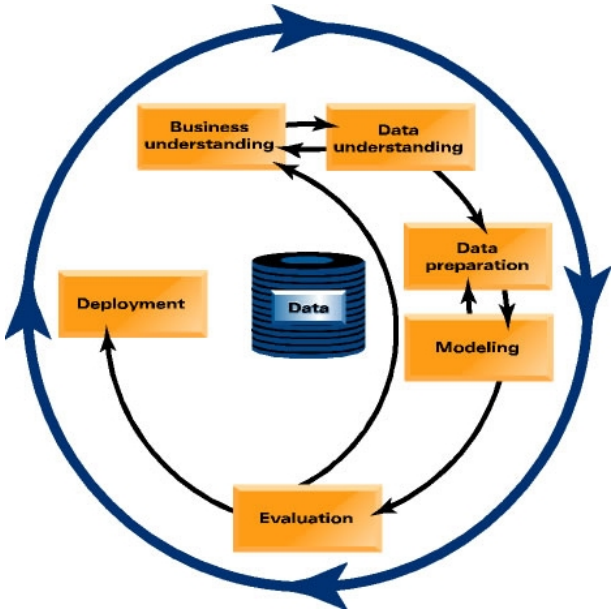


Abbildung 1.3: Das CRISP-Modell

Firmen aus verschiedenen Industriebereichen[3] entwickelten, produktunabhängigen Data Mining-Standard. CRISP-DM steht für *Cross-Industry Standard Process for Data Mining* und wurde mit Unterstützung der EU ab 1997 entwickelt[4]. Laut einer Internet-Umfrage des Data Mining-Portals KDnuggets.com im Juli 2002 ist das CRISP-Modell das mit Abstand am häufigsten in den Unternehmen verwendete Modell (siehe Abb.1.2). Viele Teilnehmer der Umfrage meinten, der Vorteil von CRISP-DM liege darin, dass es hilft, ein Geschäftsproblem in ein Data Mining-Problem umzuwandeln, ohne dabei das Geschäftsziel aus den Augen zu verlieren.
Auch hier soll nun im weiteren CRISP-DM als Grundlage für die Prozessbeschreibung dienen.
Der Data Mining-Prozess besteht aus 6 Schritten:

1. Verstehen der Geschäftsabläufe (business understanding)
2. Verstehen der Daten (data understanding)
3. Vorverarbeitung der Daten (data preparation)
4. Modellierung (modeling)
5. Evaluierung des Datenmodells (evaluation)
6. Umsetzen des Datenmodells (deployment)

[3]NCR, Daimler Chrysler, SPSS, Integral Solutions Limited(ISL), OHRA
[4]Für weitere Informationen siehe www.crisp-dm.org

Abbildung 1.3 illustriert den Prozess mit seinen einzelnen Schritten. Sie verdeutlicht auch eine wesentliche Eigenschaft des Data Mining-Prozesses: er läuft keineswegs linear durch die einzelnen Phasen, sondern springt immer wieder zu vorherigen Phasen zurück. Es hängt vom Ergebnis der einzelnen Phase ab, welche Phase oder welcher Teil einer Phase als nächstes durchgeführt wird (wenn man z.B. bei der Untersuchung der Daten feststellt, dass man zu wenig Daten hat, muss man noch einmal einen Schritt zurückgehen und neue Daten hinzunehmen). Der äußere Kreis in der Abbildung soll den Charakter des Data Mining als geschlossenen Prozess hervorheben: Ein Data Mining Prozess geht auch nach der Erarbeitung einer Lösung weiter, denn was man während des Prozesses gelernt hat, kann Anstoß zu neuen, oft detaillierteren Geschäftsfragen sein. Bei späteren Data Mining-Prozessen profitiert man immer von der Erfahrung, die man während der vorangegangenen Prozesse gesammelt hat.
Im Folgenden sollen nun die einzelnen Schritte genauer betrachtet werden (siehe auch [Ca00]).

1.3.1 Business understanding

Unerlässliche Voraussetzung für erfolgreiches Data Mining ist ein Verständnis für die Geschäftsprozesse, denn ohne diesen Hintergrund ist es unmöglich, die Probleme, die man lösen möchte, genau zu identifizieren, die Daten für das Mining vorzubereiten oder die Ergebnisse richtig zu interpretieren. Neben der Entwicklung dieses Verständnisses soll im ersten Schritt das genaue Ziel des Data Mining-Prozesses festgelegt werden und es soll herausgearbeitet werden, was zur Erreichung dieses Ziels alles notwendig ist. Es sollen also Fragen wie die Folgenden geklärt werden:

- Was ist das Ziel der Analyse?
  - Will man neue Erkenntnisse gewinnen (z.B. Identifikation von Kundengruppen) oder sollen Geschäftsprozesse unterstützt werden (z.B. Entscheidungshilfe bei der Vergabe von Krediten)?
  - Welche Prozesse sollen wie unterstützt werden? (Life Cycle Management, Neukundenwerbung, Cross Selling,...)
  - Umsetzung über welche Kanäle? (Filialen, Mailing, Internet, Außendienst,...)
- Wie müssen die Ergebnisse aussehen, damit sie verwertbar sind?
- Welche Personen sind in das Projekt einzubinden, damit es über alle Phasen hinweg Erfolg hat (Einbindung von Experten)?

1.3.2 Data understanding

Dieser Schritt beginnt mit dem Sammeln von (firmeninternen und externen) Daten. Anschließend sollte man versuchen, sich mit den Daten vertraut zu machen, qualitativ schlechte Daten zu erkennen und auch schon eventuell interessante Teilmengen der Daten zu identifizieren. Es sollen Fragen geklärt werden wie z.B.:

- Welche Daten stehen in Bezug zu welchem Teil des erstellten Modells?

- Wie soll mit unreinen Daten (bei denen beispielsweise Werte fehlen) verfahren werden?
- Welche Metadaten sind mit den Daten verknüpft?
- Was ist die zugrunde liegende zeitliche Dimension oder zeitliche Repräsentation der Daten (z.B. für Zeitreihenanalyse)?
- Wie sind die Daten gespeichert?
- Sind die Daten repräsentativ für die aktuelle Fragestellung?

1.3.3 Data preparation

Diese Phase umschließt alle Schritte, die notwendig sind, um aus den Rohdaten eine Datenbank zu konstruieren, die den Modellierungs-Tools als Eingabe dient. Sie besteht im Wesentlichen aus 5 Teilschritten:

1. Selektion

   Auswahl der Daten, die tatsächlich in das Verfahren einfließen sollen. Die Auswahl ist unter anderem abhängig von den Data Mining-Zielen, der Qualität und eventuellen technischen Beschränkungen (z.B. Beschränkung des Datenvolumens). Auswahl bedeutet nicht nur Auswahl bestimmter Tabellen, sondern auch innerhalb der Tabellen Auswahl bestimmter Attribute (Spalten) oder Datensätze (Zeilen).

2. Bereinigung

   Die Qualität der Daten muss so weit gesteigert werden wie es das jeweilig gewählte Verfahren erfordert. Dies geschieht u.a. durch Auswahl reiner Daten (d.h. ohne fehlende/falsche Werte), Einfügen geeigneter Default-Werte oder auch Ersetzen fehlender Werte durch geschätzte Werte.

3. Daten konstruieren

   Je nach Ziel des Data Mining-Verfahrens sind manche Attribute nicht sehr aussagekräftig, oft lässt sich aber aus verschiedenen Attributen ein neues generieren (z.B. Länge·Breite = Fläche), das sinnvoll in das Verfahren eingebunden werden kann. Ebenso kann es sinnvoll sein, verschiedene Werte eines Attributs zu einem zusammenzufassen („Diskretisierung“) oder verschiedene Objekte zu einem zusammenzufassen („Aggregation“).

4. Daten integrieren

   In diesem Schritt werden verschiedene Tabellen zu einer zusammengefasst. Welche Tabellenwerte dabei in die neue Tabelle eingehen, hängt von den Zielen des Verfahrens ab.

5. Daten formatieren

   Manche Data Mining-Tools stellen bestimmte Anforderungen an die Reihenfolge der Datensätze und/oder Attribute, z.B. dass das erste Feld für jeden Datensatz

einen Identifizierer enthalten muss oder dass das letzte Feld das vorherzusagende Attribut enthalten soll. Dafür müssen die Daten natürlich noch entsprechend formatiert werden.

Es ist sehr wahrscheinlich, dass diese Datenaufbereitung des öfteren wiederholt werden muss (jedoch nicht zwangsläufig in der hier beschriebenen Reihenfolge), da beispielweise bei späteren Schritten erkannt wird, dass weitere Attribute in die Berechnungen miteinbezogen werden müssen bzw. dass gewisse Attribute miteinander korrelieren und deshalb ein Teil davon ausgeschlossen werden muss.
Diese Phase des Prozesses ist mit Abstand die aufwändigste. Man schätzt, dass sie 60-80% des gesamten Zeitaufwands benötigt [dV01].

1.3.4 Modeling

In dieser Phase kommen nun die Data Mining-Algorithmen zum Einsatz. Sie gliedert sich in 4 Teilschritte:

1. Wahl des Verfahrens

   Für denselben Problemtyp stehen in der Regel verschiedene Techniken (bzw. Algorithmen[5]) zur Erstellung eines Modells zur Auswahl, z.B. Entscheidungsbäume, neuronale Netze, Single Link usw. Die Wahl der Technik(en) und die Parameterwahl innerhalb der einzelnen Techniken werden natürlich stark von der Problemstellung beeinflusst. Da die einzelnen Verfahren meist unterschiedliche Annahmen über die Daten machen, kann es sein, dass man noch einmal zur Datenaufbereitungsphase zurückgehen und die Daten entsprechend präparieren muss.

2. Erstellen eines Testdesigns

   Ein Testdesign ist eine Prozedur oder ein Mechanismus, mit dessen Hilfe man später die Qualität und die Gültigkeit des Modells überprüfen kann. Soll z.B. eine Klassifikation durchgeführt werden, so wird oft die Fehlerrate als Qualitätskriterium genommen. Hierzu werden üblicherweise die Daten in Test- und Trainingsdaten aufgeteilt. Anschließend wird das Modell anhand der Trainingsdaten erstellt und dann mit Hilfe der (seperaten) Testdaten seine Qualität abgeschätzt.

3. Erstellen des Modells

   Nun kann durch Anwenden des gewählten Verfahrens auf die vorbereiteten Daten ein (oder mehrere) Modell(e) erstellt werden.

4. Bewertung des Modells

   In der Regel werden viele Modelle erstellt, indem zum einen verschiedene Techniken verwendet werden und zum anderen einzelne Faktoren der jeweiligen Technik (z.B. die Parameterwahl) modifiziert werden. Die entstandenen Modelle werden (u.a. mit Hilfe des gewählten Testdesigns) bewertet und nach Qualität

[5] im Folgenden werden „Technik“ und „Algorithmus“ als Synonyme verwendet.

geordnet. Ist man der Meinung, dass das Ergebnis z.B. durch eine andere Parameterwahl noch verbessert werden kann, so wird der Modellierungsschritt so lange iteriert, bis man meint, das bestmögliche Ergebnis gefunden zu haben.

1.3.5 Evaluation

Nach dem letzten Schritt hat man nun die Modelle gefunden, die in technischer Hinsicht am besten sind. In dieser Phase sollen diese Modelle nun auf ihren geschäftlichen Nutzen hin untersucht werden. Es muss geklärt werden, ob die Modelle den geschäftlichen Erfolgskriterien (die in der ersten Phase festgelegt wurden) gerecht werden und welche dies am besten tun. Hat man diese gefunden, wird der gesamte Prozess noch einmal durchgegangen, um sicher zu gehen, dass man im Verlauf des Prozesses keine Fehler gemacht oder wichtige Faktoren übersehen hat. Ist man mit dem Ergebnis zufrieden, kann man zur Umsetzung des Modells übergehen, andernfalls werden - in Abhängigkeit der verbleibenden Ressourcen - neue Iterationen durchgeführt oder sogar ein ganz neues Projekt begonnen.

1.3.6 Deployment

Mit dem Finden eines guten Modells ist der Data Mining-Prozess noch nicht zu Ende: nun müssen Strategien erarbeitet werden, wie das Modell am besten in Geschäftsprozesse umgesetzt werden kann und welche dieser Prozesse den größten Nutzen für das Unternehmen haben. Hierbei ist natürlich nicht nur Data Mining-Fachwissen, sondern auch Fachwissen im Bezug auf die Geschäftsabläufe gefragt. Hat man sich für eine Umsetzung entschieden, wird ein genauer Plan erstellt, der die einzelnen dazu nötigen Schritte enthält.

1.4 Wichtige Data Mining-Verfahren

Wie im vorigen Abschnitt schon angedeutet, hat man innerhalb der Data Mining-Tools verschiedene Verfahren zur Auswahl. Manche dieser Verfahren sind auf bestimmte Problemarten zugeschnitten, andere sind universeller auf mehrere Typen von Problemen anwendbar. Oft ist es auch sinnvoll, verschiedene Verfahren miteinander zu kombinieren. So wird etwa häufig bei der Erstellung von Vorhersagemodellen zuerst eine Cluster-Analyse durchgeführt und anschließend werden für die größten bzw. wichtigsten Cluster Vorhersagemodelle entwickelt.
Auch im Bezug auf Data Mining-Verfahren ist man sich in der Literatur nicht ganz einig, welche Verfahren die wichtigsten sind. Dies liegt wohl zum einen daran, dass sich manche Autoren an einem bestimmten Tool orientiert und folglich die darin implementierten Verfahren beschrieben haben. Zum anderen mag es daran liegen, dass ein Teil der Autoren aus der Wissenschaft stammt, und dadurch die Verfahren, die mehr im Mittelpunkt der Forschungsarbeit stehen, auch in ihren Büchern mehr in den Mittelpunkt gerückt werden. Jedoch gibt es einige „Klassiker“ unter den Verfahren, z.B. Cluster-Analyse (Segmentierung) oder Klassifikation, ohne deren Erläuterung ein Data Mining-Buch nicht vollständig wäre. Diese und weitere Verfahren, die häufig Erwähnung finden, sollen im Folgenden erläutert werden (siehe auch

[CHS+98], [KWZ98]).

1.4.1 Vorhersagen

Vorhersagemodelle sind mit dem menschlichen Lernprozess verwandt, bei dem man Beobachtungen benutzt, um sich ein Modell von der zugrunde liegenden Struktur der beobachteten Dinge zu erstellen. Wenn zum Beispiel einem kleinen Kind im Laufe der Zeit verschiedene Exemplare von Hunden gezeigt werden, so kann es später aufgrund gewisser Charakteristika von Hunden neue Tiere eindeutig als Hund bzw. „Nicht-Hund“ klassifizieren. Beim Data Mining funktioniert die Erstellung von Vorhersagemodellen nach dem gleichen Prinzip: Man verwendet bereits vorhandene Daten, um deren wesentliche Charakteristika zu bestimmen. Damit das Modell Vorhersagen machen kann, werden dem Algorithmus die korrekten Antworten für bereits gelöste Fälle gegeben. Diese Funktionsweise eines Algorithmus nennt man *überwachtes Lernen (supervised learning).*
Die Erstellung eines Vorhersagemodells erfolgt in zwei Phasen: Training und Test. Beim Training wird mit einem ausreichend großen Teil aller verfügbaren Daten ein Modell erstellt, welches anschließend beim Testen mit einem kleineren Teil der Daten auf seine Genauigkeit und physikalische Performance hin überprüft wird.
Vorhersagemodelle werden in vielen Branchen häufig angewendet, z.B. für das Management der Kunden (welche Kunden wollen eventuell zur Konkurrenz wechseln?), Kreditbewilligungen (welche Kunden können ihren Kredit sehr wahrscheinlich (nicht) zurückzahlen?) oder cross selling (welche Produkte will ein Kunde, der Produkt X kauft, eventuell noch kaufen?).
Man unterscheidet zwei Arten von Vorhersagemodellen: *Klassifikation* und *(numerische) Wertvorhersage*. Sie haben beide das Ziel, eine interessante Variable möglichst gut vorherzusagen, jedoch unterscheiden sie sich in der Art der vorherzusagenden Variablen.
Bei der Klassifikation kann diese Variable für jeden Datensatz einen von endlich vielen, vorbestimmten Werten (*Klassenwert)* annehmen. Bei der Vorhersage „untreuer“ Kunden wären beispielsweise die zwei Werte (*Klassen*) „BLEIBT“ und „WECHSELT“ denkbar.
Bei der numerischen Wertvorhersage kann die Variable einen beliebigen numerischen Wert (innerhalb eines Intervalls) annehmen. Dies ist zum Beispiel der Fall, wenn eine Versicherung die zu erwartende Schadenssumme eines potentiellen neuen Versicherungsnehmers vorhersagen will. Ein Spezialfall der Wertvorhersage ist das so genannte *scoring*, bei dem die vorherzusagende Variable eine Wahrscheinlichkeit darstellt. Dies ist dann so zu interpretieren, dass ein gewisses Ereignis umso wahrscheinlicher ist, je höher der Wert ist, den die Variable annimmt. Typische Anwendungen hierfür sind z.B. die Vorhersage der Wahrscheinlichkeit von Kreditkartenmissbrauch oder der Wahrscheinlichkeit, dass ein Kunde auf einen Werbebrief antworten wird.

1.4.2 Assoziationen

Das Ziel von Assoziationen ist es, Regeln zu finden, mit deren Hilfe man von der Präsenz gewisser Dinge oder Eigenschaften auf die Präsenz anderer Din-

ge/Eigenschaften schließen kann. Bei Warenkorbanalysen wird dies häufig angewendet: man denke sich eine Datenbank bestehend aus den Transaktionen (Einkäufen) der Kunden. Jede Transaktion besteht aus mehreren Posten, nämlich denjenigen, die von einem Kunden gleichzeitig gekauft wurden. Wendet man nun hierauf Techniken zum Entdecken von Assoziationen an, wird man Affinitäten zwischen gewissen Posten entdecken, d.h. man findet Posten, die häufig zusammen gekauft werden. Diese Affinitäten werden durch Assoziationsregeln repräsentiert. Wie so eine Warenkorbanalyse aussehen kann zeigt das Beispiel unten.
Das Ableiten der Regeln ist im Allgemeinen nicht das Problem, denn die Assoziationsalgorithmen können dies sehr effizient tun. Die eigentliche Herausforderung liegt in der Beurteilung der Vertrauenswürdigkeit und des Nutzens der Regeln. Diese zwei Faktoren werden durch die zwei Parameter *Konfidenz* und *Support* ausgedrückt.
Hat man eine Assoziationsregel der Form $A \Rightarrow B$, so berechnet sich

- die Konfidenz durch

$$conf(A \Rightarrow B) = \frac{\#Transaktionen\ mit\ A\ und\ B}{\#Transaktionen\ mit\ A} ,$$

  sie drückt also die relative Häufigkeit der Transaktionen mit A und B in der Menge der Transaktionen mit A aus.

- der Support durch

$$supp(A \Rightarrow B) = \frac{\#Transaktionen\ mit\ A\ und\ B}{\#\ Transaktionen} ,$$

  er drückt also die relative Häufigkeit der Transaktionen mit A und B in der Menge aller Transaktionen aus.

Oft wird bei Assoziationsregeln auch noch der *Lift* betrachtet, welcher folgendermaßen definiert ist:

$$lift(A \Rightarrow B) = \frac{conf(A \Rightarrow B)}{P(B)},\ mit\ P(B) = \frac{\#Transaktionen\ mit\ B}{\#Transaktionen}$$

Der Lift gibt an, um wieviel sich die Wahrscheinlichkeit, dass B gekauft wird, ändert, wenn auch A gekauft wird. Ein Lift von 1 bedeutet etwa, dass sich die Wahrscheinlichkeit, dass B gekauft wird, nicht verändert wenn gleichzeitig auch A gekauft wird. A hat in diesem Fall keinen Einfluss auf den Kauf von B. Ein Lift von 4 bedeutet, dass die Wahrscheinlichkeit, dass B in einem Warenkorb vorkommt, viermal so hoch ist als normal, falls in dem Warenkorb auch A vorkommt.

Beispiel für eine Assoziationsregel
In einem Laden wurden die Artikel A, B, C, D und E in folgenden Kombinationen gekauft:
Transaktion 1 = {A, C}
Transaktion 2 = {A, B, C, D, E}
Transaktion 3 = {C, E, E}

Transaktion 4 = {A, B, D}
Transaktion 5 = {D, D, E}
Transaktion 6 = {B, C}
Transaktion 7 = {A, B, C, E}
Transaktion 8 = {C, D, D, D}
Transaktion 9 = {E}
Transaktion 10 = {A, B}

Für die Regel $A \Rightarrow B$ ergibt sich dann

- eine Konfidenz von $conf(A \Rightarrow B) = \frac{4}{5} = 80\%$,
- ein Support von $supp(A \Rightarrow B) = \frac{4}{10} = 40\%$ und
- ein Lift von $lift(A \Rightarrow B) = \frac{4/5}{5/10} = \frac{8}{5} = 1.6$.

Wenn nun beispielsweise A für Hemden und B für Krawatten steht, so bedeutet diese Regel anschaulich:
Wenn ein Kunde ein Hemd kauft, so kauft er in 80% aller Fälle auch eine Krawatte. Dies ist in 40% aller Einkäufe der Fall. Kauft ein Kunde ein Hemd, so steigt die Wahrscheinlichkeit, dass er auch eine Krawatte kauft, um 60%.
Natürlich ist man meist an Regeln mit hohen Konfidenz- und Support-Werten interessiert. Sind die Werte zu niedrig, so erhält man viele mögliche Kombinationen von Produktassoziationen, was oft nicht sehr aussagekräftig ist. Beim Lift ist man nicht nur an großen, sondern auch an kleinen Werten interessiert, denn die kleinen Werte (gemeint sind Werte kleiner als 1) geben Aufschluss darüber, ob ein Artikel hinderlich für den Verkauf eines anderen Artikels ist; beispielsweise wird die Regel *Coca Cola* $\Rightarrow$ *Pepsi Cola* wohl meist einen sehr kleinen Support haben, denn nur wenige, die Coca Cola kaufen, werden gleichzeitig auch Pepsi Cola kaufen.
Oft kommt es auch vor, dass die Daten in natürlicher Weise miteinander korrelieren, was zu unbrauchbaren Ergebnissen führt. So ist es etwa nicht sehr erstaunlich, dass Flaschen und Pfand immer zusammen gekauft werden. Die Wahrscheinlichkeit solcher Korrelationen steigt mit der Anzahl der Datensätze; in großen Datenbeständen sind solche Korrelationen also fast immer zu finden.
Der wesentliche Vorteil dieses Data Mining-Verfahrens liegt darin, dass es so einfach ist. Man muss lediglich zwei Parameter (Support und Konfidenz) festsetzen und die Regeln, die man erhält, sind leicht zu interpretieren.
Ein Nachteil von Assoziationen ist, dass der unternehmerische Wert einer Regel nicht berücksichtigt werden kann. Wenn also beispielsweise Chips häufig zusammen mit einem bestimmten Getränk verkauft werden, so macht es für den Algorithmus keinen Unterschied, ob dieses Getränk teurer Wein oder einfach nur Mineralwasser ist, für den Einzelhändler jedoch macht dieser Unterschied sehr viel aus [CHS+98].

1.4.3 Cluster-Analyse

Da auf die Cluster-Analyse in Kapitel 2 ausführlich eingegangen wird, soll an dieser Stelle nur ein kurzer Überblick gegeben werden.

Ziel einer Cluster-Analyse ist es, einen gegebenen Datenbestand so in Teilmengen (*Cluster*) einzuteilen, dass Datensätze innerhalb eines Clusters möglichst ähnlich, Datensätze aus unterschiedlichen Clustern aber möglichst verschieden sind. Diese „Ähnlichkeit“ bzw. „Verschiedenheit“ wird durch Ähnlichkeitsmaße bzw. Distanzmaße (siehe Kap. 2.4) ausgedrückt, deren Wahl großen Einfluß auf das Ergebnis der Clusterung hat.
Es gibt verschiedene Arten von Clustering-Verfahren, die am häufigsten verwendeten sind *partitionierende* und *hierarchische Verfahren:*
Partitionierende Verfahren zerlegen eine Datenmenge so in k Cluster, dass jedes Cluster mindestens einen Datensatz enthält und dass jeder Datensatz zu genau einem Cluster gehört.
Hierarchische Verfahren hingegen erzeugen keine einfache Zerlegung der Datenmenge, sondern eine hierarchische Struktur der Daten, aus der man eine Clusterung ableiten kann [ES00].
Im Gegensatz zu den Vorhersage-Verfahren gehört die Cluster-Analyse zu den Methoden des unüberwachten Lernens (*unsupervised learning*): es wird keine Klasseneinteilung vorgegeben, vielmehr ist es Aufgabe des Verfahrens, anhand der Ähnlichkeiten bzw. Distanzen der Daten die Klasseneinteilung zu erzeugen.
In einem Data Mining-Prozess ist das Clustering (nach dem Preprocessing) oft der erste Schritt, da auf diese Weise homogene Datengruppen identifiziert werden können, auf die dann speziell eingegangen werden kann. Darum findet die Cluster-Analyse auch in fast allen Bereichen Anwendung, wie z.B.

- bei Versicherungen (Identifizierung von Gruppen von Versicherten mit einer hohen durchschnittlichen Schadenssumme)
- in der Stadtplanung (Finden von Häusergruppen anhand des Haustyps, des Werts und der geographischen Lage)
- in den Wirtschaftswissenschaften, insbesondere in der Marktforschung
- im WWW
  - Klassifikation von Dokumenten im WWW
  - Erkennung ähnlicher Bewegungsmuster der User durch Clusterung der Einträge in der Log-Datei
- in der Bildverarbeitung
- in der Mustererkennung
- im Marketing (Erkennen unterschiedlicher Kundengruppen für gezielte Marketing-Aktionen)
- in der Landnutzung (Identifizieren von Gebieten mit gleicher Nutzung)
- in der Erdbeben-Forschung (Clusterung beobachteter Erdbeben-Epizentren)

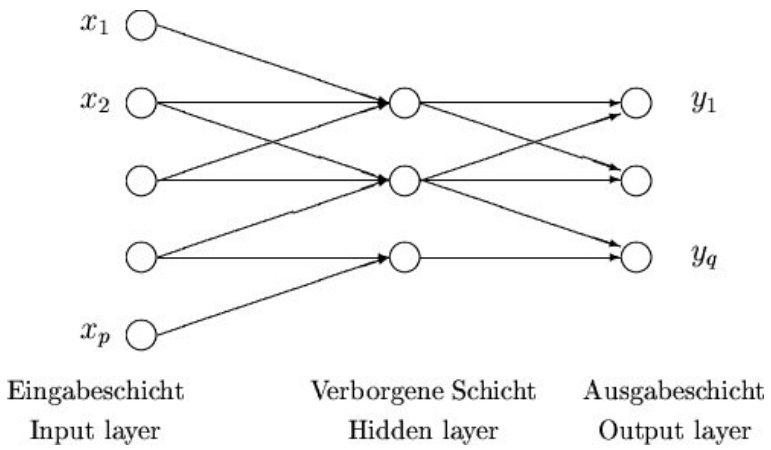


Abbildung 1.4: Beispiel eines neuronalen Netzes

1.5 Data Mining-Techniken

Hat man sich für ein Data Mining-Verfahren entschieden, so muss noch eine Data Mining-Technik gewählt werden, mit deren Hilfe das Verfahren umgesetzt wird. Hierbei gibt es keine $1:1$ bzw. $1:n$-Verknüpfung, d.h. ein Verfahren kann von verschiedenen Techniken umgesetzt werden, aber eine Technik kann auch verschiedene Verfahren umsetzen. Bei jedem Verfahren gibt es jedoch Techniken, die bevorzugt verwendet werden. Im Folgenden werden neuronale Netze und Entscheidungsbäume als Techniken für numerische Wertvorhersage und Klassifikation vorgestellt, Clustering-Techniken werden in Kapitel 2 ausführlich behandelt.

1.5.1 Neuronale Netze

Neuronale Netze werden in der Regel für Klassifikationen oder numerische Wertvorhersagen (*Regression*) verwendet. Die Terminologie ist der Biologie entlehnt: In neuronalen Netzen soll die Funktionsweise des menschlichen Nervensystems imitiert werden. Die Architektur sieht so aus, dass Basiselemente (*Neuronen*) in verschiedene Schichten (*layers*) gruppiert und miteinander verbunden werden, wobei ein Neuron immer nur mit Neuronen aus den direkt benachbarten Schichten verkoppelt sein kann. Die Schichten werden meist in sichtbare und verborgene Schichten (*hidden layers*) eingeteilt. Die sichtbaren Schichten sind die Eingabe- und die Ausgabeschicht (*input/output layer*), welche die Eingabewerte übernehmen bzw. die Ausgabewerte bereitstellen. Zwischen diesen beiden Schichten liegen eventuell mehrere verborgene Schichten. Abbildung 1.4 zeigt ein neuronales Netz mit nur einer verborgenen Schicht [Dil01].
Jedes Neuron besitzt ein gewisse Anzahl an Eingabeleitungen und eine oder mehrere Ausgabeleitungen. In der Eingabeschicht hat jedes Neuron nur eine Eingabeleitung, über die jeweils eine der Eingabegrößen eingeht. Der Ausgabewert berechnet sich mit Hilfe einer Schwellwertfunktion, die als Eingabe die Summe der gewichteten Eingabewerte erhält. Bei einem neuronalen Netz wird also ein bestimmtes Muster in den input

layer eingegeben, welches mittels einfach vernetzter Elemente in den hidden layers weiterverarbeitet wird und schließlich im output layer ein Ausgabemuster erzeugt. Nach dem Entwurf einer Architektur muss das neuronale Netz anhand von Beispieldaten trainiert werden (siehe Kap. 1.4.1). Wichtig sind hierbei die Gewichte der Eingabewerte: sie sind die Parameter, die während der Trainingsphase verändert werden. Die Schwellwertfunktion hingegen ist eine Invariante; sie wird beim Entwurf einer Architektur für eine Problemstellung gewählt und anschließend nicht mehr verändert.

Genetische Algorithmen

Genetische Algorithmen, die der Lösung von Optimierungsaufgaben dienen, sind ähnlich wie neuronale Netze stark von biologischen Vorstellungen geprägt: sie sind dem Evolutionsprozess nachempfunden (*survival of the fittest*). Die zugrunde liegenden Daten (*Individuen*) werden binär kodiert (*Chromosomen*) und anschließend zufällig verteilten „Mutationen“ ausgesetzt. Mutation bedeutet entweder die Veränderung eines Bits im Bitstring oder die „Kreuzung“ zweier Individuen, also Auseinanderschneiden der zugehörigen Bitstrings an einer zufällig gewählten Stelle, Austauschen zweier Hälften und Zusammenkleben. Im darauffolgenden Selektionsschritt wird die Fitness jedes Individuums mit Hilfe einer Fitnessfunktion bewertet. Die Fitness bestimmt die Chance auf Reproduktion und damit auf Übernahme in die nächste Generation. Nach mehreren Generationen werden aus der „aktuellen“ Generation die Individuen mit den höchsten Fitnesswerten als Lösung des Optimierungsproblems gewählt.
Beim Data Mining kann man genetische Algorithmen zwar nicht unbedingt direkt zum Finden von Strukturen und Zusammenhängen in den Daten verwenden, sie können aber beispielsweise eingesetzt werden, um Lernprozesse anderer Data Mining-Algorithmen, wie z.B. neuronaler Netze, zu steuern. (Man denke sich bei einem neuronalen Netz die Menge aller möglichen Modelle als Individuen einer Population, jedes Individuum wird durch einen Bitstring repräsentiert, in dem ein bestimmte Verteilung der Gewichte kodiert ist. Ein „fittes“ Individuum entspricht dann einer guten Gewichtsverteilung).
Die Stärke genetischer Algorithmen liegt in der Bearbeitung von Aufgaben, bei der eine große Anzahl von Parametern eingehen [Mer01]. Dies macht sie für den Einsatz beim Data Mining interessant. Jedoch bringen sie auch einen großen Rechenaufwand für das Pre- und Postprocessing mit sich.

1.5.2 Entscheidungsbäume

Entscheidungsbäume werden für Klassifikationsaufgaben verwendet. Sie sind wahrscheinlich die am meisten verwendete Data Mining-Technik [dV01], was wohl unter anderem daran liegt, dass sie für den Anwender leichter nachvollziehbar sind als andere Techniken und es deswegen auch einfacher ist, zu interagieren um das gewünschte Ergebnis zu erhalten.
Ein Entscheidungsbaum ist ein Baum, bei dem alle Knoten und Kanten markiert sind:

- jedes Blatt des Baumes enthält eine Klasse, die als Ergebnis zurückgegeben wird

- jeder innere Knoten enthält ein Merkmal, nach dessen Ausprägung verzweigt wird
- jede Kante enthält eine Merkmalsausprägung des Merkmals, das im Anfangsknoten der Kante steht.

Der Aufbau eines Entscheidungsbaumes (*growing phase*) erfolgt wie bei neuronalen Netzen mit Trainingsdaten. Ist das Ergebnis mit den Testdaten zufriedenstellend, so wird der Baum im Allgemeinen noch beschnitten (*pruning phase*), da kleinere Entscheidungsbäume meist eine bessere Klassifikationsgüte besitzen [ES00] und leichter zu interpretieren sind.
Die meisten Entscheidungsbaum-Klassifikatoren führen binäre Splits durch, deren Ergebnis dann eventuell durch weitere (binäre) Splits noch verfeinert wird. Es sind aber auch Splits mit mehr als zwei Merkmalsausprägungen möglich. Die Strategie, nach der gesplitet wird, ist entscheidend für die Qualität des entstehenden Baumes. Gesucht sind Splits, die möglichst reine Partitionen (bezüglich der Klassenzugehörigkeit) erzeugen. Um diese Qualität zu messen, wurden verschiedene Maße entwickelt, wie z.B. der Informationsgewinn oder der Gini-Index (für genaue Definitionen dieser Maße siehe [ES00]).
Die Laufzeit eines Algorithmus zum Finden des optimalen Entscheidungsbaumes ist exponentiell in der Anzahl der Attribute. Darum wird ein *Greedy-Algorithmus* verwendet: Es wird jeweils nur das nächste Splitattribut ausgewählt, und es wird kein Backtracking durchgeführt.

Kapitel 2

Cluster-Analyse

In diesem Kapitel soll nun ein spezielles Data Mining-Verfahren genauer betrachtet werden: das Clustering. (Ein kurzer Überblick wurde bereits in Kap. 1.4.3 gegeben). Ziel einer Cluster-Analyse ist es, die einer Menge von Objekten zugrunde liegenden Strukturen zu entdecken, indem die Objekte entweder in (meist) disjunkte Teilmengen aufgeteilt werden oder sie als eine Hierarchie von Gruppen und Untergruppen dargestellt werden. Hierbei wird ein Objekt in der Regel durch eine Menge von Attributen repräsentiert. Man möchte also eine große Datenmenge durch eine kleinere ersetzen, die leichter zu interpretieren und zu handhaben ist, ohne dabei viele Informationen über die Daten zu verlieren. Hierbei sollen homogene Untergruppen durch möglichst gut gewählte Repräsentanten dargestellt werden. Wie dies konkret aussehen kann wird im ersten Abschnitt dieses Kapitels anhand eines Beispiels erläutert.
Ein wesentliches Merkmal der Cluster-Analyse ist die Tatsache, dass es sich um ein Verfahren des *unüberwachten Lernens* handelt, d.h. die Charakterisierung der Gruppen ist vorab nicht bekannt, sondern entsteht erst im Laufe des Verfahrens, im Gegensatz zur Klassifikation, bei der vorab Gruppen-Label definiert werden. Eine gute Clusterung zeichnet sich dadurch aus, dass die Objekte bzw. Datensätze innerhalb einer Gruppe möglichst ähnlich und die Datensätze unterschiedlicher Gruppen möglichst verschieden sind. Wie sich schon vermuten lässt, hängt das Ergebnis einer Cluster-Analyse stark davon ab, wie man diese Ähnlichkeit definiert. Denn je nach Wahl des Ähnlichkeitsmaßes werden eventuell nur kugelförmige oder längliche Cluster gefunden. Möchte man „unförmigere“ Cluster finden, muss man eventuell ein anderes Ähnlichkeits- bzw. Distanzmaß wählen. Auf die verschiedenen Maße, die hierbei möglich sind, wird in Abschnitt 2.4 eingegangen.
Doch an ein gutes Clustering-Verfahren werden noch weitere Anforderungen gestellt:

- Skalierbarkeit (siehe Def. 1.1.1)

  Die Performance vieler Cluster-Analyse-Verfahren ist bei kleinen Datenmengen recht gut, nimmt aber mit Zunahme der Datenmenge rapide ab. Gesucht sind Algorithmen, die auch bei großen Datenmengen noch eine vertretbare Laufzeit haben.

- Unabhängigkeit von der Datenreihenfolge

  Ändert man die Reihenfolge, in der die Daten eingegeben werden, so sollte

ein gutes Clustering-Verfahren die gleiche oder doch zumindest eine ähnhliche Clusterung liefern.

- Anwendbarkeit ohne Vorkenntnisse

  Viele Clustering-Verfahren verlangen vom Benutzer das Festsetzen gewisser Parameter, wie z.B. die Anzahl der Cluster oder die maximale Clustergröße. Da die Clusterung wesentlich von diesen Parametern abhängt, ist es wichtig, diese Parameter möglichst gut zu wählen. Dies kann der Benutzer aber nur, wenn er gewisse Kenntnisse über seine Daten hat und in etwa weiß, wie der gewählte Clustering-Algorithmus funktioniert. Hier wäre eine Unterstützung durch das Verfahren (z.B. (halb-)automatische Parameterwahl) wünschenswert.

- Toleranz gegenüber fehlerhaften Daten

  Oft kommt es vor, dass bei einigen Datensätzen Attributwerte fehlen oder falsch sind (*Ausreißer*). Bei einem schlechten Verfahren verfälscht dies die Ergebnisse und macht sie unbrauchbar.

- Invarianz unter Skalentransformationen

  Ändert man die Maßeinheit für ein bestimmtes Attribut (z.B. von Grad Celsius auf Grad Fahrenheit), so sollte sich das Ergebnis des Clustering-Verfahrens nicht oder nur unwesentlich verändern.

- Anwendbarkeit auch bei hoher Dimensionalität

  Je höher die Anzahl der Attribute ist, d.h. je größer die Dimension des Datenraumes ist, desto weiter liegen die einzelnen Datensätze auseinander und desto schwieriger ist es, Cluster zu finden. Ein guter Algorithmus sollte auch bei hoher Dimensionalität gute Ergebnisse liefern.

- Leicht interpretierbare Ergebnisstruktur

  Die gefundene Clusterung sollte für den Anwender interpretierbar sein, da er sonst keinen Nutzen daraus ziehen kann.

Abschließend sei noch einmal darauf hingewiesen, dass das Clustering (als Spezialfall eines Data Mining-Verfahrens) im Kontext des in Kapitel 1.3 beschriebenen Prozesses zu sehen ist, d.h. zu einer erfolgreichen Cluster-Analyse gehören auf jeden Fall die Vor- und Nachverarbeitung der Daten, sowie die richtige Interpretation und Umsetzung des gewonnenen Wissens.

2.1 Ein Beispiel: Lebensstile in Augsburg

In einem drei Jahre dauernden Lehrforschungsprojekt am Lehrstuhl Soziologie und empirische Sozialforschung an der wirtschaftswissenschaftlichen Fakultät der Universität Augsburg versuchten Studenten, Aufschluss über die in Augsburg vorherrschenden Lebensstile zu gewinnen. Neben diesem Hauptziel sollten auch noch Erkenntnisse über sozialstrukturelle und politische Aspekte, Stadtimage, Verkehrsverhalten, Wohnen, Konsum- und Medienpräferenzen gewonnen werden. Die Studenten entwickelten

einen 16-seitigen Fragebogen und befragten damit knapp 2000 Augsburger Haushalte. Aus den daraus erhaltenen Daten wurden dann mittels Cluster-Analyse unter Einbeziehung von 78 Variablen aus unterschiedlichen Bereichen folgende 8 Lebensstile ermittelt:

1. die hochkapitalisierten Midlife-men
2. die gutsituierten Hardrock-Familienväter
3. die kleinbürgerlichen Arbeiter und Angestellten
4. die schlechtsituierten, konservativen Älteren
5. die linken, jungledigen Intellektuellen
6. die extrem Unextremen
7. die jungen Techno-Mieter
8. die religiösen Volksmusik-Rentner

Die Benennung der Cluster erfolgte anhand derjenigen Variablen, die für das jeweilige Cluster am charakteristischsten waren. Dies waren beispielsweise

- für die hochkapitalisierten Midlife-men: Altersdurchschnitt 54 Jahre - höchstes Nettoeinkommen - aus „guten“ Verhältnissen - mit Hochschulabschluss und akademischem Beruf - überwiegend Männer - meist verheiratet mit Kindern - Sympathien für die FDP.
- für die gutsituierten Hardrock-Familienväter: Altersdurchschnitt 39 Jahre - hoher Verdienst - Vorlieben für Rock, Blues und Heavy Metal - meist verheiratet mit Familie - überwiegend Männer - relative Vorlieben für Erotik- und Horrorfilme.
- für die linken, jungledigen Intellektuellen: Altersdurchschnitt 29 Jahre - Vorlieben für ÖDP und PDS - meist ledig - meist Studenten und Abiturienten - Kritik an der Kirche - Vorlieben für Heavy Metal, Grunge und Punk - Ablehnung von Volksmusik und Schlager - Vorlieben für Science-Fiction und Fantasyfilme - Ablehnung von Heimatfilmen.
- für die religiösen Volksmusik-Rentner: Altersdurchschnitt 65 Jahre - Vorliebe für Volksmusik - Ablehnung von Rock - meist verheiratete Rentner - meist politisch uninteressiert - religiös.

Ein wesentliches Ergebnis war, dass für das Bilden von Clustern zum einen das Alter und das Nettoeinkommen, zum anderen die Musik- und Fernsehpräferenzen eine große Rolle spielten, dass also sowohl soziodemographische als auch alltagsästhetische Variablen für die gesellschaftliche „Gruppierung“ maßgebend sind.
An der Charakterisierung der Gruppen wird auch ein wesentlicher Vorteil der Cluster-Analyse deutlich: hätte man ein Verfahren verwendet, bei dem man die Gruppen vorab schon definieren muss, hätte man niemals diese Gruppenstruktur gefunden, da

es unmöglich ist, solche Gruppen zu „erraten“. Aber auch ein Problem der Cluster-Analyse wird hieran deutlich: auch bei einer guten Clusterung ist es nicht immer einfach, die gefundenen Cluster treffend und aussagekräftig zu bezeichnen, sodass sofort klar ist, welche Art von Objekten (bzw. in diesem Fall Menschen) in diesem Cluster zu finden ist (wer sind wohl „extrem Unextreme“?... klar ist, dass die betreffenden Personen wohl in irgendeiner Weise durchschnittlich sind, aber auf welche Attribute bezieht sich das?).
Im Rahmen dieses Projekts wurden auch noch weitere interessante Ergebnisse gefunden, die anschließend von der Stadt Augsburg und von ortsansässigen Betrieben weiterverwendet wurden. So fand man z.B. heraus, dass für ein positives Wohnerlebnis in erster Linie die Umweltsituation bzw. die ruhige Lage, Freizeitmöglichkeiten und Fahrradwege in der Nähe des Wohnviertels wichtig sind und erst in zweiter Linie das soziale Umfeld, etwa ein hoher Ausländeranteil, eine Rolle spielen.

(M.Hilbert, D.Steinhübl: Lebensstile in der Stadt. Eine empirische Studie am Beispiel Augsburgs. Praxis Sozialforschung, hg. v. J.Cromm und H.Giegler, Band 2, München/Mering, 1998.
www.presse.uni-augsburg.de/unipress/up19992&3/artikel_27.html)

2.2 Grundsätzliches zur Cluster-Analyse

2.2.1 Terminologie

Wie auch beim Begriff *Data Mining* ist die Terminologie beim Thema *Clustering* nicht eindeutig. Vor allem in der älteren Literatur wird unter den verschiedensten Bezeichnungen Bezug auf dieses Verfahren genommen: *Cluster-Analyse, Q-Analyse, Typologie, Klassifikation, numerische Taxonomie, Gruppierung* oder *unüberwachte Mustererkennung.* Diese unterschiedliche Nomenklatur rührt wohl daher, dass dieses Verfahren schon seit langem in den verschiedensten Bereichen wie Psychologie, Zoologie, Botanik, Soziologie, künstliche Intelligenz etc. zu Hause ist und jeder dieser Bereiche „seinen“ Begriff dafür geprägt hat [Eve74]. Problematisch sind solche Begriffe wie *Klassifikation*, die früher als Synonym für *Cluster-Analyse* verwendet wurden, die heute aber ein ganz anderes Verfahren, nämlich das in Kapitel 1.4.1 beschriebene Verfahren des überwachten Lernens, bezeichnen. In der neueren Literatur findet man jedoch fast nur noch die Begriffe *Clustering* bzw. *Cluster-Analyse* (im Englischen *clustering* bzw. *cluster analysis*).

2.2.2 Entwicklung

Wie eben schon angedeutet, ist die Cluster-Analyse keine neue Disziplin, sondern eine schon lange existente, die in den letzten Jahren durch den Data Mining-Boom an Aufmerksamkeit und Bedeutung gewonnen hat. Dass sie schon früh entwickelt wurde, liegt wohl an der Natur des Menschen, der immerzu bestrebt ist, in seiner Umwelt ähnliche Dinge zu einer Gruppe zusammenzufassen (z.B. Dackel, Pudel, Schäferhund etc. wird zu der Gruppe „Hund“ zusammengefasst), um einerseits mit der alltäglichen Reizüberflutung fertig zu werden und um andererseits seine Umwelt besser verstehen zu können. Schon die alten Griechen und Römer entwickelten anhand physikalischer

Eigenschaften der Menschen einige Typologien, die die Menschen in Gruppen (z.B. unterschiedlichen Temperaments) unterteilten. Ein sehr bekanntes Beispiel ist wohl auch die von Linne im 18. Jahrhundert entwickelte Klassifikation der Tier- und Pflanzenwelt. Zu dieser Zeit war das Finden von Typologien aber noch eher eine Kunst als ein wissenschaftliches Verfahren. Nach und nach wurden von der subjektiven Beurteilung eines einzelnen Menschen unabhängige Verfahren entwickelt, welche jedoch einen enormen Rechenaufwand mit sich brachten. Darum war die Erfindung des Computers natürlich von wesentlicher Bedeutung für die Weiterentwicklung der Cluster-Analyse. In den 60er und 70er Jahren wurden die verschiedensten Algorithmen entwickelt, von denen viele heute noch oft Verwendung finden. Damals waren es vor allem die verschiedenen Wissenschaften (Psychologie, Biologie, Soziologie etc.), die die Entwicklung der Cluster-Analyse vorantrieben. Dies änderte sich in den 90er Jahren, als der Preisverfall bei den Speichermedien und die Entwicklung immer besserer Prozessoren das Speichern und das schnelle Bearbeiten großer Datenmengen ermöglichte und somit das Interesse von Industrie und Wirtschaft an Data Mining und damit indirekt auch an der Cluster-Analyse weckten.

2.2.3 Anwendung

Die Anwendungsbereiche der Cluster-Analyse lassen sich grob in zwei Gebiete aufteilen:

- Cluster-Analyse als eigenständige Anwendung zur Strukturierung der Daten
- Cluster-Analyse als Vorverarbeitungsschritt für andere Algorithmen

Beispiele für die Anwendung von Cluster-Analyse in den verschiedensten Bereichen gibt es wohl endlos viele. Einige wichtige sind in Kapitel 1.4.3 aufgelistet.

2.2.4 Notwendigkeit von Clustering-Verfahren

Bei einer Cluster-Analyse sollen Daten optimal[1] in Gruppen eingeteilt werden. Da ist natürlich der naive Ansatz nahe liegend, aus allen möglichen Gruppierungen der Daten die beste auszusuchen (sog. *Methode der totalen Enumeration*). Betrachtet man jedoch die Anzahl der möglichen Gruppierungen, so wird sofort klar, dass dieser Ansatz schon bei sehr kleinen Datenmengen zu unvertretbarem Rechenaufwand führt:

Definition 2.2.1 (Stirlingsche Zahlen) *Sei $\mathcal{M} = \{m_1, m_2, \ldots, m_n\}$ eine nichtleere n-elementige Menge und $\mathcal{C}$ eine Zerlegung von $\mathcal{M}$ in k disjunkte Teilmengen (Cluster), deren Vereinigung $\mathcal{M}$ ergibt. Die Anzahl $S(n,k)$ der bei k Teilmengen möglichen Partitionen $\mathcal{C}$ von $\mathcal{M}$ heißen* Stirlingsche Zahlen zweiter Art.

Lemma 2.2.2 *Für die Stirlingschen Zahlen gelten die folgenden rekursiven Beziehungen*

$$S(n+1,k) = S(n,k-1) + k \cdot S(n,k)$$

[1] Optimal bedeutet in diesem Zusammenhang, dass ein vom jeweiligen Algorithmus abhängiges Optimalitätskriterium möglichst gut erfüllt sein soll

$$mit\ S(n,1) := 1,\ S(n,n) := 1,\ und\ S(n,k) := 0\ für\ n < k$$

$$S(n+1,k+1) = \sum_{i=0}^{n} \binom{n}{i} S(i,k) \ = \ \sum_{i=k}^{n} \binom{n}{i} S(i,k)$$

Möchte man also beispielsweise 25 Datensätze in 5 Cluster unterteilen, so muss man nach dem Ansatz der totalen Enumeration $S(25,5) = 2,437 \cdot 10^{15}$ Partitionen betrachten! Ist die Anzahl der Cluster nicht unbedingt vorherbestimmt, so ist die Zahl der zu betrachtenden Gruppierungen noch wesentlich größer, nämlich

$$B(n) = \sum_{i=1}^{n} S(n,i) \text{ (die sog. } \textit{Bellschen Zahlen} \text{ oder } \textit{Exponentialzahlen})$$

in diesem Fall also $B(25) = 4,639 \cdot 10^{17}$.
Würde die Berechnung des Optimalitätskriteriums für eine Partition eine Mikrosekunde dauern, so würde man zum Auffinden des Optimums bei $n = 50$ und $k = 3$ rund $4 \cdot 10^{12}$ Jahre benötigen!
Bei der Cluster-Analyse wird eine große Anzahl überflüssiger, schlechter Clusterungen ausgeschlossen und das Optimum aus einer geringeren Zahl besserer Gruppierungen bestimmt, sodass das Verfahren mit einem realistischen Rechenaufwand durchführbar wird. Der „Preis“, den man dafür bezahlt, ist die Tatsache, dass das mittels Cluster-Analyse gefundene Optimum meist nur ein lokales Optimum ist. Dies ist oft aber nicht so gravierend, da einerseits das globale Optimum häufig nur wenige Prozent besser ist und andererseits nicht zwangsläufig die beste Clusterung benötigt wird, sondern bereits eine sehr gute Clusterung ausreicht [SL77].

2.2.5 Wichtige Begriffe und Definitionen

In den vorangegangenen Abschnitten wurden Begriffe verwendet, deren Bedeutung dem Leser wahrscheinlich intuitiv klar ist. Um sicher zu gehen, dass alle darunter das Gleiche verstehen und um spätere Verwirrung zu vermeiden, sollen diese Begriffe hier noch einmal formal definiert werden. Des weiteren werden noch einige Definitionen gegeben, die für die weiteren Abschnitte grundlegend sind.

Datenmatrix, Objekte, Attribute

Grundlage jeder Cluster-Analyse ist eine Menge $\mathcal{M} = \{m_1, m_2, \ldots, m_n\}$ von $n \in \mathbb{N}$ *Objekten.* Diese Objekte werden oft auch als *Datensätze* oder *Records* bezeichnet. Jeder dieser Datensätze m_i besteht aus p Werten, wobei jeder Wert die Ausprägung eines bestimmten *Merkmals* darstellt. So könnte z.B. ein Merkmal „Farbe“ sein und die Auspägungen ROT, GELB, GRÜN und BLAU haben. Auch für den Begriff *Merkmal* gibt es synonym verwendete Ausdrücke: *Attribut, Variable, Feld.* Analog dazu werden für die verschiedenen Ausprägungen eines Merkmals die Begriffe *Merkmals-, Attribut-, Variablen-,* oder *Feldwert* verwendet.
In der Regel werden die Datensätze in einer Matrix angeordnet:

Definition 2.2.3 (Datenmatrix) *Die $(n \times p)$-Matrix der n Datensätze mit ihren jeweils p Attributwerten,*

$$X = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1j} & \cdots & x_{1p} \\ \vdots & \vdots & & \vdots & & \vdots \\ x_{i1} & x_{i2} & \cdots & x_{ij} & \cdots & x_{ip} \\ \vdots & \vdots & & \vdots & & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nj} & \cdots & x_{np} \end{pmatrix}$$

heißt Datenmatrix. *In der i-ten Zeile dieser Matrix (i=1,...,n) stehen die p Attributwerte des i-ten Datensatzes und in der j-ten Spalte (j=1,...,p) stehen die n (nicht notwendig verschiedenen) Werte, die das j-te Attribut auf den n Datensätzen annimmt.*

Zur Notation:
Im Folgenden bezeichne $X_i := (x_{i1}, x_{i2}, \ldots, x_{ip})^T$ den p-dimensionalen Attributvektor des i-ten Datensatzes. Da der Datensatz m_i durch diesen Vektor eindeutig bestimmt ist, wird auch er zur Vereinfachung mit diesem Vektor identifiziert (also $m_i = X_i$). Das j-te Attribut sei im Folgenden mit A_j bezeichnet.

Attribut-Typen

Bei den Attributen ist es wichtig, die verschiedenen Typen zu kennen, die auftreten können, da der Typ die Wahl des Ähnlichkeitsmaßes mitbestimmt. Auf der obersten Ebene werden *qualitative* und *quantitative Attribute* unterschieden, welche sich jeweils noch einmal in „Untertypen“ aufspalten:

ATTRIBUTE				
quantitativ		qualitativ		
stetig	diskret numerisch	binär	kategorisch	ordinal

Die einzelnen Typen sind folgendermaßen charakterisiert:

Definition 2.2.4 (Qualitatives Attribut) *Ein Attribut A_j heißt* qualitativ*, wenn auf seinem Wertebereich die Operation „=“ bzw. $\neq$ “ definiert ist, d.h. wenn man für je zwei Werte von A_j überprüfen kann, ob sie übereinstimmen oder nicht.*

Beispiele für qualitative Attribute sind Farbe, Automarke, Geschlecht, Versicherungsklasse, etc. Ihr Wertebereich wird oft durch eine Teilmenge von $\mathbb{N}$ repräsentiert. Zum Beispiel könnte man für das Attribut „Farbe“ die Werte ROT, GELB, GRÜN und BLAU darstellen als 1, 2, 3 und 4. Bei dieser Verschlüsselung kann auch eine auf dem Wertebereich vorhandene Ordnung beibehalten werden, wie dies beispielsweise bei der Verschlüsselung der Werte SCHWACH, MITTEL und STARK des Attributs „Graduierung“ durch 1, 2 und 3 der Fall ist (je größer die Zahl, desto stärker die Graduierung).

Qualitative Attribute werden noch einmal unterteilt in *binäre*, *ordinale* und *kategorische Attribute*.

- *Ordinales Attribut*

  Ein ordinales Attribut ist dadurch charakterisiert, dass auf seinem Wertebereich zusätzlich ein Ordnung definiert ist. Es ist also nicht nur die Operation „= “ definiert, sondern auch die Operation „< “. Zwei Werte können miteinander verglichen werden, jedoch kann man nicht messen, wie groß der Unterschied der beiden Werte ist. So kann man beispielsweise bei einem Attribut „Zufriedenheit“, das Werte zwischen 1 (= sehr zufrieden) und 5 (= sehr unzufrieden) annehmen kann, sagen, dass eine 1 größere Zufriedenheit bedeutet als eine 4. Man kann jedoch nicht sagen, dass man bei einer 1 „um 3 zufriedener“ ist als bei einer 4.

- *Kategorisches Attribut*

  Im Gegensatz zum ordinalen Attribut ist auf dem Wertebereich eines kategorischen Attributs keine Ordnung definiert. Zwei Werte können also nur auf Übereinstimmung überprüft und nicht in ein Verhältnis zueinander gesetzt werden. Ein Spezialfall der kategorischen Attribute sind die binären Attribute. Auch ordinale Attribute können als kategorisch betrachtet werden, jedoch wird dann die auf ihnen definierte Ordnung nicht berücksichtigt.

- *Binäres Attribut*

  Ein binäres Attribut nimmt nur zwei Werte an, die im Allgemeinen durch 0 und 1 repräsentiert werden. Oft geben sie an, ob eine bestimmte Eigenschaft vorhanden (= 1) oder nicht vorhanden (= 0) ist. Jedes nicht-binäre Attribut mit k verschiedenen Merkmalsausprägungen kann in k binäre Attribute umgewandelt werden, indem man es als eine Folge von Abfragen auf das Vorhandensein der einzelnen Ausprägungen modelliert. So kann also z.B. das obige Farbattribut umgewandelt werden in die Attribute rot(1)/nicht-rot(0), gelb(1)/nicht-gelb(0), grün(1)/nicht-grün(0) und blau(1)/nicht-blau(0). Eine eventuell vorhandene Ordnung geht bei einer solchen Umwandlung allerdings verloren.

Häufig werden kategorische und ordinale Attribute unter dem Begriff *mehrstufige Attribute* zusammengefasst.

Definition 2.2.5 (Quantitatives Attribut) *Ein Attribut A_j heißt* quantitativ*, wenn auf seinem Wertebereich alle arithmetischen Rechenoperationen definiert sind.*

Insbesondere kann man bei einem quantitativen Attribut die Differenz je zweier Werte seines Wertebereichs bestimmen. So macht es etwa bei dem Attribut „Länge“ (in cm) durchaus Sinn, zu sagen, dass etwas, das 20 cm lang ist, um 5 cm länger ist als etwas, das 15 cm lang ist. Weitere Beispiele für quantitative Attribute sind Körpergröße, Temperatur, Lebensdauer und Fahrtzeit. Sie werden noch einmal unterteilt in *stetige* und *diskret numerische* Attribute: bei diskret numerischen Attributen besteht der Wertebereich aus abzählbar vielen Alternativen (häufig $\mathbb{N}$, z.B. bei Anzahl Kinder, Lebensdauer in Jahren), wohingegen stetige Attribute überabzählbar viele Alternativen annehmen können (Körpergröße, Länge eines Telefonats, etc.).

2.3 Ähnlichkeits- und Distanzmaße

Wie eingangs schon erläutert zeichnet sich eine gute Clusterung unter anderem dadurch aus, dass die Datensätze innerhalb eines Clusters möglichst ähnlich, Datensätze unterschiedlicher Cluster aber möglichst unähnlich sind. Hierzu ist es natürlich erforderlich, diese Ähnlichkeit zu messen. In der Praxis werden verschiedene Ähnlichkeitsmaße verwendet, von denen keines besser oder schlechter als das andere ist. Ein Maß kann aber sehr wohl den gegebenen Daten und insbesondere dem gegebenen Problem angemessener sein als das andere. Des weiteren hängt die Wahl des Ähnlichkeitsmaßes natürlich von den Attributtypen ab, die in den Datensätzen vorkommen.
Im ersten Abschnitt wird Ähnlichkeit bzw. Distanz formal definiert, bevor dann in den weiteren Abschnitten auf die verschiedenen Maße genauer eingegangen wird.

2.3.1 Definitionen

In der Literatur findet man verschiedene Definitionen für das Ähnlichkeitsmaß. Meist unterscheiden sie sich aber nur in der Festlegung des Wertebereichs: bei vielen ist der Wertebereich das Einheitsintervall (z.B. [Boc74]), bei manchen aber auch ein beliebiges Intervall $[a, b] \subset \mathbb{R}$ (z.B. [SL77]). Hier soll es das Einheitsintervall sein.

Definition 2.3.1 (Ähnlichkeitsmaß) *Eine reellwertige Funktion s:* $\mathcal{M} \times \mathcal{M} \longrightarrow \mathbb{R}$ *heißt* Ähnlichkeitsmaß *auf der Menge* $\mathcal{M}$*, mit der verkürzten Schreibweise* $s_{ij} := s(i, j)$*, wenn für* $1 \leq i, j \leq n$ *folgende Axiome erfüllt sind:*

1. $0 \leq s_{ij} \leq 1$
2. $s_{ii} = 1$
3. $s_{ij} = s_{ji}$

Sind zusätzlich die Bedingungen

4. $s_{ij} = 1 \Rightarrow X_i = X_j$
5. $|s_{ij} + s_{jk}|s_{ik} \leq s_{ij}s_{jk}$

erfüllt, so spricht man von einem metrischen Ähnlichkeitsmaß*.*
Die Werte s_{ij} *heißen* Ähnlichkeitskoeffizienten *oder* Ähnlichkeiten*.*

In der Regel werden das vierte und das fünfte Axiom nicht gefordert, viele der bei den Clustering-Algorithmen verwendeten Ähnlichkeitsmaße erfüllen sie aber. Mit Hilfe des vierten Axioms kann man maximal ähnliche Objekte (z.B. Patienten, die dieselben Krankheitssymptome haben) zu einer Äquivalenzklasse zusammenfassen. Das fünfte Axiom entspricht der Dreiecksungleichung bei Distanzmaßen (siehe Kap. 2.3.3).

Definition 2.3.2 (Ähnlichkeitsmatrix) *Die* $(n \times n)$*-Matrix*

$$S = \begin{pmatrix} 1 & s_{12} & \cdots & s_{1n} \\ s_{21} & 1 & \cdots & s_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ s_{n1} & s_{n2} & \cdots & 1 \end{pmatrix}$$

heißt Ähnlichkeitsmatrix. *Wegen Axiom 3 ist sie symmetrisch und wegen Axiom 2 sind alle Diagonalelemente gleich 1.*

Oft ist es einfacher, anstatt der Ähnlichkeit zweier Objekte deren Unähnlichkeit bzw. Distanz anzugeben (z.B. wenn der räumliche Abstand zweier Punkte deren (Un-)Ähnlichkeit bestimmt). Ganz analog zur Ähnlichkeit definiert man hierfür die Unähnlichkeit bzw. Distanz zwischen Datensätzen:

Definition 2.3.3 (Distanzmaß) *Eine reellwertige Funktion d:* $\mathcal{M} \times \mathcal{M} \longrightarrow \mathbb{R}_+$ *heißt* Distanzmaß *auf der Menge* $\mathcal{M}$*, mit der verkürzten Schreibweise* $d_{ij} := d(i, j)$*, wenn für* $1 \leq i, j \leq n$ *folgende Axiome erfüllt sind:*

1. $d_{ij} \geq 0$
2. $d_{ii} = 0$
3. $d_{ij} = d_{ji}$

Sind zusätzlich die Bedingungen

4. $d_{ij} = 0 \Rightarrow X_i = X_j$
5. $d_{ik} \leq d_{ij} + d_{jk}$

erfüllt, so handelt es sich um ein metrisches Distanzmaß *oder einfach um eine* Metrik. *Die Werte* d_{ij} *heißen* Distanz *oder* Abstand.

$d_{ij} = 0$ bedeutet, dass X_i und X_j minimalen Abstand haben. Ebenfalls analog zu oben werden bei Cluster-Analyse-Verfahren keine metrischen Distanzmaße gefordert, die meisten von den Algorithmen verwendeten Maße erfüllen diese zusätzlichen Axiome aber. Metrische Distanzmaße entsprechen dem räumlichen Abstandsbegriff und erlauben es, zur Gruppierung von Objekten die geometrische Anschauung zu Hilfe zu nehmen [Boc74].

Definition 2.3.4 (Distanzmatrix) *Die* $(n \times n)$*-Matrix*

$$D = \begin{pmatrix} 0 & d_{12} & \cdots & d_{1n} \\ d_{21} & 0 & \cdots & d_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ d_{n1} & d_{n2} & \cdots & 0 \end{pmatrix}$$

heißt Distanzmatrix. *Wegen Axiom 3 ist sie symmetrisch und wegen Axiom 2 sind alle Diagonalelemente gleich 0.*

Distanz- und Ähnlichkeitsmaße lassen sich sehr leicht ineinander umwandeln. Hierzu benötigt man lediglich eine geeignete *Transformationsfunktion.*

Satz 2.3.5 *Ist* $d : \mathcal{M} \times \mathcal{M} \to \mathbb{R}_+$ *ein Distanzmaß gemäß Definition (2.3.3) und* $f : \mathbb{R} \to \mathbb{R}$ *eine nicht-negative, monoton fallende Funktion mit* $f(0) = 1$*, so definiert* $s := f(d)$ *ein Ähnlichkeitsmaß, das die Axiome (1)-(3) aus Defintion (2.3.1) erfüllt.*

Häufig verwendete Transformationen sind [SL77]:

$$s := \frac{1}{1 + ad^2} \quad \text{, wo} \quad a = \max_{i \neq j} d_{ij}$$

$$s := 1 - \frac{d}{a} \quad \text{, wo} \quad a = \max_{i \neq j} d_{ij}$$

$$s := \frac{(H - d^2)}{H + d^2} \quad \text{, wo} \quad \text{H:= mittlere Distanz}$$

Satz 2.3.6 *Ist* $s : \mathcal{M} \times \mathcal{M} \to [0,1]$ *ein Ähnlichkeitsmaß gemäß Definition (2.3.1) und* $g : \mathbb{R} \to \mathbb{R}$ *eine nicht-negative, monoton fallende Funktion mit* $g(1) = 0$*, so definiert* $d := g(s)$ *ein Distanzmaß, das die Axiome (1)-(3) aus Definition (2.3.3) erfüllt.*

Häufig verwendete Transformationen sind [SL77]:

$$d := 1 - s \qquad d := \sqrt{1 - s}$$

$$d := 1 - s^2 \qquad d := \sqrt{1 - s^2}$$

$$d := \frac{1}{s} - 1 \qquad d := -\log s$$

Fast alle Algorithmen arbeiten nicht mit den so genannten Rohdaten der Datenmatrix, sondern mit den Daten in der Ähnlichkeits- bzw. Distanzmatrix. Diese muss natürlich zuerst mittels geeigneter Ähnlichkeits- bzw. Distanzmaße aus den Rohdaten berechnet werden. Da diese Matrix symmetrisch ist und alle Diagonalelemente gleich 1 bzw. 0 sind, müssen anstatt n^2 Werten nur $n(n\text{-}1)/2$ Werte berechnet werden. Meist haben die Rohdaten mehrere bzw. viele Attribute unterschiedlichen Typs. Bei der Berechnung der Ähnlichkeiten/Distanzen muss dies berücksichtigt werden, d.h. für die verschiedenen Attribut-Typen müssen unterschiedliche Maße gewählt werden. Im Folgenden werden nun die gebräuchlichsten kurz vorgestellt.

2.3.2 Ähnlichkeit und Distanz bei quantitativen Attributen

In diesem Abschnitt gilt die Annahme, dass die betrachtete Datenmenge $\mathcal{M} = \{X_1, \ldots, X_n\}$ nur quantitative Attribute besitzt. Dies bedeutet insbesondere, dass man die einzelnen Datensätze als Punkte bzw. Vektoren im $\mathbb{R}^p$ auffassen kann, wobei p die Anzahl der Attribute ist.

Die euklidische Distanz

Fasst man die Objekte als Punkte im p-dimensionalen reellen Raum auf, so ist es naheliegend, als Distanz den euklidischen Abstand zu nehmen:

$$d_{ij} := \|X_i - X_j\| := \sqrt{\sum_{k=1}^{p} (x_{ik} - x_{jk})^2}$$

(Zur Notation: x_{ij} bezeichne den Wert des j-ten Attributs für X_i).

Die euklidische Distanz ist metrisch (erfüllt also die Axiome (4) und (5) aus Definition (2.3.3)), translationsinvariant und invariant bezüglich orthogonaler linearer Transformationen (also Drehungen und Spiegelungen) der X_i (i=1,...,n). Sie ändert sich also nicht unter Abbildungen der Art

$$X_i \mapsto CX_i + a \qquad , \ i = 1, \ldots, n$$

wo $a \in \mathbb{R}^p$ ein beliebiger Vektor und $C \in \mathbb{R}^{p \times p}$ eine orthogonale Matrix ist.
Jedoch ist dieses Distanzmaß nicht skaleninvariant, darum ist es wichtig, dass die gewählten Einheiten zur Messung der p Attribute vergleichbar sind. Ist dies nicht der Fall, so müssen die Attributwerte vor der Berechnung der Distanzen zuerst noch normiert werden, beispielsweise indem man die Wertebereiche der Attribute auf das Einheitsintervall abbildet und die Attributwerte entsprechend umrechnet.

Die L_r-Distanzen

Die L_r-Distanzen stellen eine Verallgemeinerung der euklidischen Distanz dar. Sie sind folgendermaßen definiert:

$$d_{ij}^{(r)} := \sqrt[r]{\sum_{k=1}^{p} |x_{ik} - x_{jk}|^r} \qquad , \ r \in \mathbb{N}$$

Für r=2 erhält man die euklidische Distanz. Auch die L_1-Distanz, die so genannte *Manhattan-, City-Block-,* oder *Taxifahrer-Distanz*, spielt in der Cluster-Analyse eine wichtige Rolle.
Für jedes $r \in \mathbb{N}$ ist die L_r-Distanz metrisch. Mit wachsendem r fallen die größeren Differenzen zwischen den Werten eines Attributs immer mehr ins Gewicht, wohingegen kleine Differenzen an Bedeutung verlieren, wenn r größer wird. Für $r \to \infty$ erhält man als Grenzfall

$$d_{ij}^{\infty} = \max_k |x_{ik} - x_{jk}|,$$

wobei diese L_∞-Distanz in der Cluster-Analyse eine untergeordnete Rolle spielt, da sie Ausreißern ein starkes Gewicht gibt.
L_r-Distanzen sind translationsinvariant, allerdings sind sie weder skaleninvariant noch invariant bezüglich orthogonaler Transformation (außer für $r = 2$). Die Daten müssen also vor der Berechnung der Distanzen geeignet normiert werden.

Die Mahalanobis-Distanz

Durch eine andere Verallgemeinerung der euklidischen Distanz erhält man die Mahalanobis-Distanz:
Sei $n > p$ und $\bar{X} := (\bar{x}_{\cdot 1}, \ldots, \bar{x}_{\cdot p}) := \frac{1}{n} \sum_{k=1}^{n} X_k$ der Mittelwertvektor von $X_1, \ldots, X_n$.

Sei weiter

$$K := (k_{\nu\mu}) := \frac{1}{n}\sum_{j=1}^{n}(X_j - \bar{X})(X_j - \bar{X})^T$$

die empirische $p \times p$-Kovarianzmatrix zu $X_1, \ldots, X_n$ mit den Elementen

$$k_{\nu\mu} := \frac{1}{n}\sum_{j=1}^{n}(x_{j\nu} - \bar{x}_{\cdot\nu})(x_{j\mu} - \bar{x}_{\cdot\mu}) \qquad , \ \nu, \mu = 1, \ldots, p$$

Dann ist die Mahalanobis-Distanz definiert durch

$$d_{ij}^2 := (X_i - X_j)^T K^{-1}(X_i - X_j) = \sum_{\nu=1}^{p}\sum_{\mu=1}^{p} k^{\nu\mu}(x_{i\nu} - x_{j\nu})(x_{i\mu} - x_{j\mu})$$

wobei $K^{-1} := (k^{\nu\mu})$ die Inverse der Kovarianzmatrix ist.
Die Mahalanobis-Distanz ist translationsinvariant und invariant bezüglich aller linearer nicht-singulärer Transformationen der Datensätze, d.h. sie ändert sich nicht unter Abbildungen der Art

$$X_i \mapsto CX_i + a \qquad , \ i = 1, \ldots, n$$

wobei $a \in \mathbb{R}^p$ und $C \in \mathbb{R}^{p \times p}$ mit $\det(C) \neq 0$.
Des weiteren ist diese Distanz skaleninvariant, die Daten brauchen also nicht wie bei den L_r-Distanzen normiert zu werden. Ein weiterer Vorteil dieses Maßes ist die Tatsache, dass etwaige Korrelationen zwischen den Attributen eliminiert werden, da für die Berechnung der Distanz zwischen X_i und X_j nicht nur diese beiden Datensätze herangezogen werden, sondern alle Datensätze in die Berechnung mit einfließen. Daher spielt die Mahalanobis-Distanz bei quantitativen Attributen eine große Rolle.

2.3.3 Ähnlichkeit und Distanz bei binären Attributen

In diesem Abschnitt gilt die Annahme, dass die betrachtete Datenmenge $\mathcal{M} = \{X_1, \ldots, X_n\}$ nur binäre Attribute besitzt und die zwei möglichen Werte jedes Attributs durch 0 bzw. 1 repräsentiert werden, wobei $x_{ij} = 1$ $(= 0)$ bedeuten soll, dass Attribut A_j bei Objekt X_i (nicht) vorhanden ist.
Fast alle Ähnlichkeitsmaße für binäre Daten beruhen ausschließlich auf den Auftretenshäufigkeiten der vier möglichen Kombinationen der Attributwerte (1-1, 1-0, 0-1, 0-0), die sich ergeben, wenn man zwei Objekte X_i, X_j in einem Attribut A_k vergleicht:

		X_j		
		1	0	
X_i	1	a_{ij}	b_{ij}	$a_{ij} + b_{ij}$
	0	c_{ij}	d_{ij}	$c_{ij} + d_{ij}$
		$a_{ij} + c_{ij}$	$b_{ij} + d_{ij}$	$p = a_{ij} + b_{ij} + c_{ij} + d_{ij}$

Zur Vereinfachung werden im Folgenden die Abkürzungen $a := a_{ij}$, $b := b_{ij}$, $c := c_{ij}$, $d := d_{ij}$ verwendet.

Die verschiedenen Ähnlichkeitsmaße, die man in der Literatur findet, unterscheiden sich meist nur darin, ob und wie Übereinstimmung in einem gemeinsam vorhandenen Attribut(1-1), Nichtübereinstimmung (1-0, 0-1) und Übereinstimmung in einem gemeinsam nicht vorhandenen Attribut (0-0) gewichtet werden. Sie sind meist Spezialfälle der Funktionenschar

$$s_{ij}^{\delta\lambda} = \frac{a + \delta d}{a + \delta d + \lambda(b + c)}$$

wobei $\lambda > 0$ und $0 \leq \delta \leq 1$ geeignet zu wählende Gewichtsfaktoren darstellen. $\delta = 0$ bedeutet beispielsweise, dass das gemeinsame Nicht-Vorhandensein eines Attributs nicht berücksichtigt wird, wohingegen es für $\delta = 1$ voll berücksichtigt wird. Der Faktor λ gewichtet je nach Wert die Übereinstimmung ($\lambda < 1$) bzw. die Nicht-Übereinstimmung ($\lambda > 1$) stärker.
Neben den allgemeinen Forderungen an ein Ähnlichkeitsmaß (siehe Kap. 2.3.1) verlangt man bei Ähnlichkeitsmaßen für binäre Attribute oft zusätzlich, dass sie

- monoton mit d wachsen
- symmetrisch in b und c sind
- monoton mit b und c fallen

Jedoch erfüllen nicht alle Maße für binäre Attribute diese Forderungen.
Binäre Attribute können in zwei Typen unterschieden werden, was auch eine Unterteilung der auf sie angewendeten Ähnlichkeitsmaße impliziert[Boc74]:

Definition 2.3.7 *Ein Attribut A_k heißt* symmetrisch*, wenn die Aussage „A_k ist bei den Objekten X_i und X_j vorhanden“ (also $x_{ik} = x_{jk} = 1$) gleich viel über die Ähnlichkeit von X_i und X_j aussagt wie die Angabe „A_k ist bei den Objekten X_i und X_j nicht vorhanden“(also $x_{ik} = x_{jk} = 0$).*

Ein Beispiel für ein symmetrisches Attribut ist das Geschlecht (männlich/weiblich): hat ein Objekt die Eigenschaft „männlich“, dann sagt das ebensoviel über das Objekt aus, wie wenn es die Eigenschaft „nicht männlich“ hat, denn dann weiß man, dass es weiblich ist.

Definition 2.3.8 *Ein Attribut A_k heißt* unsymmetrisch*, wenn die Aussage „A_k ist bei den Objekten X_i und X_j vorhanden“($x_{ik} = x_{jk} = 1$) ein anderes Gewicht besitzt als die Aussage „A_k ist bei den Objekten X_i und X_j nicht vorhanden“($x_{ik} = x_{jk} = 0$).*

Ein unsymmetrisches Merkmal ist z.B. das Attribut „Farbe“ mit den Alternativen „rot“ und „nicht rot“: bei der Aussage „beide Objekte sind rot“ weiß man, dass sie beide die gleiche Farbe haben, wohingegen man aus der Aussage „beide Objekte sind nicht rot“ nicht auf die gleiche Farbe schließen kann.
Mit dieser Einteilung der Attribute kann man nun auch die Ähnlichkeitsmaße klassifizieren:

Definition 2.3.9 *Ein Ähnlichkeitsmaß s für n Objekte $X_1, \ldots, X_n$ mit p binären Attributen $A_1, \ldots, A_p$ heißt* (vertauschungs-)invariant*, wenn für alle p Attribute die Alternativen „0“ und „1“ symmetrisch behandelt werden, d.h. der Wert s_{ij} ändert sich nicht, wenn man für einzelne Attribute A_k (oder auch alle Attribute) „0“ und „1“ vertauscht, also x_{ik} und x_{jk} durch $1 - x_{ik}$ und $1 - x_{jk}$ ersetzt.*

Ein Ähnlichkeitsmaß ist insbesondere dann invariant, wenn sein Wert nur von der Anzahl $a+d$ der übereinstimmenden und der Anzahl $b+c$ der nicht übereinstimmenden Attribute abhängt.
Invariante Ähnlichkeitsmaße eignen sich gut zur Bearbeitung symmetrischer binärer Attribute; bei unsymmetrischen Attributen hingegen ist ihre Anwendung wenig sinnvoll. Nicht invariante Ähnlichkeitsmaße sollten immer dann verwendet werden, wenn manche oder auch alle Attribute unsymmetrisch sind.
Nach dieser groben Klassifizierung sollen nun einige konkrete Ähnlichkeitsmaße für binäre Attribute betrachtet werden.

Der M-Koeffizient

Das durch

$$s_{jk} := \frac{a+d}{p} = 1 - \frac{b+c}{p} = \frac{1}{1 + \frac{b+c}{a+d}} = 1 - \frac{\|X_i - X_j\|^2}{p}$$

definierte Ähnlichkeitsmaß für binäre Merkmale heißt *M-Koeffizient*; es gibt den relativen Anteil der übereinstimmenden Attribute von X_i und X_j an. Der M-Koeffizient hat folgende Eigenschaften:

- er ist invariant: die Alternativen „0“ und „1“ werden also gleich behandelt.
- er erfüllt die in der Einleitung dieses Abschnittes geforderten Zusatzbedingungen (Symmetrie in b und c, monotones Wachstum mit d, monotones Fallen mit b und c).
- $0 \leq s_{ij} \leq 1$
- $s_{ij} = 1$ genau für identische Attributvektoren $X_i,\ X_j$ (d.h. X_i und X_j stimmen in allen Attributen überein)
- $s_{ij} = 0$ genau für komplementäre Vektoren $X_i,\ X_j$ (d.h. man erhält X_j aus X_i, indem man jede 0 durch eine 1 ersetzt und umgekehrt)

Wegen des Nenners $p = (a+d)+(b+c)$ werden beim M-Koeffizienten die übereinstimmenden Attribute von X_i und X_j gleich gewichtet wie die nicht übereinstimmenden. In der Praxis kommt es jedoch häufig vor, dass man Übereinstimmungen stärker (oder auch schwächer) gewichten möchte als Nicht-Übereinstimmungen. Dies erzielt man, indem man in Zähler und Nenner den übereinstimmenden Attributen ein Gewicht w $(0 < w < 1)$ und den nicht übereinstimmenden das Gewicht $1 - w$ zuordnet. Man erhält somit den *modifizierten* bzw. *verallgemeinerten M-Koeffizienten*

$$s_{ij} := \frac{w(a+d)}{w(a+d) + (1-w)(b+c)}$$

mit $0 \leq s_{ij} \leq 1$.
Für $w = \frac{1}{2}$ erhält man den (ungewichteten) M-Koeffizienten.
Wichtige Spezialfälle sind

- das Ähnlichkeitsmaß von Rogers & Tamintoto (1960), bei dem die nicht übereinstimmenden Attribute im Nenner doppelt gewichtet werden:

$$w = \frac{1}{3} \quad , \qquad s_{ij} := \frac{a+d}{p+(b+c)} = \frac{a+d}{(a+d)+2(b+c)}$$

- das so genannte *erste Ähnlichkeitsmaß von Sokal & Sneath* (1963), bei dem die übereinstimmenden Attribute in Zähler und Nenner doppelt gewichtet werden:

$$w = \frac{2}{3} \quad , \qquad s_{ij} := \frac{2(a+d)}{2(a+d)+(b+c)} = 1 - \frac{b+c}{2p-(b+c)}$$

Auch der modifizierte M-Koeffizient erfüllt für alle w mit $0 \leq w \leq 1$ die allgemeinen Forderungen an ein Ähnlichkeitsmaß sowie die speziellen Forderungen an ein Maß für binäre Attribute.

Sind alle Attribute unsymmetrisch und zwar derart, dass die „1“ jeweils die aussagekräftigere der beiden Alternativen ist, so ist es sinnvoll, ein Ähnlichkeitsmaß zu wählen, das die Anzahl der übereinstimmenden „1“-Komponenten stärker gewichtet als die Anzahl der übereinstimmenden „0“-Komponenten. Dies geschieht beim *S-Koeffizienten*:

Der S-Koeffizient

Vernachlässigt man beim M-Koeffizenten in Zähler und Nenner die Anzahl a der übereinstimmenden "0"-Komponenten, so ergibt sich

$$s_{ij} := \frac{d}{p-a} = \frac{d}{d+b+c} = \frac{1}{1+\frac{b+c}{d}} \quad ,$$

der so genannte *S-Koeffizient.* Er hat folgende Eigenschaften:

- er ist nicht vertauschungsinvariant: Übereinstimmende „1“-Komponenten werden voll gewichtet, übereinstimmende „0“-Komponenten hingegen gar nicht.
- er erfüllt die in der Einleitung dieses Abschnittes geforderten Zusatzbedingungen (Symmetrie in b und c, monotones Wachstum mit d, monotones Fallen mit b und c).
- für $a < p$ ist $0 \leq s_{ij} \leq 1$
- $s_{ij} = 1$ genau für identische Attributvektoren $X_i,\ X_j$
- $s_{ij} = 0$ genau für komplementäre Attributvektoren $X_i,\ X_j$

Auch beim S-Koeffizienten gibt es die modifizierte Version, bei der die übereinstimmenden „1“-Komponenten das Gewicht w , $0 < w < 1$, und die übereinstimmenden „0“-Komponenten das Gewicht $1 - w$ bekommen. Man erhält somit

$$s_{ij} := \frac{wd}{wd + (1-w)(b+c)} = \frac{1}{1 + \frac{1-w}{w}\frac{b+c}{d}} \quad ,$$

den so genannten *modifizierten* oder *verallgemeinerten S-Koeffizienten.*
Wichtige Spezialfälle hiervon sind

- das Ähnlichkeitsmaß von Dice (1945), bei dem die übereinstimmenden „1“-Komponenten relativ zur Gesamtzahl der in den beiden Attributvektoren vorkommenden Einsen gemessen wird:

$$w = \frac{2}{3} \quad , \qquad s_{ij} := \frac{2d}{2d + (b+c)}$$

- das so genannte *zweite Ähnlichkeitsmaß von Sokal & Sneath,* bei dem nicht übereinstimmende Attribute im Nenner doppelt gewichtet werden:

$$w = \frac{1}{3} \quad , \qquad s_{ij} := \frac{d}{d + 2(b+c)} = 1 - \frac{2(b+c)}{d + 2(b+c)}$$

Auch der modifizierte S-Koeffizient erfüllt für alle w mit $0 \leq w \leq 1$ die allgemeinen Forderungen an ein Ähnlichkeitsmaß sowie die speziellen Forderungen an ein Maß für binäre Attribute.

2.3.4 Ähnlichkeit und Distanz bei mehrstufigen Attributen

Wie in Kapitel 2.2.5 schon bemerkt, lassen sich kategorische Attribute auf binäre Attribute abbilden, indem man jede Alternative des Attributs als eine Abfrage auf Vorhandensein dieser Alternative betrachtet. Nimmt man diese Transformation vor, so ist es anschließend möglich, die oben beschriebenen Ähnlichkeitsmaße für binäre Attribute darauf anzuwenden. Der Vorteil dieser Methode ist, dass sie nicht sehr schwierig durchzuführen ist und keine weiteren Anforderungen an die Datenqualität stellt. Jedoch hat diese Vorgehensweise den Nachteil, dass es – vor allem wenn ein Attribut viele Alternativen hat – zu Verzerrungen kommt. So wird beispielsweise bei Anwendung des M-Koeffizienten (siehe Kap. 2.3.3) die Ähnlichkeit um so größer, je mehr Alternativen ein transformiertes kategorisches Attribut hat. Es ist also sinnvoll, für mehrstufige Attribute spezielle Ähnlichkeitsmaße zu verwenden. Möchte man bei einem Attribut mit geordneten Alternativen, dass diese Ordnung bei der Berechnung der Ähnlichkeit berücksichtigt wird, so muss man für diese Attribute ein anderes Maß wählen als für kategorische Attribute.

Ähnlichkeitsmaße für kategorische Attribute

In diesem Abschnitt gilt die Annahme, dass die betrachtete Datenmenge $\mathcal{M} = \{X_1, \ldots, X_n\}$ nur kategorische Attribute $A_1, \ldots, A_p$ mit den Alternativen $A_{k1}, \ldots, A_{ka_k}$ $(k = 1, \ldots, p)$ besitzt.

Beim ***verallgemeinerten M-Koeffizienten für kategorische Attribute*** wird die Anzahl u_{ij} der übereinstimmenden Attribute relativ zur Gesamtzahl p aller Attribute gemessen:

$$s_{ij} := \frac{u_{ij}}{p}$$

Der verallgemeinerte M-Koeffizient hat folgende Eigenschaften:

- $0 \leq s_{ij} \leq 1$
- $s_{ij} = 0$ genau dann, wenn keine Komponente bei X_i und X_j übereinstimmt
- $s_{ij} = 1$ genau dann, wenn X_i und X_j in allen Komponenten übereinstimmen
- er ist unabhängig von der Reihenfolge der Alternativen eines Attributs.

Wie beim M-Koeffizienten für binäre Attribute kann man auch hier ein Gewicht w, $0 < w < 1$, für die Anzahl u_{ij} der übereinstimmenden Attribute und das Gewicht $1 - w$ für die Anzahl v_{ij} der nicht übereinstimmenden Attribute einfügen:

$$s_{ij} := \frac{wu_{ij}}{wu_{ij}(1-w)v_{ij}} = \frac{w}{(2w-1) + (1-w)\frac{p}{u_{ij}}} \leq 1$$

Der verallgemeinerte M-Koeffizient ist unabhängig von den Anzahlen $a_1, \ldots, a_p$ der für die einzelnen Attribute verfügbaren Alternativen. Oft ist es aber wünschenswert, dass die Übereinstimmung von X_i und X_j in einem Attribut mit 50 Alternativen höher bewertet wird als die Übereinstimmung in einem Attribut mit 5 Alternativen. Dieser Forderung trägt das *Ähnlichkeitsmaß von Hyvärinen (1962)* Rechnung:

$$s_{ij} := \frac{1}{m} \sum_{k=1}^{p} a_i \cdot \delta(x_{ik}, x_{jk})$$

$$mit \;\; m := \sum_{i=1}^{p} a_i \qquad und \;\; \delta(u,v) := \begin{cases} 1 & : \quad u = v \\ 0 & : \quad u \neq v \end{cases}$$

Der Faktor $\frac{1}{m} \sum_{k=1}^{p} a_i$ bewirkt, dass die Übereinstimmung in einem Attribut umso stärker gewichtet wird, je mehr Alternativen das Attribut hat. Aufgrund der Definition von δ haben nicht übereinstimmende Komponenten das Gewicht 0, werden also nicht berücksichtigt. Ist die Anzahl a_k der Alternativen für alle Attribute $A_1, \ldots, A_p$ gleich, so stimmt dieses Maß bis auf den Faktor 1/p mit dem verallgemeinerten M-Koeffizienten überein.

Ähnlichkeitsmaße bei ordinalen Attributen

In diesem Abschnitt gilt die Annahme, dass die betrachtete Datenmenge $\mathcal{M} = \{X_1, \ldots, X_n\}$ nur diskret numerische Attribute $A_1, \ldots, A_p$ mit den geordneten Alternativen $A_{k1} \prec \ldots \prec A_{ka_k}$ $(k = 1, \ldots, p)$ besitzt.
Bei geordneten Alternativen sollte ein gutes Ähnlichkeitsmaß einen umso größeren Wert liefern je näher zwei Alternativen in der geordneten Folge zusammenstehen. Ein naheliegender Ansatz ist es, die für quantiative Attribute definierten Maße, wie z.B.

die euklidische Distanz (bzw. ein daraus abgeleitetes Ähnlichkeitsmaß), zu verwenden. Hierbei werden die einzelnen Werte x_{ij} in der Datenmatrix nicht als Zeichen für die jeweilige Alternative gelesen, sondern als Zahl. Da die einzelnen Attribute in der Regel unterschiedliche Wertebereiche besitzen und insofern nicht vergleichbar sind, müssen die Daten bei dieser Methode erst normiert werden, bevor eine Ähnlichkeit bzw. Distanz berechnet werden kann.
Häufig erweisen sich jedoch für diese Art von Attributen *probabilistische Ähnlichkeitsmaße* als sehr sinnvoll, da sie die willkürliche Deutung der Zahlen in der Datenmatrix als quantitative Daten vermeiden. Die genaue Beschreibung eines solchen Maßes würde den Rahmen dieser Arbeit sprengen, der interessierte Leser kann dies aber z.B. in [Boc74], S.72f, nachlesen.

2.3.5 Objekte mit Attributen unterschiedlichen Typs

In den vorigen Abschnitten wurden Fälle untersucht, bei denen die betrachteten Objekte jeweils Attribute eines einzigen Typs besitzen. Dies kommt in der Praxis aber sehr selten vor, in der Regel haben die Datensätze Attribute unterschiedlichen Typs. Um dennoch eine Ähnlichkeit zweier Datensätze angeben zu können, ist es naheliegend, aus den oben behandelten Maßen für die einzelnen Attributtypen ein so genanntes *aggregiertes Ähnlichkeitsmaß* abzuleiten.

Gewichteter Mittelwert der Attributgruppen Hierbei wird zur Berechnung der Ähnlichkeit zweier Objekte $X_i,\ X_j$ zunächst für die Gruppe der qualitativen und die Gruppe der quantitativen Attribute einzeln die Ähnlichkeit berechnet, indem man eines der oben erläuterten Ähnlichkeitsmaße verwendet. Anschließend wird hieraus ein gewichteter Mittelwert als Ähnlichkeit von X_i und X_j berechnet. Manchmal wird auch eine Linearkombination der Werte als aggregiertes Ähnlichkeitsmaß verwendet, wobei hierbei dann die Ähnlichkeit der Objekte größer als 1 werden kann.

Gewichteter Mittelwert der einzelnen Attribute Hier werden die Attribute gleichen Typs nicht zu einer Gruppe zusammengefasst, sondern es wird zunächst für jedes Attribut $A_l,\ l \in \{1, \ldots, p\}$, einzeln die Ähnlichkeit berechnet. Dies geht ebenfalls mit den oben beschriebenen Verfahren, indem man den Spezialfall eines einzigen Attributs betrachtet. Jedes Attribut $A_l,\ l \in \{1, \ldots, p\}$, hat also sein eigenes Ähnlichkeitsmaß s_{ij}^l, das beim Vergleich zweier Objekte $X_i = \{x_{i1}, \ldots, x_{ip}\},\ X_j = \{x_{j1}, \ldots, x_{jp}\}$ die Ähnlichkeit von x_{il} und x_{jl} mißt. Das gewichtete, aggregierte Ähnlichkeitsmaß mit Gewichten $w_l \geq 0$ ist dann definiert als

$$s_{ij} := \frac{1}{\sum_{l=1}^{p} w_l} \sum_{l=1}^{p} (w_l \cdot s_{ij}^l).$$

Anstatt die Ähnlichkeitsmaße für die einzelnen Attributtypen zu einem neuen zu aggregieren, kann man auch direkt ein neues Ähnlichkeitsmaß definieren, das die verschiedenen Attributtypen berücksichtigt. Dies geht zum Beispiel, indem man zunächst

folgende Werte definiert:

$$u_{ij} := \# \left\{ \begin{array}{c} \text{qualitative Attribute } A_l \\ \text{mit } x_{il} = x_{jl} \end{array} \right\} + \# \left\{ \begin{array}{c} \text{quantitative Attribute } A_l \\ \text{mit } |x_{il} - x_{jl}| \leq \alpha_1 \end{array} \right\}$$

$$v_{ij} := \# \left\{ \begin{array}{c} \text{qualitative Attribute } A_l \\ \text{mit } x_{il} \neq x_{jl} \end{array} \right\} + \# \left\{ \begin{array}{c} \text{quantitative Attribute } A_l \\ \text{mit } |x_{il} - x_{jl}| > \alpha_2 \end{array} \right\}$$

wobei α_1 und α_2 Ähnlichkeitsschranken sind mit $0 < \alpha_1 \leq \alpha_2 \leq 1$, welche besagen, dass zwei Objekte $X_i,\ X_j$ in einem Attribut A_l als ähnlich gelten, falls $|x_{il} - x_{jl}| \leq \alpha_1$ gilt, und dass sie als unähnlich gelten, falls $|x_{il} - x_{jl}| > \alpha_2$ gilt. Dabei kann $\alpha_1 = \alpha_2$ sein, es kann aber auch sein, dass es einen Bereich zwischen den beiden Schranken gibt, in dem weder auf Ähnlichkeit noch auf Unähnlichkeit geschlossen werden kann. u_{ij} (bzw. v_{ij}) zählt also die Anzahl der Merkmale, in denen die Objekte X_i und X_j als ähnlich (bzw. unähnlich) betrachtet werden. Mit Hilfe dieser Werte kann man beispielsweise folgende Ähnlichkeitsmaße definieren:

$$s_{ij} := \frac{u_{ij}}{p} \text{ , analog zum M-Koeffizienten (siehe S.41)}$$

$$s_{ij} := \frac{u_{ij}}{u_{ij} + v_{ij}}, \text{ in Anlehnung an das Maß von Rogers \& Tamintoto (siehe S.41)}$$

Wichtig ist bei all diesen Ähnlichkeitsmaßen - egal, ob aggregierte Maße oder für gemischte Attribute neu definierte Maße -, dass die quantitativen Attribute vorher normiert werden, da es sonst zu starken Verzerrungen kommen kann.

2.3.6 Ähnlichkeit und Distanz zwischen Clustern

Bei den bisher betrachteten Ähnlichkeits- bzw. Distanzmaßen wird immer ein Paar von Objekten verglichen. Bei der Cluster-Analyse jedoch interessiert man sich in der Regel für die Ähnlichkeit bzw. Unähnlichkeit von Clustern, sowie die (Un-)Ähnlichkeit eines Objekts zu einem Cluster. Neben dem zugrundeliegenden Ähnlichkeitsmaß ist ein Cluster-Analyse-Algorithmus durch die Art und Weise definiert, wie er die Distanz zweier Cluster berechnet. Bei allen Algorithmen wird die Clusterdistanz auf die Distanz zweier Objekte (die *Repräsentanten* der Cluster) zurückgeführt, auf die dann die oben erläuterten Maße angewendet werden können. So ist beispielsweise beim Median-Verfahren die Cluster-Distanz definiert als die Distanz zwischen den zentralsten Objekten der Clusters (sog. *Mediane*, siehe S.54). Auf die verschiedenen Cluster-Distanzmaße wird an dieser Stelle nicht genauer eingegangen, da sie bei der Beschreibung der einzelnen Algorithmen (Kap. 2.4.2 und Kap. 2.4.3) erläutert werden.

2.4 Wichtige Cluster-Analyse-Algorithmen

2.4.1 Einteilung der Clustering-Verfahren

Möchte man einen Überblick über die Cluster-Analyse-Verfahren geben, so stellt sich dasselbe Problem wie bei der Cluster-Analyse selbst: man hat eine Vielzahl an Objekten (nämlich die Verfahren), die anhand verschiedener Attribute (die Eigenschaften

der Verfahren) in möglichst homogene und klar voneinander getrennte Gruppen aufgeteilt werden sollen. Hinzu kommt das Problem der uneinheitlichen Terminologie auf diesem Gebiet, sodass es sein kann, dass verschiedene Begriffe ein und dasselbe Verfahren bzw. sehr ähnliche Verfahren bezeichnen. Wie auch beim Begriff „Cluster-Analyse“ selbst (und seinen vielen Synonymen, siehe Kap. 2.2.1) liegt dies wohl daran, dass die Verfahren in den unterschiedlichsten Wissenschaftszweigen entwickelt wurden, wobei die jeweiligen Autoren aber nicht immer voneinander Kenntnis hatten und es folglich vorkam, dass gleiche bzw. sehr ähnliche Verfahren mit verschiedenen Namen belegt wurden [Eck80].
In der Literatur werden die Cluster-Analyse-Verfahren meist anhand des Clustering-Resultats, des Clustering-Prozesses und/oder des Clustering-Kriteriums eingeteilt. Hierbei werden jeweils folgende Unterscheidungen gemacht:

- Bei Einteilung nach dem Resultat:
  - *Hierarchische* vs. *nicht-hierarchische* (bzw. *partitionierende*) Verfahren (siehe Kap. 2.4.2 und Kap. 2.4.3)
  - *Disjunkte* vs. *nicht-disjunkte* Verfahren
    Bei nicht-disjunkten Verfahren können sich die Cluster überlappen, während sie bei disjunkten Verfahren elementfremd sind.
  - *Exhaustive* vs. *nicht-exhaustive* Verfahren
    Bei exhaustiven (oder auch *erschöpfenden*) Verfahren muss jedes Objekt mindestens einem Cluster zugewiesen werden, während bei nicht-exhaustiven Verfahren Objekte auch „übrig“ bleiben dürfen.
- Bei Einteilung nach dem Prozess:
  - *Iterative* vs. *nicht-iterative* Verfahren
    Iterative Verfahren wenden einen Lösungsschritt wiederholt an, während nicht-iterative Verfahren aus einem einzigen Schritt oder mehreren verschiedenen Schritten bestehen.
  - Bei hierarchischen Verfahren:
    Agglomerative vs. *divisive* Verfahren
    Agglomerative Verfahren fassen kleinere Cluster zu immer größeren zusammen, während divisive Verfahren große Cluster in immer kleinere unterteilen.
  - *Polythetische* vs. *monothetische* Verfahren
    Polythetische Verfahren verwenden simultan alle Attribute der Objekte, während monothetische Verfahren bei jedem Schritt jeweils nur ein Attribut zur Gruppierung verwenden.
- Bei Einteilung nach dem Kriterium:
  - *Globale* vs. *partielle* Verfahren
    Bei globalen Verfahren erfolgt die Clusterbildung anhand eines Gütekriteriums für die gesamte Partition, während partielle Verfahren Kriterien verwenden, die sich nur auf Teilaspekte der Partition beziehen.

Am häufigsten findet die Unterscheidung in hierarchische und partitionierende Verfahren Verwendung, wobei dann meist die hierarchischen noch einmal in agglomerative und divisive Verfahren unterteilt werden. Diese Einteilung soll auch im Folgenden den Rahmen für die kurze Erläuterung einiger Cluster-Analyse-Algorithmen bilden. Hierbei sei noch bemerkt, dass diese beiden Verfahrenstypen keineswegs den gesamten Bestand an Cluster-Analyse-Algorithmen abdecken, wohl aber den größten Teil. Daneben gibt es noch andere Verfahren, wie z.B. dichte-, raster-, oder modellbasierte Verfahren.

2.4.2 Hierarchische Verfahren

Hierarchische Verfahren erzeugen ein System von ineinander geschachtelten Clustern, wobei das größte Cluster die gesamte Objektmenge umfasst und die kleinsten Cluster jeweils nur aus einem Objekt bestehen. Die Cluster sind in verschiedene Ähnlichkeits- oder Distanzebenen eingeteilt, und zwar so, dass die Cluster einer Ebene paarweise disjunkt sind und dass die Vereinigung aller Cluster einer Ebene gerade die Objektmenge ergibt. Man erhält also nicht eine einzige Clusterung, sondern eine Menge von Clusterungen, die sich in der Anzahl der Cluster, der Homogenität innerhalb der Cluster und der Heterogenität der Cluster unterscheiden. Dies zeigt schon einen Vorteil des Verfahrens: man kann entsprechend den Wünschen an Clusteranzahl bzw. Homogenität/Heterogenität einfach die entsprechende Ähnlichkeitsebene wählen, ohne dass man vorher Werte wie beispielsweise die Clusteranzahl festlegen muss.
Formal lässt sich ein hierarchisches Cluster-System folgendermaßen beschreiben [Eck80]:

Definition 2.4.1 (Cluster-Verfeinerung) *Sei $\mathcal{M} = \{X_1, \ldots, X_n\}$ eine Menge von Objekten und $C_j = \{c_1^j, \ldots, c_{k_j}^j\}$, $j \in \mathbb{N}$, eine Clusterung (Partition) dieser Menge. Eine Clusterung C_i heißt* Verfeinerung *von C_j ($C_i \subseteq C_j$) genau dann, wenn für jedes Cluster $c_p^i \in C_i$ ein Cluster $c_q^j \in C_j$ existiert, in dem es vollständig enthalten ist. Eine echte Verfeinerung ($C_i \subset C_j$) liegt vor, wenn $C_i \subseteq C_j$ und $C_i \neq C_j$.*

Definition 2.4.2 (Hierarchisches Cluster-System) *Sei $\mathcal{L}$ die Menge aller möglichen Clusterungen von $\mathcal{M}$. Ein* hierarchisches Cluster-System *ist eine geordnete Menge von Partitionen $C_0, \ldots, C_{n-1}$ mit den folgenden Eigenschaften:*

1. *$C_0, \ldots, C_{n-1} \in \mathcal{L}$*
2. *C_0 ist die triviale Partition der n Objekte in n Cluster mit jeweils einem Objekt; C_{n-1} ist die triviale Partition der n Objekte in ein einziges Cluster.*
3. *$C_i \subset C_{i+1}$ für $0 \leq i \leq n-2$*

Ein weiterer Vorteil der hierarchischen Clustering-Verfahren liegt in der anschaulichen Darstellung der geordneten Menge von Partitionen mittels eines so genannten *Dendrogramms.* Ein solches besteht aus einer Hierarchie von Clustern und einer Distanz- oder Clusterskala, auf welcher für jede Clusterung C_i der Distanzwert[2] d_i abgetragen

[2]Der Distanzwert leitet sich von der Cluster-Distanz des jeweiligen Verfahrens ab (welche bei den einzelnen Verfahren im Folgenden beschrieben wird).

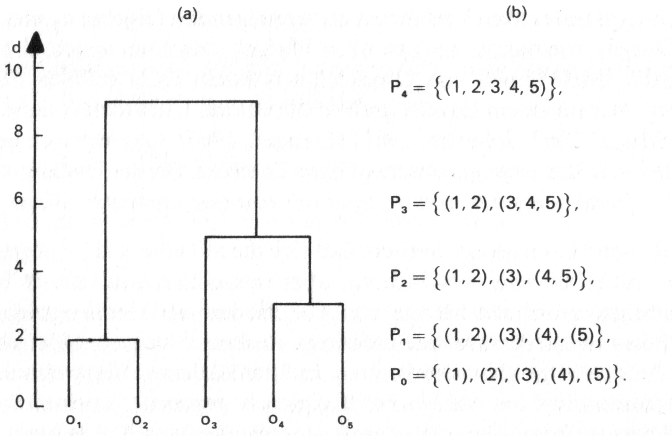


Abbildung 2.1: (a) Dendrogramm für 5 Objekte, (b) geordnete Menge der Partitionen [Eck80]

wird. Dadurch sind die jeweiligen Hierarchiestufen genau bestimmt. Mit abnehmender Anzahl der Cluster wächst d monoton, also $d_0 < d_1 < \ldots < d_{n-1}$. Abbildung 2.1 zeigt ein Dendrogramm für eine Menge von 5 Objekten.
Abgesehen von den verschiedenen Clusterungen und den zugehörigen Distanzwerten kann man einem Dendrogramm aber auch noch weitere nützliche Informationen entnehmen. So kann man beispielsweise ablesen, wie die verschiedenen Cluster miteinander zusammenhängen: eine *intensive Clusterstruktur* liegt vor, wenn die Cluster durch sukzessive Fusion von Clustern gleicher Größe entstehen, wohingegen das sukzessive Hinzufügen von benachbarten Einzelobjekten zu einer *schwachen Clusterstruktur* führt. Deutlich ausgeprägte und gut separierte Cluster erkennt man daran, dass sie über einen weiten Bereich der Distanzskala unverändert bleiben. Ein Objekt, das erst relativ spät einem großen Cluster zugeordnet wird, ist ein Ausreißer.
Das Dendrogramm ist isomorph zur Darstellungsform des Cluster-Systems als geordnete Menge der Partitionen [Eck80], d.h. bei der Darstellung als Dendrogramm gehen keinerlei Informationen verloren, sondern es werden vielmehr die enthalten Informationen besser sichtbar.
Eine wesentliche Eigenschaft der hierarchischen Clustering-Verfahren (im Gegensatz zu den partitionierenden Verfahren) ist die Nicht-Revidierbarkeit der Zuordnung eines Objekts zu einem Cluster, d.h. sind Objekte in einem Schritt des Verfahrens demselben (bzw. bei divisiven Verfahren verschiedenen) Clustern zugeordnet worden, so können sie nicht in einem späteren Schritt wieder getrennt (bzw. vereinigt) werden.
Wie in Kapitel 2.4.1 bereits erwähnt, lassen sich hierarchische Verfahren in *agglomerative* und *divisive* Verfahren unterteilen. Agglomerative Verfahren beginnen mit der feinsten Partition C_0, welche durch sukzessives Verschmelzen von je zwei Clustern immer weiter vergröbert wird, bis schließlich die gröbste Partition C_{n-1} erreicht wird, bei der alle Objekte in einem Cluster vereinigt sind. Divisive Verfahren gehen den umgekehrten Weg: sie beginnen mit der gröbsten Partition C_{n-1} und verfeinern diese schrittweise indem sie in jedem Schritt ein Cluster aufspalten bis schließlich jedes Ob-

jekt ein eigenes Cluster bildet und somit die Stufe der feinsten Partition C_0 erreicht ist.

Agglomerative Verfahren

Die grundlegenden Schritte eines hierarchisch-agglomerativen Algorithmus lassen sich folgendermaßen angeben [SL77]:
Eingabe: Distanzmatrix[3] D

1. Beginne mit der feinsten Partition $C_0 := \{c_1, \ldots, c_n\}$, wobei $c_i := \{X_i\}$
2. Suche in der Distanzmatrix die beiden Cluster c_p und c_q mit der geringsten Distanz zueinander, also $d_{pq} = min_{i \neq j} d_{ij}$ (gibt es mehrere minimale Distanzen, so entscheidet ein willkürlich zu wählendes Kriterium, z.B. die zuerst aufgefundene kleinste Distanz)
3. Fusioniere die Cluster c_p und c_q zum neuen Cluster c_q^{neu}, wodurch sich die Anzahl der Cluster um 1 reduziert; $c_q^{neu} := c_p \cup c_q$.
4. Ändere die q-te Zeile und Spalte der Distanzmatrix, indem die Abstände zwischen dem neuen Cluster c_q^{neu} und allen anderen Clustern neu berechnet werden und streiche die p-te Zeile und Spalte der Matrix.
5. Beende nach $n - 1$ Schritten, wenn also die gröbste Partition vorliegt, d.h. alle Objekte ein einziges Cluster bilden. Andernfalls fahre beim zweiten Schritt mit der aktualisierten Distanzmatrix fort.

Der zentrale Unterschied der verschiedenen agglomerativen Verfahren besteht darin, wie im vierten Schritt die Distanz des neu gebildeten Clusters zu den anderen Clustern berechnet wird. Viele gebräuchliche Verfahren verwenden eine von Lance & Williams 1966 entwickelte Rekursionsformel. Danach lässt sich die Distanz d_{qi} des durch Fusion der Cluster c_p und c_q entstandenen Clusters c_q^{neu} zu einem beliebigen anderen Cluster c_i folgendermaßen berechnen:

$$d_{qi}^{neu} = \alpha_p d_{pi} + \alpha_q d_{qi} + \beta d_{pq} + \gamma |d_{pi} - d_{qi}| \tag{2.1}$$

Je nach Verfahren nehmen die Parameter $\alpha_p,\ \alpha_q,\ \beta$ und γ andere Werte an.

Single-Linkage Das *Single-Linkage-* oder *Nearest-Neighbour-Verfahren* erhält man, wenn man bei Formel (2.1) die Parameter wie folgt wählt:

$$\alpha_p = \alpha_q = \frac{1}{2},\ \beta = 0 \text{ und } \gamma = -\frac{1}{2}$$

Somit erhält man

$$d_{qi}^{neu} = \frac{1}{2}(d_{pi} + d_{qi}) - \frac{1}{2}|d_{pi} - d_{qi}|$$

[3] Da sich Distanzen leicht in Ähnlichkeiten umrechnen lassen (und umgekehrt, siehe Kap. 2.3.1), kann man natürlich auch eine Ähnlichkeitsmatrix zugrunde legen.

oder anschaulicher

$$d_{qi}^{neu} = min(d_{pi}, d_{qi}).$$

Die Distanz zweier Cluster entspricht also dem Abstand der am nächsten zusammenliegenden Objekte der zwei Cluster (wobei die zwei Objekte nicht im selben Cluster liegen dürfen). Vereinigt werden in einem Fusionsschritt dann jeweils die beiden Cluster, die die „nearest neighbours“ haben. Auf einer Distanzebene d_k sind folglich alle Objekte zu einem Cluster zusammengefaßt, die zu mindestens einem anderen Objekt dieses Clusters eine Distanz kleiner oder gleich d_k besitzen.
Dieses Verfahren ist *stark kontrahierend:* es tendiert dazu, einzelne Objekte mit bereits vorhandenen Clustern zu fusionieren bzw. zu verketten, was sich anschaulich so interpretieren lässt, dass der Raum um das entsprechende Cluster zusammengezogen („kontrahiert“) wird. Dies führt zu dem für das Single Linkage typischen *Aneinanderreihungs-Effekt* (engl. *chaining*), welcher besonders durch Objekte begünstigt wird, die im Grenzbereich zweier (oder mehrerer) Cluster liegen. Schon wenige solcher Objekte können zu einer nicht mehr interpretierbaren Lösung führen. Liegen jedoch lang gestreckte oder U-förmige Cluster vor (was aber eher selten vorkommt), so ist dieser chaining-Effekt von Vorteil, weil solche nicht-konvexen Cluster damit entdeckt werden [Eck80].
Des weiteren hat dieses Verfahren die Eigenschaft, dass die Clusterlösung unter monotonen Transformationen der Distanzmatrix invariant bleibt, d.h. sind zu einer gegebenen Datenmatrix zwei Distanzmatrizen gleich bezüglich der Rangordnung ihrer Werte, so liefert das Single-Linkage-Verfahren auf ihnen identische Cluster-Hierarchien.
Single-Linkage ist das älteste und zugleich einfachste Clustering-Verfahren. Eine wichtige Rolle spielt es auch in der Graphentheorie beim Finden von minimalen aufspannenden Bäumen.

Complete Linkage Setzt man in (2.1) für

$$\alpha_p = \alpha_q = \frac{1}{2},\ \beta = 0 \text{ und } \gamma = \frac{1}{2},$$

so erhält man die Rekursionsformel

$$d_{qi}^{neu} = \frac{1}{2}(d_{pi} + d_{qi}) + \frac{1}{2}|d_{pi} - d_{qi}|$$

oder anschaulicher

$$d_{qi}^{neu} = max(d_{pi}, d_{qi}),$$

was das so genannte *Complete-Linkage-* oder *Furthest-Neighbour-Verfahren* definiert. Hier wird die Distanz zweier Cluster berechnet, indem man die Distanz ihrer am weitesten voneinander entfernt liegenden Objekte („furthest neighbours“) bestimmt. In einem Fusionsschritt werden dann jeweils die beiden Cluster miteinander verschmolzen, für die diese maximale Distanz am kleinsten ist. Die so auf einer bestimmten Ebene der Hierarchie gebildeten Cluster haben daher die Eigenschaft, dass die Objekte eines Clusters nicht nur dem nächsten Nachbarn, sondern auch allen anderen Objekten des Clusters (im Sinne des Fusionskriteriums) ähnlich sind. Darum neigt dieses Verfahren dazu, homogene Cluster zu bilden.

Im Gegensatz zum Single-Linkage ist das Complete-Linkage *(raum-)dilatierend,* d.h. es tendiert dazu, die Punkte im r-dimensionalen Merkmalsraum in mehrere kleine und kompakte Cluster zusammenzufassen, wodurch sich anschaulich eine Ausweitung des Raumes ergibt.
Mit dem Single-Linkage gemeinsam ist diesem Verfahren die Eigenschaft der Invarianz unter monotonen Transformationen der Distanzmatrix sowie die einfache rechnerische Durchführung. Gemeinsam ist ihnen auch der Nachteil, dass die Fusion zweier Cluster völlig von einem einzigen Distanzwert abhängt, was zur Folge hat, dass diese beiden Verfahren sehr sensibel auf Messfehler reagieren.

Average-Linkage (UPGMA) Das *Average-Linkage-Verfahren* erhält man aus (2.1) mit folgenden Parameterwerten:

$$\alpha_p = \alpha_q = \frac{1}{2} \text{ und } \beta = \gamma = 0$$

Dies ergibt die Rekursionformel

$$d_{qi}^{neu} = \frac{1}{2}(d_{pi} + d_{qi})$$

Bei der Fusion zweier Cluster wird hier zur Berechnung der Distanzen dieses neuen Clusters zu allen anderen der Mittelwert der Distanzen der zwei ursprünglichen Cluster zu den anderen Clustern genommen. Um es von den anderen Mittelwert-Verfahren (s.u.) zu unterscheiden, wird dieses Verfahren in der neueren Literatur oft auch als *Unweighted Arithmetic Average Clustering* oder *Unweighted Pair-group Method Using Arithmetic Averages (UPGMA)* bezeichnet.
Dieses Verfahren ist weder kontrahierend noch dilatierend und zählt daher zu den so genannten *konservativen* (raumerhaltenden) Verfahren. Somit kann das Average-Linkage-Verfahren zwischen den beiden Extremen Single-Linkage und Complete-Linkage eingeordnet werden. Oft führt es auch zu geeignten Kompromissen.

Weighted Average-Linkage (WPGMA) Dieses Verfahren ist eine gewichtete Verallgemeinerung des Average-Linkage-Verfahrens, bei dem die Anzahl der Objekte in den Clustern als Gewichte eingehen (es wird manchmal auch als *Group-Average-Linkage* bezeichnet). Die Parameter lauten hier

$$\alpha_p = \frac{n_p}{n_p + n_q},\ \alpha_q = \frac{n_q}{n_p + n_q},\ \text{und } \beta = \gamma = 0,\ \text{wobei } n_k := |c_k|,$$

was folgende Rekursionsformel liefert:

$$d_{qi}^{neu} = \frac{n_p d_{pi} + n_q d_{qi}}{n_p + n_q}$$

Den obigen Spezialfall erhält man für $n_p = n_q = 1$. Anschaulicher lässt sich die Formel auch folgendermaßen ausdrücken:

$$d_{qi}^{neu} = \frac{1}{(n_p + n_q)n_i} \sum_{k=1}^{n_p+n_q} \sum_{j=1}^{n_i} d_{kj}$$

wobei d_{kj} die Distanz zwischen $X_k \in c_q^{neu}$ und $X_j \in c_i$ ist. Hier wird die Distanz zwischen zwei Clustern also bestimmt, indem man das arithmetische Mittel aller Objektdistanzen zwischen den beiden Clustern berechnet. In einem Fusionsschritt werden jeweils die beiden Cluster verschmolzen, für die der Durchschnitt aller Objektdistanzen minimal ist.
Aufgrund der Definition der Distanz als Mittelwert können hier nur Distanzmaße verwendet werden, für die der Mittelwert eine sinnvolle Größe darstellt (dies gilt natürlich auch für den Spezialfall des UPGMA-Verfahrens). Wie auch der Spezialfall ist das WPGMA-Verfahren konservativ.

Centroid-Verfahren Beim *Centroid-Verfahren* werden die Objekte $X_{p1}, \ldots, X_{pn_p}$ eines Clusters c_p durch ihren Schwerpunkt (Centroiden) $\bar{X}_p$ ersetzt. Als Distanz zwischen den Clustern wird die der räumlichen Vorstellung sehr nahe liegende euklidische Distanz zwischen den Cluster-Centroiden zugrunde gelegt:

$$d^2_{\bar{X}_q\bar{X}_i} = \sum_{k=1}^{r}(\bar{x}_{qk} - \bar{x}_{ik})^2,$$

wobei $\bar{x}_{qk}$ das arithmetische Mittel der Werte des k-ten Attributs im Cluster c_q ist. Ersetzt man nun das durch Fusion aus den Clustern c_p und c_q entstandene Cluster c_q^{neu} durch den neuen Schwerpunkt $\bar{X}_q^{neu}$, so erhält man für die Distanz zwischen dem neuen Schwerpunkt und dem Schwerpunkt eines anderen Clusters c_i

$$d^2_{\bar{X}_q^{neu}\bar{X}_i} = \frac{n_p}{n_p+n_q}\sum_{k=1}^{r}(\bar{x}_{pk}-\bar{x}_{ik})^2+\frac{n_q}{n_p+n_q}\sum_{k=1}^{r}(\bar{x}_{qk}-\bar{x}_{ik})^2-\frac{n_pn_q}{(n_p+n_q)^2}\sum_{k=1}^{r}(\bar{x}_{pk}-\bar{x}_{qk})^2$$

Die Parameter für die Rekursionsformel lauten also

$$\alpha_p = \frac{n_p}{n_p+n_q},\ \alpha_q = \frac{n_q}{n_p+n_q},\ \beta = -\frac{n_pn_q}{(n_p+n_q)^2} \text{ und } \gamma = 0,$$

sodass man

$$d_{qi}^{meu} = \frac{n_p}{n_p+n_q}d_{pi} + \frac{n_q}{n_p+n_q}d_{qi} - \frac{n_pn_q}{(n_p+n_q)^2}d_{pq}$$

als Rekursionsformel erhält.
Bei jedem Fusionsschritt werden jeweils die beiden Cluster vereinigt, deren euklidischer Centroidabstand minimal ist. Da für jedes Cluster c_i die Anzahl n_i seiner Objekte in die Rechnung miteingeht sind die Cluster entsprechend ihrer Größe gewichtet. Das Verfahren kann auch für andere Distanzfunktionen verwendet werden, was jedoch nicht empfehlenswert ist, da dies zu schwer interpretierbaren Ergebnissen führen kann [SL77].
Zwar ist die Nähe zur räumlichen Vorstellung gut für Anschauungszwecke, jedoch bringt diese Eigenschaft auch die Gefahr mit sich, dass es zu *Inversionen* kommen kann, was bedeutet, dass die Distanz zwischen den Clustern mit abnehmender Clusterzahl nicht monoton wächst. Dies liegt daran, dass bei der Fusion zweier Cluster der neue Schwerpunkt näher bei einem der anderen Schwerpunkte liegen kann als die zwei ursprünglichen Schwerpunkte. Treten Inversionen auf, so wird das zugehörige Dendrogramm (in dem es dann zu Kreuzungen kommt) schwer interpretierbar. Abbildung 2.2 zeigt ein Dendrogramm mit einer Inversion.

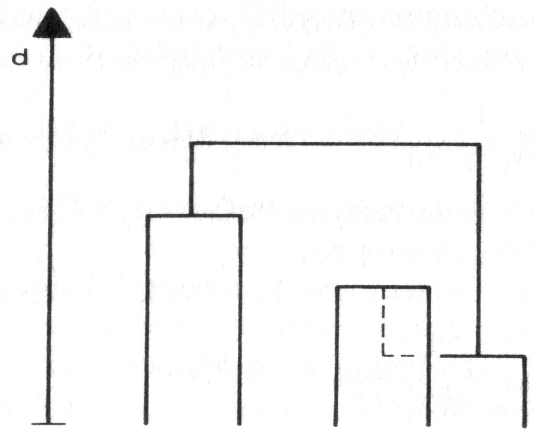


Abbildung 2.2: Dendrogramm mit Inversion [Eck80]

Median-Verfahren Wählt man in (2.1) für die Parameter

$$\alpha_p = \alpha_q = \frac{1}{2},\ \beta = -\frac{1}{4} \text{ und } \gamma = 0,$$

so erhält man das *Median-Verfahren* mit der Rekursionsformel

$$d_{qi}^{neu} = \frac{1}{2}(d_{pi} + d_{qi}) - \frac{1}{4}d_{pq}$$

Für $n_p = n_q$ stimmt dieses Verfahren mit dem Centroid-Verfahren überein. Es wurde 1967 von Gower vorgeschlagen um einen Mangel des Centroid-Verfahrens zu beseitigen: durch die Gewichtung mit der Clustergröße hat letzteres nämlich die Eigenart, dass bei der Fusion zweier sehr unterschiedlich großer Cluster der neue Schwerpunkt sehr viel näher am Schwerpunkt des großen Clusters liegt und dadurch die charakteristischen Eigenschaften des kleinen Clusters weitgehend verloren gehen. Beim Medoid-Verfahren hingegen wird jedem Cluster unabhängig von seiner Größe dasselbe Gewicht gegeben, indem für zwei zu verschmelzende Cluster $c_p,\ c_q$ willkürlich $n_p = n_q$ gesetzt wird. Legt man die euklidische Distanz zugrunde, so bedeutet dies anschaulich, dass die beiden Schwerpunkte $\bar{X}_p,\ \bar{X}_q$ durch den Mittelpunkt ihrer Verbindungslinie (Median) ersetzt werden. Hieran wird auch deutlich, dass dieses Verfahren - wie das Centroid-Verfahren - an die euklidische Distanz gebunden ist, wenn man das Ergebnis anschaulich interpretieren will. Ebenfalls gemeinsam mit dem Centroid-Verfahren ist dem Median-Verfahren, dass es konservativ ist und dass es zu Inversionen kommen kann.

Wards Verfahren Wie auch beim Centroid- und Median-Verfahren werden bei *Wards Verfahren* (Ward & Hook, 1963) die Cluster durch ihre Schwerpunkte repräsentiert. Ursprünglich war es ein allgemeiner agglomerativer Algorithmus, der in jedem Fusionsschritt eine bestimmte Funktion zu optimieren versucht. Später, nachdem Ward sein Verfahren anhand der Fehlerquadratsumme erläuterte, wurde diese

Methode dann als *error sum of squares method* bekannt. Im Gegensatz zu den bisher vorgestellten Verfahren, die in jedem Schritt die (in jeweils unterschiedlichem Sinne) ähnlichsten Cluster vereinigen, wird hier versucht, in jedem Schritt jeweils die beiden Cluster zu vereinigen, die den geringsten Zuwachs zu einem *Heterogenitätsmaß,* nämlich der Fehlerquadratsumme innerhalb der Cluster, liefern. Hierbei ist die Fehlerquadratsumme E_p eines Clusters c_p definiert durch

$$E_p := \sum_{k=1}^{r} \sum_{j=1}^{n_p} (x_{jk} - \bar{x}_{pk})^2,$$

die Fehlerquadratsumme E_W innerhalb der Cluster durch

$$E_W := \sum_{p=1}^{N} E_p, \text{ wobei } N := \textit{Anzahl Cluster}$$

und die Fehlerquadratsumme zwischen den Clustern durch

$$E_B := \sum_{p=1}^{N} n_p (\bar{X}_p - \bar{X})^2,$$

wobei $\bar{X}$ den Mittelwert aller Objekte $X_1, \ldots, X_n$ bezeichne.
Die totale Fehlerquadratsumme E_T, die definiert ist durch

$$E_T := \sum_{j=1}^{n} (X_i - \bar{X})^2,$$

ist ein konstanter Wert, der sich zerlegen lässt in die Fehlerquadratsumme innerhalb der Cluster, E_W, und die Fehlerquadratsumme zwischen den Clustern, E_B, also $E_T = E_W + E_B$. Vereinigt man zwei Cluster $c_p,\ c_q$, so beträgt der Zuwachs von E_W

$$\Delta E_{pq} = E_q^{neu} - E_p - E_q$$

Da E_T konstant ist, entspricht der Fehlerquadratzuwachs genau der Fehlerquadratsumme, die vor der Fusion zwischen den Clustern bestanden hat. Dies bedeutet, dass die Fehlerquadratzuwächse über die Fusionsstufen addierbar sind (auf der niedrigsten Stufe (feinste Partition) ist $E_W = 0$ und $E_B = E_T$, auf der höchsten Stufe (ein einziges Cluster) ist $E_W = E_T$ und $E_B = 0$). Wards Verfahren ist das einzige rekursive Verfahren, das ein Inter-Cluster-Distanzmaß mit dieser Eigenschaft besitzt [Eck80], was die Interpretation der Ergebnisse erleichtert.
Der Fehlerquadratzuwachs lässt sich auch als quadrierte euklidische Centroid-Distanz darstellen:

$$\Delta E_{pq} = \frac{n_p n_q}{n_p + n_q} \sum_{k=1}^{r} (\bar{x}_{pk} - \bar{x}_{qk})^2.$$

Bei gleichem Centroid-Abstand und gleichem Größenverhältnis $\frac{n_p}{n_q}$ werden demzufolge als erstes solche Cluster verschmolzen, für die die Gesamtzahl der in ihnen enthaltenen Objekte $n_p + n_q$ am kleinsten ist, d.h. zu Beginn des Verfahrens werden besonders

in den Regionen des Merkmalsraums Cluster gebildet, in denen viele Objekte dicht beieinander liegen.
Bei gleichem Centroid-Abstand und gleicher Gesamtgröße $n_p + n_q$ der zu fusionierenden Cluster werden zuerst diejenigen Cluster vereinigt, bei denen die absolute Größendifferenz $|n_p - n_q|$ am höchsten ist, was bedeutet, dass häufig kleinere mit größeren Clustern fusioniert werden. Das Verfahren tendiert also zur Bildung etwa gleich großer Cluster [Boc74].
Legt man diesem Verfahren eine andere als die quadrierte euklidische Distanz zugrunde, so gehen die Meinung über die Qualität der Ergebnisse auseinander. So meint Eckes, [Eck80], dass ein anderes Distanzmaß wie beim Centroid-Verfahren zu schwer interpretierbaren Ergebnissen führt, während Steinhausen&Langer, [SL77], der Meinung sind, dass sich diese Methode auch auf andere Distanzen gut anwenden lässt.
Auch Wards Verfahren lässt sich mittels der Formel (2.1) ausdrücken (Wishart, 1969), nämlich mit den Koeffizienten

$$\alpha_p = \frac{n_p + n_i}{n_p + n_q + n_i},\ \alpha_q = \frac{n_q + n_i}{n_p + n_q + n_i},\ \beta = -\frac{n_i}{n_p + n_q + n_i} \text{ und } \gamma = 0$$

was folgende Rekursionsformel ergibt:

$$d_{qi}^{neu} = \frac{1}{n_p + n_q + n_i}[(n_p + n_i)d_{pi} + (n_q + n_i)d_{qi} - n_i d_{pq}].$$

Divisive Verfahren

Divisive Verfahren spielen aufgrund ihres meist hohen Rechenaufwands und der im Vergleich zu agglomerativen Verfahren schlechteren Ergebnisse in der Praxis eine sehr untergeordnete Rolle und werden hier deshalb nur kurz beschrieben.
Zu jedem der oben vorgestellten agglomerativen Algorithmen kann man den entsprechenden divisiven Algorithmus konstruieren, indem man das Divisionskriterium, anhand dessen ein Cluster möglichst gut in zwei Teilcluster aufgespalten wird, entsprechend wählt. Im Gegensatz zu den agglomerativen können die divisiven Verfahren nicht nur polythetisch sondern auch monothetisch vorgehen.

Divisiv-polythetische Verfahren Das wohl bekannteste divisiv-polythetische Verfahren ist das von Edwards & Cavalli-Sforza 1965 entwickelte Verfahren, dessen agglomeratives Gegenstück das Verfahren von Ward ist. Zu Beginn wird das Anfangscluster $c_0 = \{X_1, \ldots, X_n\}$ so in zwei Teilcluster zerlegt, dass die Fehlerquadratsumme zwischen den Clustern maximal wird. Dies geschieht mittels totaler Enumeration der $2^{n-1} - 1$ möglichen Partitionen mit den zugehörigen Fehlerquadratsummen-Werten, woraus dann die Partition ausgewählt wird, die E_B maximiert bzw. E_W minimiert.[4] Ebenso wird dann in jedem weiteren Schritt jeweils ein Cluster c_p zerlegt, indem alle $2^{n_p-1} - 1$ Zweiteilungen erzeugt werden und die in obigem Sinne beste ausgewählt wird. Das Verfahren endet, wenn jedes Objekt ein einzelnes Cluster bildet, d.h. wenn $E_W = 0$ bzw. $E_B = E_T$ ist.
Die Methode der totalen Enumeration führt schon bei kleinen Datenmengen zu einem enormen Rechenaufwand. Auch die Verbesserung von Scott & Symons (1971), bei

[4] Zur Definition von E_B und E_W siehe *Wards Verfahren,* S. 54.

der die Anzahl der zu vergleichenden Partitionen auf $(2^r - 2)\binom{n}{r}$ (wobei r = Anzahl Attribute) beschränkt wird, brachte keine praxisrelevante Laufzeit-Verbesserung.
Eine etwas bessere, aber für den praktischen Gebrauch ebenfalls indiskutable Laufzeit hat das divisive Verfahren von MacNaughton-Smith et al. (1964), das auch unter dem Namen *dissimilarity analysis* bekannt ist. Hierbei wird zu Beginn für jedes Objekt die Distanz zu dem aus den übrigen $n-1$ Objekten bestehenden Cluster c_p berechnet. Mit dem (bzw. einem) Objekt X_j, für das diese Distanz d_{jp} maximal ist, wird dann ein neues Cluster c_q gebildet. Anschließend wird für jedes der $n-1$ in Cluster c_p verbleibenden Objekte X_i $(i = 1, \ldots, n;\ i \neq j)$ die Distanz d_{ip} zu den anderen $n-2$ Objekten in diesem Cluster, sowie die Distanz d_{iq} zu dem neuen Cluster c_q berechnet. Gibt es ein Objekt in c_p, für das $d_{ip} - d_{iq} > 0$ gilt, so wird unter allen Objekten, die diese Bedingung erfüllen, jenes mit der größten Differenz in das Cluster c_q verschoben. Dieser Schritt (inklusive der Neuberechnung der Distanzen) wird so lange wiederholt, bis es kein Objekt mehr in Cluster c_p gibt, für das $d_{ip} > d_{iq}$ gilt. Anschließend werden die beiden so entstandenen Cluster nach demselben Verfahren geteilt. Dies wird so lange fortgeführt, bis eine weitere Teilung nicht mehr sinnvoll ist, was durch ein entsprechendes Abbruchkriterium modelliert werden kann.

Divisiv-monothetische Verfahren Im Gegensatz zu polythetischen Verfahren, bei denen alle Attribute gleichzeitig berücksichtigt werden, betrachtet man bei monothetischen Verfahren jeweils immer nur ein Attribut, das so genannte *Divisionsattribut.* Meist werden binäre Attribute verwendet, sodass bei der Zweiteilung eines Clusters alle Objekte, die das betrachtete Attribut besitzen, zu einem Teilcluster zusammengefaßt werden können und alle Objekte, die das Attribut nicht besitzen, zu einem zweiten. Kommen stetige oder diskret numerische Attribute vor, so wird ein Grenzwert festgelegt, anhand dessen in zwei Teilgruppen aufgespalten wird [Eck80].
Monothetische Verfahren liefern gute Ergebnisse, wenn der seltene Fall eintritt, in dem die Objekte so wenig Attribute besitzen, dass jedes Attribut zur Clusterbildung herangezogen werden kann. In diesem Fall nämlich erzeugen monothetische Verfahren völlig homogene Cluster. Meist ist die Attributanzahl jedoch sehr groß und man steht vor dem Problem, die richtigen Attribute auszuwählen und vor allem sie in der richtigen bzw. für eine gute Clusterung besten Reihenfolge als Divisionskriterium anzuwenden.
Vorteile monothetischer Verfahren sind der geringe Rechenaufwand sowie die leichte Interpretierbarkeit der Ergebnisse, zumindest in Bezug auf die zur Teilung verwendeten Attribute. Demgegenüber stehen jedoch die Nachteile, dass zum einem der Informationsgehalt der Daten bei diesen Verfahren längst nicht so gut genutzt wird wie bei polythetischen Verfahren, und zum anderen sehr leicht schlechte Clusterungen entstehen können, da Objekte, die in allen bis auf einem Attribut übereinstimmen, unterschiedlichen Clustern zugewiesen werden, wenn anhand dieses unterscheidenden Attributs eine Clusterteilung durchgeführt wird.

2.4.3 Partitionierende Verfahren

Partitionierende Verfahren, manchmal auch *k-Clustering* genannt, bestimmen zu einer vorgegebenen Clusterzahl k eine Zerlegung der Datenmenge derart, dass ein bestimm-

tes *Optimierungskriterium* möglichst gut erfüllt wird bzw. eine *Zielfunktion* minimiert oder maximiert wird. Ausgehend von einer Anfangspartition versucht der Algorithmus durch iteratives Verschieben einzelner Objekte von einem Cluster in ein anderes das gewählte Kriterium schrittweise zu optimieren bis keine Verbesserung mehr möglich ist. Der allgemeine Algorithmus (*hill climbing algorithm*) sieht also folgendermaßen aus:

1. Gib eine Anfangspartition mit k Clustern vor (Initiierungsphase)

2. Prüfe, ob sich durch Verschieben eines einzelnen Objekts aus seinem ursprünglichen Cluster in eines der anderen $k-1$ Cluster eine Verbesserung im Wert der gewählten Zielfunktion ergibt. Wenn ja, dann verschiebe das Objekt in das Cluster, mit dem die größte Verbesserung erzielt wird. Aktualisiere die Clusterstruktur.

3. Wiederhole Schritt 2 so lange, bis durch das Verschieben von Objekten von einem Cluster in ein anderes keine Verbesserung der Zielfunktion mehr erreicht werden kann oder ein Abbruchkriterium erfüllt ist.

Die verschiedenen Partitionierungsalgorithmen unterscheiden sich im Wesentlichen nur durch die Wahl des Optimierungskriteriums in Schritt 2.
Bei der Generierung der Anfangspartition gibt es verschiedene Möglichkeiten:

- Standard-Anfangspartition: Ordne die Objekte entsprechend der Eingabereihenfolge den Clustern zu, also das erste Objekt in das erste Cluster, das zweite Objekt in das zweite Cluster,..., das k-te Objekt in das k-te Cluster, das $(k+1)$-te Objekt wieder in das erste Cluster usw. bis alle Objekte einem Cluster zugeordnet sind.

- Wähle die ersten k Objekte oder eine zufällige Stichprobe von k Objekten als „Startzentren“ der k Cluster. Ordne die übrigen $n-k$ Objekte jeweils dem (z.B. im euklidischen Sinne) nächsten Startzentrum oder dem nächsten aus Startzentrum und bereits zugewiesenen Objekten berechneten Mittelpunkt (Centroid) zu.

- Führe eine hierarchische Cluster-Analyse durch und wähle die Hierarchie-Ebene mit k Clustern als Anfangspartition.

Wenn man von „partitionierenden Verfahren“ spricht, so ist im Allgemeinen stets von den eben erläuterten iterativen Verfahren die Rede. Der Vollständigkeit wegen sei an dieser Stelle aber erwähnt, dass es daneben auch noch nicht-iterative Verfahren gibt: Hierbei wird in einem einzigen Schritt eine Partition erzeugt und die Zuordnung der Objekte zu den Clustern ist nicht revidierbar. Diese Verfahren werden aber fast nie verwendet und haben daher keine praktische Bedeutung.

Die bei partitionierenden Verfahren verwendete Zielfunktion lässt sich formal folgendermaßen definieren:

Definition 2.4.3 *Sei* $\mathcal{M} := \{X_1, \ldots, X_n\}$ *eine Menge von Objekten und* $\mathcal{L} := \{C_1, C_2, \ldots\}$ *die Menge aller Clusterungen (Partitionen) von* $\mathcal{M}$*. Eine* Zielfunktion *(bzw. ein* Gütekriterium*)* $z : \mathcal{L} \to \mathbb{R},\ C_i \mapsto z(C_i)$ *ordnet jeder Clusterung* C_i *von* $\mathcal{M}$ *eine reelle Zahl zu.*

Je nach Definition der Zielfunktion z soll das Maximum oder das Minimum über alle Clusterungen bestimmt werden. Wie schon in Kapitel 2.2.4 erläutert, würde hierbei die Methode der totalen Enumeration aller Clusterungen bereits bei kleinen Datenmengen zu einem enormen Rechenaufwand führen. Bei der Suche des Optimums muss man sich also auf einen Teil der Partitionen beschränken, womit man natürlich das Risiko eingeht, jene Partitionen zu „übersehen“, die ein Optimum der Zielfunktion liefern würden. Man findet also eventuell nur ein lokales Optimum. Es hat sich jedoch gezeigt, dass dieses meist nur wenige Prozent schlechter ist als das globale Optimum und man trotzdem ein sehr gutes Clustering-Ergebnis erzielt [SL77]. Die Einschränkung auf einen Teil der Partitionen erzielt man, indem man mit einem - je nach Wahl der Anfangspartition - mehr oder weniger zufälligen Wert der Zielfunktion beginnt und auf dem oben beschriebenen Weg so lange in eine Richtung „läuft“, bis man das nächste lokale Optimum erreicht hat.

k-means-Clustering

Das wohl bekannteste Optimierungskriterium ist das *Varianzkriterium* oder *Fehlerquadratsummen-Kriterium,* das beim k-means-Clustering-Verfahren verwendet wird. Dieser Algorithmus von MacQueen (1967) ist eigentlich nur eine Variante des 1965 von Forgy vorgeschlagenen Verfahrens, bei dem ausgehend von der (z.B. nach einer der oben genannten Methoden berechneten) Anfangspartition mit k Clustern die Centroide (bzw. Mittelpunkte, engl. *means*) der Cluster berechnet werden. Dies setzt natürlich voraus, dass alle Attribute entweder stetig oder derart diskret numerisch sind, dass der Mittelwert ein sinnvoller Wert ist. Anschließend werden alle Objekte erneut betrachtet und jedes Objekt dem Cluster zugewiesen, dessen Centroid es am nächsten liegt, wobei die quadrierte euklidische Distanz zur Abstandsdefinition verwendet wird. Nach einem Durchlauf durch alle Objekte werden die Centroide neu berechnet. Diese zwei Schritte werden so lange iteriert, bis bei einem Durchlauf kein Objekt mehr verschoben wird.
Bei der Variante von MacQueen werden die Centroide nicht erst nach jedem Durchlauf durch alle Datensätze neu berechnet, sondern sofort nach der Zuordnung eines Objektes zu einem anderen Cluster. Die Abbruchbedingung ändert sich dann dahingehend, dass abgebrochen wird, wenn n-mal hintereinander kein Objekt in ein anderes Cluster verschoben wird.
Der Algorithmus sieht also folgendermaßen aus:

1. Bestimme eine Anfangspartition in k Cluster

2. Berechne für jedes Cluster $c_i = \{X_1^{(i)}, \ldots, X_{n_i}^{(i)}\},\ i \in \{1, \ldots, k\}$, den Centroiden $\bar{X}^{(i)} := (\bar{x}_1^{(i)}, \ldots, \bar{x}_r^{(i)})$, wobei $\bar{x}_j^{(i)} := \frac{1}{n_i} \sum_{s=1}^{n_i} x_{js}^{(i)}$ für $j = 1, \ldots, r$ (und $X_q^{(i)} := (x_{q1}^{(i)}, \ldots, x_{qp}^{(i)}),\ q \in \{1, \ldots, n_i\},\ r :=$Anzahl Attribute)

3. Berechne für jedes Objekt $X_q^{(i)},\ i \in \{1,\ldots,k\},\ q \in \{1,\ldots,n_i\}$, die Distanzen $d^2_{X_q^{(i)}\bar{X}^{(j)}} := \left\|X_q^{(i)} - \bar{X}^{(j)}\right\|^2 = \sum_{l=1}^{r}\left(x_{ql}^{(i)} - \bar{x}_l^{(j)}\right)^2$ für $j = 1,\ldots,k$. Falls $d^2_{X_q^{(i)}\bar{X}^{(j)}} < d^2_{X_q^{(i)}\bar{X}^{(i)}}$ für ein $j \neq i$, ordne $X_q^{(i)}$ dem Cluster c_j zu, für das dieser Wert minimal wird. Berechne in diesem Fall die Centroide der Cluster c_i und c_j neu.

4. Beende, wenn n-mal hintereinander kein Objekt das Cluster gewechselt hat. Andernfalls gehe zu Schritt 3.

Technisch gesehen wird bei diesem Algorithmus folgende Zielfunktion, die so genannte *Varianz*, minimiert:

$$z(C) := \sum_{i=1}^{k}\sum_{q=1}^{n_i} d^2_{X_q^{(i)}\bar{X}^{(i)}} = \sum_{i=1}^{k}\sum_{q=1}^{n_i}\left\|X_q^{(i)} - \bar{X}^{(i)}\right\|^2$$

Die wichtigsten Eigenschaften von k-means sind:

- die Laufzeit ist $\mathcal{O}(tkn)$, wobei t=Anzahl Iterationen, k=Anzahl Cluster, n=Anzahl Objekte; in der Regel gilt $k, t << n$.
- das Clustering-Ergebnis hängt neben der Anfangspartition auch von der Eingabe-Reihenfolge ab, wobei dieser Einfluß empirischen Erfahrungen von MacQueens zufolge nicht sehr groß ist [SL77].
- die Anzahl k der Cluster muss vorher festgelegt werden, was sehr schwierig sein kann, wenn man keine Informationen über die Daten hat (meist wird das Verfahren mehrmals mit unterschiedlichen k-Werten angewendet).
- da die gefundenen Cluster relativ „rund“ sind, ist das Verfahren ungeeignet zur Entdeckung nicht-konvexer Cluster.
- problematisch ist auch die Situation von sehr unterschiedlich großen Clustern, da dann Objekte, die am Rand eines großen Clusters liegen, einem kleinen benachbarten Cluster zugeordnet werden, wenn sie dessen Schwerpunkt näher liegen, als dem Schwerpunkt des großen Clusters.
- Außenseiter können das Ergebnis stark verfälschen, da sie bei Zuweisung zu einem Cluster dessen Centroiden weit nach außen verschieben.
- Objekte mit kategorischen Attributen können nicht verarbeitet werden.

Dem letzten Punkt trägt der 1998 von Huang entwickelte Algorithmus *k-modes* Rechnung. Hierbei wird der Schwerpunkt eines Clusters durch einen Modus (engl. *mode*) ersetzt und ein Distanzmaß für kategorische Attribute verwendet. Das Update eines Clustermodus erfolgt mit Hilfe einer frequenzbasierten Methode.
Der so genannte *k-prototype*-Algorithmus ist eine Mischung aus k-means und k-modes und kann sowohl numerische als auch kategorische Daten verarbeiten.

k-medoid-Clustering

Dieses Verfahren ist dem k-means sehr ähnlich, jedoch werden hier die Cluster nicht durch ihren Schwerpunkt, sondern durch einen Medoid repräsentiert, welcher anschaulich das zentralste Objekt des Clusters ist. Im Gegensatz zum Centroid, der zwischen den Objekten des Clusters liegen kann, stellt der Medoid immer ein Objekt des Clusters dar. Analog zu k-means ist beim k-medoid-Verfahren eine Clusterung dadurch gegeben, dass jedes Objekt dem nächsten der k Medoide zugeordnet ist. Als Distanzmaß wird hier aber meist nicht die quadrierte, sondern die einfache Distanz zugrunde gelegt. Mit der beim k-means-Verfahren eingeführten Notation und $M^{(i)}$:=Medoid von Cluster c_i lautet die zu optimierende Zielfunktion z:

$$z(C) := \sum_{i=1}^{k} \sum_{q=1}^{n_i} d_{X_q^{(i)} M^{(i)}} = \sum_{i=1}^{k} \sum_{q=1}^{n_i} \left\| X_q^{(i)} - M^{(i)} \right\|$$

Der allgemeine Algorithmus sieht folgendermaßen aus:

1. Gib eine Anfangspartition mit k Clustern vor (Initiierungsphase)
2. Prüfe für jeden der $n-k$ Nicht-Medoide X_q, ob sich der Wert der Zielfunktion verbessert, wenn einer der k Medoide durch X_q ersetzt wird. Wenn ja, dann ersetze den Medoid durch X_q, bei dem die größte Verbesserung erzielt wird. Aktualisiere die Clusterstruktur.
3. Wiederhole Schritt 2 so lange, bis durch das Austauschen von Medoiden keine Verbesserung der Zielfunktion mehr erreicht werden kann oder ein Abbruchkriterium erfüllt ist.

Da Medoide nicht wie Schwerpunkte berechnet werden können, sondern geeignet ausgewählt werden müssen, realisieren die verschiedenen Implementierungen von *k-medoid* mehr oder weniger vollständige Suchverfahren, bei denen eine initiale Menge von Medoiden iterativ bezüglich der Zielfunktion verbessert wird, indem Medoide ausgetauscht werden.
Der Algorithmus PAM (Partitioning Around Medoids, 1987) führt eine sehr umfangreiche Suche durch. Hierbei wird in der Initiierungsphase als erster Medoid das zentralste Objekt ausgewählt, also das Objekt, das die obige Zielfunktion minimiert. Die anderen $k-1$ Medoide der Anfangspartition werden ebenso bestimmt: es wird jeweils das Objekt X_q als Medoid ausgewählt, für das die Zielfunktion bezüglich X_q und den bereits ausgewählten Medoiden minimal wird. Nach der Initialisierung wird für jeden Medoid geprüft, ob der Wert der Zielfunktion durch Vertauschen mit einem Nicht-Medoiden verbessert werden kann. Wenn ja, dann wird die aktuelle Clusterung dahingehend geändert, dass der Medoid mit dem Objekt vertauscht wird, mit dem die größte Verbesserung der Zielfunktion erreicht wird. Dieser Schritt wird so lange iteriert, bis keine Verbesserung mehr erzielt wird.
Auch PAM konvergiert gegen ein (möglicherweise nur lokales) Minimum. Das Ergebnis und die Laufzeit sind unabhängig von der Eingabe-Reihenfolge. Der große Nachteil von PAM ist die sehr hohe Laufzeit von $\mathcal{O}\left(n^3 + k(n-k)^2 t\right)$ (wobei t=Anzahl Iterationen), was ihn für große Datenmengen unbrauchbar macht.

Als Abwandlung von PAM mit einer besseren Laufzeit wurde von Kaufman & Rousseeuw der Algorithmus CLARA (Clustering LARge Applications, 1990) vorgeschlagen. Der Unterschied zu PAM besteht darin, dass CLARA nur mit Stichproben arbeitet. Somit wird nur ein kleiner Teil der Daten bearbeitet, und aus diesen werden die Medoide ausgewählt. Dahinter steckt der Gedanke, dass eine zufällig gewählte Stichprobe sehr ähnlich verteilt ist wie die gesamten Daten und dass folglich auch die Medoide der Stichprobe nahe bei den Medoiden der Orginaldaten liegen. CLARA bestimmt von zahlreichen Stichproben jeweils eine k-medoid-Clusterung und gibt von diesen am Ende die (bezüglich der Zielfunktion) beste Clusterung als Ergebnis aus. Durch diese Einschränkung auf Stichproben erzielt man eine Laufzeit von $\mathcal{O}\left(ks^2 + k(n-k)\right)$ (mit s=Anzahl der Stichproben), wodurch CLARA für große Datenmengen wesentlich geeigneter ist als PAM. Der Preis, den man für diese Laufzeitersparnis zahlt, ist die Tatsache, dass eventuell nicht die (lokal) beste Clusterung gefunden wird. Dies ist nämlich genau dann der Fall, wenn ein Objekt, das ein Medoid der besten Clusterung ist, in keiner der Stichproben ausgewählt wird.
Eine weitere Variation von PAM ist CLARANS (Clustering Large Applications based on RANdomized Search, 1994). Hier müssen zwei weitere Parameter, *numlocal* und *maxneighbour*, von Benutzer vorgegeben werden. Ausgehend von verschiedenen zufällig bestimmten Initialpartitionen wird die Suche nach der besten Clusterung *numlocal* mal durchgeführt und die jeweils beste Lösung behalten. Nach der jeweiligen Initialisierung wird wie bei PAM versucht, die aktuelle Clusterung durch Vertauschen eines Medoids mit einem Nicht-Medoid zu verbessern. Im Gegensatz zu PAM werden hierbei aber nicht alle möglichen Paare von Medoiden mit Nicht-Medoiden untersucht, sondern nur *maxneighbour* viele zufällig ausgewählte Paare. Außerdem wird nicht die beste aller untersuchten Vertauschungen durchgeführt, sondern die erste, die eine Verbesserung des Optimierungskriteriums bewirkt (wobei dann natürlich auch die entsprechende Schleife abgebrochen wird). Dies wird so lange iteriert, bis keine Verbesserung der Zielfunktion mehr erreicht wird [ES00]. Damit hat CLARANS eine Laufzeit von $\mathcal{O}\left(numlocal \cdot maxneighbour \cdot w \cdot n\right)$, wobei w=Anzahl der Ersetzungen, was gegenüber PAM doch eine erhebliche Verbesserung darstellt.

Kapitel 3

Der Demographic Clustering Algorithmus

Im Folgenden soll ein Cluster-Analyse-Algorithmus genauer vorgestellt werden, welcher im *IBM DB2 Intelligent Miner for Data*, dem Data Mining-Tool der IBM implementiert ist. Sein Name *Demographic Clustering Algorithmus* ist etwas irreführend, da er nicht nur auf demografische Daten angewendet werden kann, sondern auf beliebige Daten mit qualitativen und quantitativen Attributen.

3.1 Formale Beschreibung

Zieht man die in Kapitel 2.4.1 erläuterten Unterscheidungsmerkmale der Cluster-Analyse-Algorithmen heran, so lässt sich der Demographic Clustering Algorithmus charakterisieren als

- nicht-hierarchisch: es wird keine hierarchische Struktur der Daten erzeugt, sondern eine „einfache“ Zerlegung
- disjunkt: kein Datensatz ist zwei oder mehr Clustern zugeordnet
- exhaustiv: jeder Datensatz ist einem Cluster zugeordnet
- iterativ: eine Anfangspartition wird in mehreren Schritten verbessert
- polythetisch: es werden mehrere (oder alle) Attribute gleichzeitig zur Clusterbildung herangezogen
- global: das verwendete Optimalitätskriterium bezieht sich auf die gesamte Partition, nicht nur auf Teilaspekte

Die Grundidee des Algorithmus ist es, Paare von Datensätze anhand der einzelnen Attributwerte miteinander zu vergleichen, wobei die Anzahl der ähnlichen Attribute den Grad an Ähnlichkeit und entsprechend die Anzahl der unähnlichen Attribute den Grad an Unähnlichkeit der beiden Datensätze ausdrückt.

3.1.1 Optimalitätskriterium

Das beim demografischen Clustern verwendete Optimalitätskriterium beruht auf einem von Condorcet[1] entwickelten Wahlverfahren. Ausgangspunkt war hierbei folgendes Problem:
$n \in \mathbb{N}$ Kandidaten $K_1, \ldots, K_n$ stellen sich für ein politisches Amt zur Wahl. $m \in \mathbb{N}$ Wähler $W_1, \ldots, W_m$ sollen einen Kandidaten wählen. Dazu hat jeder Wähler $W_j,\ j \in \{1, \ldots, m\},\ w_j \in \mathbb{N}$ Stimmen und stellt eine persönliche totale Ordnung $\prec_j$ der Kandidaten auf:

$$K_{\pi_j(1)} \prec_j K_{\pi_j(2)} \prec_j \cdots \prec_j K_{\pi_j(n)},$$

wobei $\pi_j : \{1, \ldots, n\} \to \{1, \ldots, n\}$ eine Permutation ist und $K_i \prec_j K_k$ bedeutet, dass Wähler W_j beim direkten Vergleich von K_i mit K_k dem Kandidaten K_i den Vorzug gibt und mit allen w_j Stimmen für die Anordnung $K_i \prec_j K_k$ stimmt.
Anhand der von den Wählern aufgestellten m persönlichen totalen Ordnungen soll nun aus den $n!$ möglichen totalen Ordnungen eine Ordnung

$$K_{\pi(1)} \prec K_{\pi(2)} \prec \cdots K_{\pi(n)}$$

herausgefunden werden, welche am besten die persönlichen Ordnungen aller Wähler widerspiegelt.
Der naive Ansatz wäre, für jedes Kandidatenpaar $K_i,\ K_k$ aus den beiden möglichen Ordnungen $K_i \prec K_k$ und $K_k \prec K_i$ jene auszuwählen, welche in den persönlichen Ordnungen der Wähler öfter vorkommt. Dies scheitert jedoch daran, dass der so genannte *Condorcet-Effekt* auftreten kann, was bedeutet, dass für ein Paar $K_i,\ K_j$ wegen der Transitivität von $\prec$ sowohl $K_i \prec K_k$ als auch $K_k \prec K_i$ gelten kann, was im Widerspruch zur Antisymmetrie-Eigenschaft der totalen Ordnung steht.
Condorcet betrachtet in seinem Ansatz zwar auch einzelne Kandidatenpaare, jedoch immer im Rahmen einer totale Ordnung aller Kandidaten. Die „beste“ totale Ordnung ist nach seinem Verfahren jene aller $n!$ möglichen totalen Ordnungen, welche den größten Support hat, wobei dieser für eine totale Ordnung Y folgendermaßen definiert ist:

Definition 3.1.1 *Sei Y eine totale Ordnung der Kandidaten $K_1, \ldots, K_n$.*

$$pro(Y) := \sum_{i \neq k} (c_{ik} y_{ik} + \bar{c}_{ik}(1 - y_{ik}))$$

heißt Support für Y und gibt die Zahl der Stimmen an, welche die totale Ordnung Y unterstützen. Dabei ist c_{ik} die Zahl der Stimmen für $K_i \prec K_k$ und $\bar{c}_{ik}$ die Zahl der Stimmen für $K_k \prec K_i$ und

$$y_{ik} := \begin{cases} 1, & \textit{falls in der totalen Ordnung } K_i \prec K_k \textit{ gilt} \\ 0, & \textit{falls in der totalen Ordnung } K_k \prec K_i \textit{ gilt} \end{cases}$$

Für eine totale Ordnung Y werden bei diesem Verfahren also für jedes Kandidatenpaar diejenigen Stimmen gezählt, die für die in Y vorkommende Ordnung der

[1] Jean Antoine Nicolas de Caritat, marquis de Condorcet, 1743-1794

Kandidaten stimmen. Summiert man über alle Kandidatenpaare, so ist die totale Ordnung mit den meisten Stimmen nach Condorcets Verfahren die beste. Da die Gesamtzahl der abgegebenen Wählerstimmen konstant ist, kann man auch die Stimmen gegen eine totale Ordnung Y zählen und das Minimum über alle totalen Ordnungen bestimmen. Das Ergebnis ist das gleiche wie bei der Maximierung des Supports. Für eine genauere Beschreibung des Verfahrens siehe [Mai01], [Mic87].

Dieses Vorgehen von Condorcet lässt sich nun wie folgt auf die Cluster-Analyse übertragen:
Man betrachtet die Atrribute $A_1, \ldots, A_p$ als Wähler, die entscheiden, ob zwei Datensätze $X_i,\ X_j$ ähnlich sind oder nicht. Hierzu steht jedem Attribut $A_l,\ l \in \{1, \ldots, p\}$, ein Ähnlichkeitsmaß s_{ij}^l und eine Ähnlichkeitsschranke $\alpha_l \in [0, 1]$ zur Verfügung. Des weiteren verfügt jedes Attribut A_l über w_l Stimmen ($w_l, l \in \{1, \ldots, p\}$ sind eigentlich die Gewichte der Attribute). Wird den Attributen nun eine Partition C aller Datensätze zur Abstimmung vorgelegt, so stimmen sie folgendermaßen:

- Bei zwei Datensätzen $X_i,\ X_j$ im gleichen Cluster:

  Gilt $s_{ij}^l > \alpha_l$, so stimmt das Attribut A_l mit allen w_l Stimmen für die Partition C, andernfalls mit allen w_l Stimmen dagegen.

- Bei zwei Datensätze $X_i,\ X_j$ in verschiedenen Clustern:

  Gilt $s_{ij}^l < \alpha_l$, so stimmt das Attribut A_l mit allen w_l Stimmen für die Partition C, andernfalls mit allen w_l Stimmen dagegen.

Falls einer der Werte x_{il} oder x_{jl} fehlt oder ein Ausreißer ist, so kann sich das Attribut A_l auch enthalten.
Analog zu oben sei nun c_{ij} die Zahl der Stimmen derjenigen Attribute, für die X_i und X_j ähnlich sind, und $\bar{c}_{ij}$ die Zahl der Stimmen für die Unähnlichkeit der beiden Datensätze. Mit

$$y_{ij} := \begin{cases} 1, & \text{falls } X_i \text{ und } X_j \text{ im gleichen Cluster liegen} \\ 0, & \text{falls } X_i \text{ und } X_j \text{ in verschiedenen Clustern liegen} \end{cases}$$

ist dann der Support für eine gegebene Partition C

$$pro(C) := \sum_{i \neq j} c_{ij} y_{ij} + \bar{c}_{ij}(1 - y_{ij})$$

und die Ablehnung einer Partition C ist gegeben durch

$$anti(C) := \sum_{i \neq j} \bar{c}_{ij} y_{ij} + c_{ij}(1 - y_{ij})$$

Wie oben gilt auch hier, dass aufgrund der konstanten Gesamtstimmenzahl eine Maximierung von $pro(C)$ äquivalent ist zu einer Minimierung von $anti(C)$.
Eine Partition mit maximalem Support (bzw. mit minimaler Ablehnung) spiegelt das Abstimmungsergebnis der Attribute am besten wider. Betrachtet man nun den

Support nicht absolut, sondern relativ zur Gesamtstimmenzahl, so erhält man das so genannte *Condorcet-Kriterium*

$$\mathcal{K}'(C) := \frac{pro(C)}{\sum_{l=1}^{p} w_l} \rightarrow max.$$

Dem beim Demographic Clustering Algorithmus verwendeten Optimalitätskriterium liegt eine Verallgemeinerung des Condorcet-Kriteriums zugrunde, deren Zielfunktion folgende Form hat[2]:

$$\mathcal{K}(C) := pro(C) - anti(C) = \sum_{\substack{i,j \text{ im gleichen} \\ \text{Cluster, } i \neq j}} (s_{ij} - \alpha) - \sum_{\substack{i,j \text{ in verschie-} \\ \text{denen Clustern}}} (s_{ij} - \alpha),$$

wobei

$$s_{ij} := \frac{1}{\sum_{l=1}^{p} w_l} \sum_{l=1}^{p} (w_l \cdot s_{ij}^l)$$

ein gewichtetes aggregiertes Ähnlichkeitsmaß (siehe Kap. 3.1.2) und $\alpha \in [0, 1]$ eine Ähnlichkeitsschranke ist. Eine Maximierung von $\mathcal{K}(C)$ bedeutet eine Maximierung der Homogenität innerhalb der Cluster bei gleichzeitiger Maximierung der Distanzen zwischen den Clustern, was ja genau das Ziel einer Cluster-Analyse ist. Dabei legt der Parameter α fest, wie groß die Homogenität innerhalb der Cluster sein soll, er gibt also das „Niveau“ oder die „Ähnlichkeitsstufe“ an, auf der die Cluster-Analyse durchgeführt werden soll. Mit wachsendem α wächst in der Regel auch die Clusteranzahl der optimalen Partition [Boc74].
Um die Partition zu finden, für die $\mathcal{K}(C)$ maximal wird, ist es nicht nötig, den obigen Term zu maximieren, denn es gilt:

Lemma 3.1.2 *Statt $\mathcal{K}(C)$ zu maximieren, genügt es,*

$$\sum_{\substack{i,j \text{ im gleichen} \\ \text{Cluster, } i \neq j}} (s_{ij} - \alpha) = \left(\sum_{\substack{i,j \text{ im gleichen} \\ \text{Cluster, } i \neq j}} s_{ij} \right) - \alpha \sum_{C_i \in C} n_i(n_i - 1), \quad wo\ n_i = |C_i|,$$

zu maximieren.

Beweis:
Mit dem oben definierten y_{ij} gilt:

$$\begin{aligned} \mathcal{K}(C) = \sum_{i \neq j} (s_{ij} - \alpha) y_{ij} - \sum_{i \neq j} (s_{ij} - \alpha)(1 - y_{ij}) &= \sum_{i \neq j} (s_{ij} - \alpha)(2y_{ij} - 1) \\ &= 2 \underbrace{\sum_{i \neq j} (s_{ij} - \alpha) y_{ij}}_{:= sum} - \underbrace{\sum_{i \neq j} (s_{ij} - \alpha)}_{\substack{konstant,\ da\ unab- \\ hängig\ von\ C}} \end{aligned}$$

[2]Zur besseren Lesbarkeit wird unter dem Summenzeichen ein Datensatz durch seinen Index dargestellt, also bespielsweise X_i durch i.

Beim Maximieren können die zweite Summe sowie der Faktor 2 vor der ersten Summe vernachlässigt werden. Die Summe *sum* lässt sich umformen zu

$$sum = \left(\sum_{\substack{i,j \ im \ gleichen \\ Cluster, \ i \neq j}} s_{ij} \right) - \alpha \cdot \sum_{i \neq j} y_{ij} = \left(\sum_{\substack{i,j \ im \ gleichen \\ Cluster, \ i \neq j}} s_{ij} \right) - \alpha \cdot \sum_{C_i \in C} n_i(n_i - 1)$$

da $y_{ij} = 0$ für i, j in verschiedenen Clustern. □

Somit erhält man das beim Demographic Clustering Algorithmus verwendete Optimalitätskriterium:

$$\boxed{z(C) := \sum_{\substack{i,j \ im \ gleichen \\ Cluster, \ i \neq j}} s_{ij} - \alpha = \left(\sum_{\substack{i,j \ im \ gleichen \\ Cluster, \ i \neq j}} s_{ij} \right) - \alpha \sum_{C_i \in C} n_i(n_i - 1) \ \rightarrow max} \quad (3.1)$$

3.1.2 Ähnlichkeitsmaße

Der Algorithmus soll auch auf Datensätze anwendbar sein, deren Attribute unterschiedlichen Typs sind. Zur Berechnung der Ähnlichkeit zwischen solchen Objekten kann man – wie in Kapitel 2.3.5 erläutert – entweder aus Ähnlichkeitsmaßen für die einzelnen Attributtypen ein Ähnlichkeitsmaß auf den Objekten geeignet aggregieren oder ein ganz neues Ähnlichkeitsmaß definieren. Beim demografischen Clustern ist die erste Möglichkeit realisiert, und zwar mit dem gewichteten Mittelwert der einzelnen Attribute als Aggregationsfunktion. Hierbei wird zur Berechnung der Ähnlichkeit zweier Objekte $X_i,\ X_j$ zunächst die Ähnlichkeit für jedes Attribut einzeln berechnet, d.h. zu jedem Attribut $A_l,\ l \in \{1, \ldots, p\}$, gibt es ein Ähnlichkeitsmaß s^l, das die Ähnlichkeit s^l_{ij} von X_i und X_j nur anhand des Attributs A_l bestimmt. Bei Attributen gleichen Typs wird das gleiche Ähnlichkeitsmaß verwendet. Von den so berechneten Werten s^l_{ij} wird anschließend der gewichtete Mittelwert berechnet. Das beim Demographic Clustering Algorithmus verwendete Ähnlichkeitsmaß hat also die Form

$$\boxed{s_{ij} := \frac{1}{\sum_{l=1}^{p} w_l} \sum_{l=1}^{p} (w_l \cdot s^l_{ij})} \quad (3.2)$$

wobei $w_l \geq 0$ die Gewichte der Ähnlichkeitsmaße $s^l,\ l \in \{1, \ldots, p\}$, sind. Dieses Maß ist eine Verallgemeinerung des Ähnlichkeitsmaßes von Hyvärinen (1962) für mehrstufige Merkmale (siehe Kap.2.3.5, S.44), bei dem die Anzahlen der möglichen Alternativen für jedes Attribut als Gewichte eingehen. Dies bewirkt, dass die Übereinstimmung in einem Attribut mit vielen Alternativen höher bewertet wird als die Übereinstimmung in einem Attribut mit wenigen Alternativen.
Wie die Maße s^l für die verschiedenen Attributtypen aussehen, wird im Folgenden beschrieben.

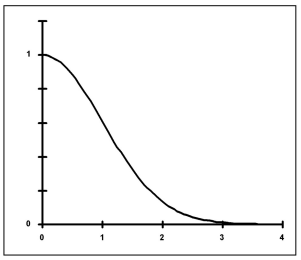


Abbildung 3.1: Die Transformationsfunktion $f(d_{ij}^l) = \exp\left(-\frac{1}{2}(d_{ij}^l)^2\right)$

Ähnlichkeitsmaß für qualitative Attribute

Für den Fall eines qualitativen Attributs A_l ist das Ähnlichkeitsmaß s^l definiert durch

$$s^l(X_i, X_j) = s_{ij}^l := \begin{cases} 1, & falls\ x_{il} = x_{jl} \\ 0, & falls\ x_{il} \neq x_{jl} \end{cases}$$

Es wird also überprüft, ob die beiden Alternativen x_{il} und x_{jl} übereinstimmen. Im ersten Fall bedeutet dies maximale Ähnlichkeit der beiden Datensätze (bezüglich dieses Attributs), im zweiten Fall minimale Ähnlichkeit.

Ähnlichkeitsmaß für quantitative Attribute

Betrachtet man bei den in Kapitel 2.3.2 vorgestellten Distanzmaßen für quantitative Attribute den Spezialfall eines einzelnen Attributs (welcher hier ja vorliegt), so ergibt sich als Distanz zweier Objekte $X_i,\ X_j$ jeweils der euklidische Abstand. Es ist daher naheliegend, dem Ähnlichkeitsmaß für quantitative Attribute das Distanzmaß $d_{ij}^l := |x_{il} - x_{jl}|$ zugrunde zu legen. Wie in Kapitel 2.3.1 erläutert, gibt es viele Möglichkeiten, aus einem Distanzmaß ein Ähnlichkeitsmaß zu generieren. Bei diesem Algorithmus wurde die Transformationsfunktion

$$f(d_{ij}^l) = exp\left(-\frac{1}{2}(d_{ij}^l)^2\right)$$

zugrunde gelegt. Sie hat den Vorteil, dass sie nahe bei null nur gering fällt, sodass man bei kleinen Differenzen immer noch Ähnlichkeiten erhält, die nahe bei eins liegen (siehe Abbildung 3.1). In die eigentliche Transformation fließt noch ein weiterer Parameter ein: die *Abstandseinheit δ_l des l-ten Attributs*, welche angibt, innerhalb welcher Distanz zwei Werte noch als ähnlich gelten. Die verwendete Transformation hat die Form

$$s_{ij}^l := exp\left(-\frac{1}{2} \cdot \left(\frac{\gamma \cdot d_{ij}^l}{\delta_l}\right)^2\right)$$

wobei der Faktor γ so gewählt wird, dass $s_{ij}^l = \frac{1}{2}$ gilt, falls $|x_{jl} - x_{il}| = \delta_l$. Um auf die zugrunde gelegte Form zu kommen, muss $\exp(\frac{1}{2}\gamma^2) = \frac{1}{2}$ bzw. $\gamma = \sqrt{2\ln 2}$ gelten.

Somit erhält man als Ähnlichkeitsmaß für quantitative Attribute

$$s_{ij}^l := \exp\left(-ln2 \cdot \left(\frac{d_{ij}^l}{\delta_l}\right)^2\right) \tag{3.3}$$

Die Abstandseinheit δ_l kann vom Benutzer festgelegt werden. Tut er dies nicht, so wird der Standardwert $\delta_l = \frac{1}{2}\sigma_l$ verwendet, wobei σ_l die empirische Standardabweichung der Attributwerte $x_{1l}, \ldots, x_{nl}$ des l-ten Attributs bezeichnet, also

$$\sigma_l := \sqrt{\frac{1}{n-1}\sum_{k=1}^{n}(x_{kl} - \bar{x}_l)^2} \qquad \text{wo} \quad \bar{x}_l := \frac{1}{n}\sum_{k=1}^{n} x_{kl}$$

3.1.3 Funktionsweise

Der Demographic Clustering Algorithmus geht so vor, dass er nach Bestimmung einer Anfangspartition (was noch genauer erläutert wird) alle Datensätze erneut durchläuft. Dabei wird für jeden Datensatz geprüft, ob die oben hergeleitete Zielfunktion (3.1) einen höheren Wert annimmt, wenn der Datensatz in ein anderes Cluster verschoben wird. Es wird auch die Möglichkeit geprüft, mit dem Datensatz ein neues Cluster zu gründen. Wird für eine der Möglichkeiten eine Verbesserung der Zielfunktion erreicht, so wird der Datensatz in das Cluster verschoben, für das der Wert der Zielfunktion am höchsten ist (bzw. es wird mit diesem Datensatz ein neues Cluster gebildet). Dies wird so lange iteriert, bis keine Verbesserung der Zielfunktion mehr erreicht wird.
Die Bestimmung der Anfangspartition erfolgt sehr ähnlich: die Datensätze werden der Reihe nach durchlaufen. Der erste Datensatz bildet das erste Cluster. Der zweite Datensatz wird entweder in dieses Cluster eingefügt oder er bildet ein neues Cluster, je nachdem, für welche Anordnung der Wert der Zielfunktion höher ist. Mit den verbleibenden Datensätzen wird analog verfahren: jeder Datensatz wird in jenes der bereits vorhandenen Cluster eingefügt, für das die Zielfunktion den höchsten Wert annimmt oder es wird ein neues Cluster mit ihm gebildet, falls der Wert der Zielfunktion dabei noch größer wird.
Man hat also folgende zwei Algorithmen:

Algorithmus 1 Berechnung der Anfangspartition

Eingabe: Datenmenge $\mathcal{M}$, Optimalitätskriterium z

$C := \emptyset$
for all Datensätze $X_q \in \mathcal{M}$ **do**
  for all k bereits gebildete Cluster $C_i \in C$ **do**
    berechne $z(C)$ neu unter der Annahme $C_i := C_i \cup \{X_q\}$
    betrachte auch die Möglichkeit, mit X_q ein neues Cluster zu bilden,
    also $C := C \cup \{\{X_q\}\}$, und berechne auch hierfür $z(C)$
  end for
  wähle aus den $k+1$ möglichen Clusterungen diejenige, für die der Wert
  von $z(C)$ am größten ist
end for

Ausgabe: Anfangspartition C

Algorithmus 2 Optimierung der Anfangspartition

Eingabe: Anfangspartition C, Optimalitätskriterium z

while
  for all Datensätze $X_q \in \mathcal{M}$ **do**
    entferne X_q aus seinem bisherigen Cluster
    for all k bereits gebildete Cluster $C_i \in C$ **do**
      berechne $z(C)$ neu unter der Annahme $C_i := C_i \cup \{X_q\}$
      betrachte auch die Möglichkeit, mit X_q ein neues Cluster zu bilden, also $C := C \cup \{\{X_q\}\}$, und berechne auch hierfür $z(C)$
    end for
    wähle aus den $k+1$ möglichen Clusterungen diejenige, für die der Wert von $z(C)$ am größten ist
  end for
  falls (nach einem Durchlauf durch alle Daten) keine Verbesserung für $z(C)$ erzielt wird (oder ein anderes Abbruchkriterium erfüllt ist), brich ab, andernfalls fahre fort
end while

Ausgabe: Optimierte Partition C

Die Funktionsweise der Algorithmen soll an einem Beispiel verdeutlicht werden. Gegeben sei eine Datenmenge $\mathcal{M} = \{X_1, X_2, X_3, X_4\}$ mit folgender Ähnlichkeitsmatrix:

$$S = \begin{pmatrix} 1 & \frac{3}{5} & \frac{1}{2} & \frac{2}{5} \\ \frac{3}{5} & 1 & \frac{4}{5} & \frac{1}{3} \\ \frac{1}{2} & \frac{4}{5} & 1 & \frac{1}{5} \\ \frac{2}{5} & \frac{1}{3} & \frac{1}{5} & 1 \end{pmatrix}$$

Die Ähnlichkeitsschranke sei $\alpha = \frac{1}{2}$, die verwendete Zielfunktion die oben hergeleitete Funktion $z(C) := \sum_{\substack{i,j \text{ im gleichen} \\ \text{Cluster, } i \neq j}} (s_{ij} - \alpha)$. Die Datensätze werden entsprechend der Reihenfolge ihrer Indizes abgearbeitet.

1. Berechnung der initialen Partition (Algorithmus 1)

   - X_1 bildet das erste Cluster, also $C = \{\{X_1\}\},\ z(C) = 0$
   - Für das Einfügen von X_2 gibt es zwei Möglichkeiten:

$C = \{\{X_1, X_2\}\}$	$C = \{\{X_1\}, \{X_2\}\}$
$z(C) = 2(\frac{3}{5} - \frac{1}{2}) = \frac{1}{2}$	$z(C) = 0$

     Da der Wert für z bei der ersten Möglichkeit größer ist, erhält man $C = \{\{X_1, X_2\}\}$.

- Für das Einfügen von X_3 gibt es wiederum zwei Möglichkeiten:

$C = \{\{X_1, X_2, X_3\}\}$	$C = \{\{X_1, X_2\}, \{X_3\}\}$
$z(C) = 2(\frac{3}{5} - \frac{1}{2}) + 2(\frac{1}{2} - \frac{1}{2}) + 2(\frac{4}{5} - \frac{1}{2}) = \frac{4}{5}$	$z(C) = \frac{1}{5}$

  Wiederum ist der erste Wert größer und man erhält $C = \{\{X_1, X_2, X_3\}\}$.

- Da bisher nur ein Cluster existiert, gibt es auch für X_4 nur zwei Möglichkeiten:

$C = \{\{X_1, X_2, X_3, X_4\}\}$	$C = \{\{X_1, X_2, X_3\}, \{X_4\}\}$
$z(C) = 2(\frac{3}{5} - \frac{1}{2}) + 2(\frac{1}{2} - \frac{1}{2}) + 2(\frac{2}{5} - \frac{1}{2})$ $+2(\frac{4}{5} - \frac{1}{2})2(\frac{1}{3} - \frac{1}{2}) + 2(\frac{1}{5} - \frac{1}{2}) = -\frac{1}{3}$	$z(C) = \frac{4}{5}$

  Somit erhält man als Anfangspartition $C = \{\{X_1, X_2, X_3\}, \{X_4\}\}$.

2. Optimierung der Anfangspartition (Algorithmus 2)

   - X_1 wird aus der initialen Partition entfernt. Es bestehen zwei Möglichkeiten, X_1 wieder einzufügen:

$C = \{\{X_2, X_3\}, \{X_1, X_4\}\}$	$C = \{\{X_1\}, \{X_2, X_3\}, \{X_4\}\}$
$z(C) = 2(\frac{4}{5} - \frac{1}{2}) + 2(\frac{2}{5} - \frac{1}{2}) = \frac{2}{5}$	$z(C) = 2(\frac{4}{5} - \frac{1}{2}) = \frac{3}{5}$

     Da der Wert der Zielfunktion durch Verschieben von X_1 nicht erhöht werden kann, wird die Clusterung $C = \{\{X_1, X_2, X_3\}, \{X_4\}\}$ beibehalten.

   - Nun wird X_2 entfernt. Es bestehen wiederum zwei Möglichkeiten der Neuzuordnung:

$C = \{\{X_1, X_3\}, \{X_2, X_4\}\}$	$C = \{\{X_1, X_3\}, \{X_2\}, \{X_4\}\}$
$z(C) = 2(\frac{1}{2} - \frac{1}{2}) + 2(\frac{1}{3} - \frac{1}{2}) = -\frac{1}{3}$	$z(C) = 2(\frac{1}{2} - \frac{1}{2}) = 0$

     Auch hier kann keine Verbesserung erzielt werden, weswegen die Clusterung $C = \{\{X_1, X_2, X_3\}, \{X_4\}\}$ beibehalten wird.

   - Nun wird X_3 entfernt. Auch hier bestehen zwei Möglichkeiten der Neuzuordnung:

$C = \{\{X_1, X_2\}, \{X_3, X_4\}\}$	$C = \{\{X_1, X_2\}, \{X_3\}, \{X_4\}\}$
$z(C) = 2(\frac{3}{5} - \frac{1}{2}) + 2(\frac{1}{5} - \frac{1}{2}) = -\frac{2}{5}$	$z(C) = 2(\frac{3}{5} - \frac{1}{2}) = \frac{1}{5}$

     Es wird erneut keine Verbesserung erzielt, weswegen die Clusterung $C = \{\{X_1, X_2, X_3\}, \{X_4\}\}$ beibehalten wird.

   - Für die Verschiebung von X_4 besteht nur eine Möglichkeit, welche jedoch bereits bei der Berechnung der initialen Partition betrachtet wurde.

Als optimale Clusterung erhält man also $C = \{\{X_1, X_2, X_3\}, \{X_4\}\}$. Betrachtet man alle Clusterungen (was bei 4 Datensätzen noch realisierbar ist), so sieht man, dass

dies auch die beste Clusterung ist. Dies muss nicht unbedingt der Fall sein, da eine andere Eingabe-Reihenfolge der Datensätze zu einem anderen Ergebnis führen kann. In diesem Fall ist bereits die Anfangspartition die beste, sodass bei der Optimierung keine Verbesserung mehr erzielt werden kann und Algorithmus 2 nach einem Durchlauf durch die Daten abbricht.

3.2 Laufzeit

3.2.1 Analyse

Im Folgenden soll nun der Rechenaufwand sowie der Speicherbedarf für die beiden auf Seite 69 vorgestellten Algorithmen abgeschätzt werden.
Die Berechnung der initialen Partition durch Algorithmus 1 entspricht einem Iterationsschritt von Algorithmus 2, weswegen sich seine Laufzeit durch die von Algorithmus 2 abschätzen lässt. Bei Algorithmus 2 muss in einem Iterationsschritt (d.h. einem Durchlauf durch alle Datensätze) für jeden Datensatz das Optimalitätskriterium (3.1) $(N+1)$-mal berechnet werden, wobei N die aktuelle Anzahl der Cluster ist. Sehr aufwändig ist hierbei die Berechnung der Ähnlichkeiten zwischen den Datensätzen, da in jedem Iterationsschritt alle Ähnlichkeiten s_{ij} für $i \neq j$ berechnet werden müssen. Es ist naheliegend, nicht für jede Veränderung der Cluster-Struktur (Entfernen eines Datensatzes aus einem Cluster / Hinzufügen eines Datensatzes zu einem Cluster / Bildung eines neuen Clusters mit dem aktuell betrachteten Datensatz) das Optimalitätskriterium neu zu berechnen, sondern geeignete Update-Formeln zu verwenden:
Sei $C := \{C_1, \ldots, C_k\}$ die aktuelle Clusterung.

- Durch das Hinzufügen eines Datensatzes $X_r \notin C$ zu einem Cluster $C_q \in C$ erhält man die Clusterung $\bar{C} = \{C_1, \ldots, C_q \cup \{X_r\}, \ldots, C_k\}$. Dann ändert sich der Wert des Optimalitätskriteriums folgendermaßen:

$$\begin{aligned}
\Delta z := z(\bar{C}) - z(C) &\overset{(3.1)}{=} \left(\sum_{\substack{i,j \text{ im gleichen Clus-} \\ \text{ter, } i \neq j,\ i,j \in \bar{C}}} s_{ij} \right) - \alpha \sum_{C_i \in \bar{C}} n_i(n_i - 1) \\
&\quad - \left(\sum_{\substack{i,j \text{ im gleichen Clus-} \\ \text{ter, } i \neq j,\ i,j \in C}} s_{ij} \right) + \alpha \sum_{C_i \in C} n_i(n_i - 1) \\
&= \sum_{i \in C_q} s_{ri} + \sum_{i \in C_q} s_{ir} - \alpha(n_q + 1)n_q + \alpha n_q(n_q - 1) \\
&= 2 \left(\left(\sum_{i \in C_q} s_{ri} \right) - \alpha n_q \right)
\end{aligned}$$

- Durch das Entfernen eines Datensatzes X_r aus seinem Cluster C_q erhält man die Clusterung $\bar{C} = \{C_1, \ldots, C_q - \{X_r\}, \ldots, C_k\}$. Dann ändert sich der Wert

des Optimalitätskriteriums folgendermaßen:

$$\begin{aligned}\Delta z := z(\bar{C}) - z(C) &\overset{(3.1)}{=} \left(\sum_{\substack{i,j \text{ im gleichen Clus-} \\ \text{ter, } i \neq j,\ i,j \in \bar{C}}} s_{ij} \right) - \alpha \sum_{C_i \in \bar{C}} n_i(n_i - 1) \\ &\quad - \left(\sum_{\substack{i,j \text{ im gleichen Clus-} \\ \text{ter, } i \neq j,\ i,j \in C}} s_{ij} \right) + \alpha \sum_{C_i \in C} n_i(n_i - 1) \\ &= - \sum_{i \in C_q - \{r\}} s_{ri} - \sum_{i \in C_q - \{r\}} s_{ir} - \alpha(n_q - 1)(n_q - 2) + \alpha n_q(n_q - 1) \\ &= 2 \left(\left(- \sum_{i \in C_q - \{r\}} s_{ri} \right) - \alpha(n_q - 1) \right)\end{aligned}$$

Besteht das Cluster C_q nur aus einem Datensatz (nämlich X_r), so wird Δz null, was bedeutet, dass sich der Wert des Optimalitätskriteriums beim Entfernen eines solchen Clusters nicht ändert.

- Durch Bilden eines neuen Clusters $C_{k+1} := \{X_r\}$ mit einem Datensatz $X_r \notin C$ erhält man die Clusterung $\bar{C} = \{C_1, \ldots, C_k, C_{k+1}\}$. Das Bilden des neuen Clusters kann man als Hinzufügen des Datensatzes X_r zu dem Cluster $C_{k+1} = \emptyset$ betrachten, worauf dann der erste Fall anwendbar ist: Δz wird null, d.h. man kann ein Cluster mit einem einzigen Datensatz hinzufügen, ohne ein Update des Optimalitätskriteriums durchführen zu müssen.

Trotz der Verwendung von Update-Formeln lässt sich das Berechnen sämtlicher Ähnlichkeiten $s_{ij},\ i \neq j$, nicht vermeiden: zwar müssen bei einem Update nur die Ähnlichkeiten des aktuell betrachteten Datensatzes zu den Datensätzen eines einzelnen Clusters berechnet werden (nämlich des Clusters, aus dem der Datensatz entfernt bzw. in das der Datensatz eingefügt wird), aber das Optimalitätskriterium muss $(k+1)$-mal neu berechnet werden (wobei k die aktuelle Clusteranzahl ist) bevor ein Datensatz einem Cluster zugeordnet werden kann. Folglich muss auch das Update $(k+1)$-mal durchgeführt werden.
Berechnet man sämtliche Ähnlichkeiten alle zu Beginn und speichert sie für das weitere Verfahren, so ergibt sich eine Laufzeit von $\mathcal{O}(n^2)$ und auch der Speicherbedarf ist quadratisch in der Anzahl der Datensätze. Berechnet man die Ähnlichkeitsmatrix erst während des Verfahrens ohne sie zu speichern, so spart man zwar den Speicherplatz, muss aber die Matrix bei jeder Iteration neu berechnen; der zeitliche Aufwand bleibt hierbei quadratisch. Für ein Cluster-Analyse-Verfahren, das im Rahmen eines Data-Mining-Projektes eingesetzt werden soll, ist diese quadratische Laufzeit ein k.o.-Kriterium, da in der Praxis sehr große Datenmengen bearbeitet werden müssen. Man kann jedoch trotzdem eine lineare Laufzeit erzielen, indem man eine spezielle Datenstruktur für die Cluster einführt, was im folgenden Abschnitt erläutert wird. Die Verwendung dieser Datenstruktur erfordert keine Änderungen an der grundsätzlichen Funktionsweise der Algorithmen 1 und 2.

3.2.2 Laufzeitverbesserung durch spezielle Datenstruktur

Wie im obigen Abschnitt beschrieben, führt die Berechnung aller Ähnlichkeiten $s_{ij},\ i \neq j$, zu einer quadratischen Laufzeit des Verfahrens. Hier soll nun gezeigt werden, wie für einen Datensatz X_j die für das Optimalitätskriterium bzw. die Update-Formeln benötigten Summen

$$\sum_{\substack{i \in C_q \\ j \notin C_q}} s_{ij} \tag{3.4}$$

für $q = 1, \ldots, k$ effizient berechnet werden können. Da das verwendete Ähnlichkeitsmaß s aggregiert ist aus den Ähnlichkeitsmaßen $s^l (l = 1, \ldots, p)$ für die einzelnen Attribute, lässt sich die Summe (3.4) auf die Summe über die Ähnlichkeiten s^l_{ij} der einzelnen Attribute zurückführen:
Sei $C_q \in C,\ X_j \notin C_q$. Dann ist

$$\sum_{i \in C_q} s_{ij} \overset{(3.2)}{=} \sum_{i \in C_q} \frac{1}{\sum_{l=1}^{p} w_l} \sum_{l=1}^{p} w_l \cdot s^l_{ij} = \frac{1}{\sum_{l=1}^{p} w_l} \sum_{l=1}^{p} w_l \cdot \sum_{i \in C_q} s^l_{ij} \tag{3.5}$$

Die Summe der Gewichte ist unproblematisch, da es sich um einen konstanten Wert handelt, der nur einmal berechnet werden muss. Der hohe Rechenaufwand wird durch die letzte Summe verursacht. Gesucht ist also eine effiziente Berechnung der Summen

$$\sum_{i \in C_q} s^l_{ij}\ ,\ l \in \{1, \ldots, p\} \tag{3.6}$$

für $q = 1, \ldots, k$ und $X_j \notin C_q$. Hierbei geht man bei den verschiedenen Attributtypen unterschiedlich vor:

Qualitative Attribute

Sei A_l ein qualitatives Attribut mit den a_l verschiedenen Alternativen $\alpha_{l1}, \ldots, \alpha_{la_l}$. Zu einem Cluster $C_q \in C$ seien $h_{C_q}(\alpha_{l1}), \ldots, h_{C_q}(\alpha_{la_l})$ die absoluten Häufigkeiten, mit denen die einzelnen Alternativen im Cluster C_q auftreten. Dann lässt sich obige Summe (für $X_j \notin C_q$) folgendermaßen umformen:

$$\sum_{i \in C_q} s^l_{ij} = \sum_{k=1}^{a_l} s^l(x_{jl}, \alpha_{lk}) \cdot h_{C_q}(\alpha_{lk}) = h_{C_q}(x_{jl}) = total(A_l, C_q) \cdot \underbrace{\frac{h_{C_q}(x_{jl})}{total(A_l, C_q)}}_{=: p_{C_q}(x_{jl})} \tag{3.7}$$

Die zweite Gleichheit gilt, da $s^l(x_{jl}, \alpha_{lk}) = s^l(x_{jl}, x_{lk})$, was bedeutet, dass alle Summanden mit $x_{jl} \neq \alpha_{lk}$ null sind. (Bei qualitativen Attributen beträgt die Ähnlichkeit bei Übereinstimmung der Werte 1, ansonsten 0 (siehe Kap. 3.1.2)). $p_{C_q}(x_{jl})$ ist die *relative Häufigkeit* des Attributwerts x_{jl}, welche ein Schätzer für die Wahrscheinlichkeit ist, dass die Alternative x_{jl} im Cluster C_q auftritt. $total(A_l, C_q)$ zählt alle gültigen Werte, die für das Attribut A_l in Cluster C_q auftreten:

$$total(A_l, C_q) := \sum_{k=1}^{n_l} h_{C_q}(\alpha_{lk}) \leq n_q$$

A_1 :	$h_{C_q}(\alpha_{11})$	$h_{C_q}(\alpha_{12})$	$\cdots$	$h_{C_q}(\alpha_{1a_1})$	$total(A_1, C_q) \leq n_q$
A_2 :	$h_{C_q}(\alpha_{21})$	$h_{C_q}(\alpha_{22})$	$\cdots$	$h_{C_q}(\alpha_{2a_2})$	$total(A_2, C_q) \leq n_q$
	$\vdots$				$\vdots$
A_p :	$h_{C_q}(\alpha_{p1})$	$h_{C_q}(\alpha_{p2})$	$\cdots$	$h_{C_q}(\alpha_{pa_p})$	$total(A_p, C_q) \leq n_q$

Abbildung 3.2: Cluster C_q mit Statistik Σ_{C_q} (nur qualitative Attribute)

Um die Summe (3.6) für qualitative Attribute zu berechnen, benötigt man also keine mengenbasierte Darstellung des Clusters, sondern es genügt, für jedes (qualitative) Attribut A_l die absoluten Häufigkeiten $h_{C_q}(\alpha_{l1}), \ldots, h_{C_q}(\alpha_{la_l})$ zu speichern und beim Hinzufügen oder Entfernen eines Datensatzes zu aktualisieren. Auf diese Weise kann die Summe (3.6) sogar mit konstantem Zeitaufwand berechnet werden.
Sind alle Attribute qualitativ, so hat ein Cluster die in Abbildung 3.2 dargestellte Struktur.

Quantitative Attribute

Bei stetigen und diskret numerischen Attributen (aus denen sich die qualitativen Attribute zusammensetzen, siehe Kap. 2.2.5) lässt sich die Summe (3.6) nicht so einfach wie bei den qualitativen Attributen mit Hilfe der absoluten Häufigkeiten der Merkmalsausprägungen berechnen, da verschiedene Merkmalswerte meist eine von null verschiedene Ähnlichkeit besitzen. Bisher sind noch keine Verfahren bekannt, mit denen (3.6) für quantitative Attribute exakt berechnet werden kann, ohne zuvor sämtliche Ähnlichkeiten zu berechnen, was zu einer quadratischen Laufzeit führen würde. Man kann aber gute Näherungen für die Summe erzielen, was im Folgenden für stetige Merkmale gezeigt wird. Das Vorgehen bei diskret numerischen Attributen lässt sich dann auf stetige Attribute zurückführen.

Stetige Attribute

Man kann ein stetiges Attribut A_l im Cluster C_q als Zufallsvariable $X^l_{C_q}$ auffassen, die eine Dichtefunktion $f^l_{C_q}(t)$ mit $\int_{-\infty}^{\infty} f^l_{C_q}(t)dt = 1$ besitzt. Dann kann auch die Funktion $g(X^l_{C_q}) := s^l(x_{jl}, X^l_{C_q})$ für ein festes x_{jl} als eine Zufallsvariable betrachtet werden. Des weiteren kann man die Attributwerte $x_{1l}, \ldots, x_{\bar{n}_q l}$ ($\bar{n}_q \leq n_q = |C_q|$), die das Merkmal im Cluster C_q annimmt, als Stichprobe der Zufallsvariable $X^l_{C_q}$ auffassen. Es seien $\alpha_{l1}, \ldots, \alpha_{ln_l}$ die tatsächlich in den Daten auftretenden $n_l \in \mathbb{N}$ verschiedenen Werte von A_l (die so genannten *tatsächlichen Alternativen*). Mittels der absoluten Häufigkeiten $h_{C_q}(\alpha_{l1}), \ldots, h_{C_q}(\alpha_{la_l})$ und der Gesamtzahl der gültigen im Cluster C_q angenommenen Werte lassen sich die relativen Häufigkeiten $p_{C_q}(\alpha_{l1}), \ldots, p_{C_q}(\alpha_{la_l})$ der tatsächlichen Alternativen berechnen. Wenn man nun $X^l_{C_q}$ (und somit auch $g(X^l_{C_q})$) als diskret verteilte Zufallsvariable auffasst, so lässt sich die Summe (3.6) analog zu

(3.7) umformen:

$$\sum_{i \in C_q} s_{ij}^l = \sum_{k=1}^{a_l} s^l(x_{jl}, \alpha_{lk}) \cdot h_{C_q}(\alpha_{lk}) = total(A_l, C_q) \cdot \sum_{k=1}^{a_l} s^l(x_{jl}, \alpha_{lk}) \cdot \underbrace{\frac{h_{C_q}(x_{jl})}{total(A_l, C_q)}}_{=:p_{C_q}(x_{jl})}$$

$$= total(A_l, C_q) \cdot E_{C_q}(g(X_{C_q}^l))$$

Bei der zweiten Umformung wurde die Definition des Erwartungswertes für diskrete Zufallsvariablen verwendet (siehe [Bos98], S.134). Da sich die Summe nicht soweit wie bei qualitativen Attributen vereinfachen lässt, lohnt sich die Umformung nur, wenn die Zahl der tatsächlichen Alternativen deutlich kleiner ist als die Zahl der gültigen Werte für das Attribut A_l, wenn also $n_l \ll total(A_l, C_q)$ gilt. Da dies bei stetigen Attributen meist nicht der Fall ist, bringt die Umformung keinen geringeren Rechenaufwand als die direkte Berechnung der Summe (3.6). Jedoch ist der obige Erwartungswert ein Schätzer für den Erwartungswert der stetigen Zufallsvariable $s^l(x_{jl}, X_{C_q}^l)$, der definiert ist als

$$E_{C_q}(s^l(x_{jl}, X_{C_q}^l)) = \int\limits_{X_{C_q}^l} s^l(x_{jl}, \cdot) dP_{X_{C_q}^l}$$

Mit Hilfe des Transformationssatzes für Integrale ergibt sich dann

$$E_{C_q}(s^l(x_{jl}, X_{C_q}^l)) = \int\limits_{\Omega_{C_q}} s^l(x_{jl}, \cdot) \circ X_{C_q}^l dP_{C_q}$$

$$= \int\limits_{-\infty}^{\infty} s^l(x_{jl}, t) f_{C_q}^l(t) dt \overset{(3.3)}{\leq} \int\limits_{-\infty}^{\infty} 1 \cdot f_{C_q}^l(t) dt = 1 < \infty$$

Der Erwartungswert von $s^l(x_{jl}, X_{C_q}^l)$ existiert also, weswegen man die Summe (3.6) folgendermaßen approximieren kann:

$$\sum_{i \in C_q} s_{ij}^l \approx total(A_l, C_q) \cdot \int\limits_{-\infty}^{\infty} s^l(x_{jl}, t) f_{C_q}^l(t) dt \tag{3.8}$$

Die Dichtefunktion $f_{C_q}^l(t)$ von $X_{C_q}^l$ ist nicht bekannt, sie kann jedoch durch die so genannte *Histogramm-Funktion* approximiert werden. Eine Histogramm-Funktion ist eine auf einer Stichprobe basierende Treppenfunktion, deren Funktionswert in einem Treppen-Intervall mit Hilfe der Anzahl der Werte der Stichprobe, die in diesem Intervall liegen, so bestimmt wird, dass die Fläche unter der Histogramm-Funktion ebenso wie bei der Dichtefunktion eins ergibt. Abbildung 3.3 zeigt das Beispiel einer Histogramm-Funktion einer normalverteilten Zufallsvariable.
Für die Abschätzung wird zunächst ein Intervall $[r_{min,l}, r_{max,l})$ im Wertebereich von $X_{C_q}^l$ bestimmt, was z.B. mit Hilfe der 2-Sigma-Regel geschehen kann, welche besagt, dass alle Werte der Stichprobe, die um mehr als zwei Standardabweichungen vom empirischen Erwartungswert abweichen, ignoriert werden. Die Histogramm-Funktion $H_{C_q}^l(t)$ ist also außerhalb des Intervalls $[r_{min,l}, r_{max,l})$ konstant null. Anschließend

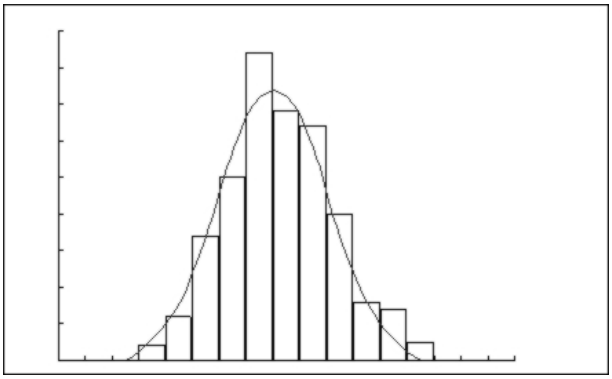


Abbildung 3.3: Histogramm-Funktion einer normalverteilten Zufallsvariable

wird dieses Intervall in $m \in \mathbb{N}$ Teilintervalle $[r_{i-1,l}, r_{i,l}),\ (i = 1, \ldots, m)$, so genannte *Buckets,* unterteilt. Der Wert $H_{C_q}^{i,l}$ der Histogramm-Funktion $H_{C_q}^{l}$ auf dem Bucket $[r_{i-1,l}, r_{i,l})$ berechnet sich dann durch

$$H_{C_q}^{i,l} := \frac{h_{C_q}([r_{i-1,l}, r_{i,l}))}{total(A_l, C_q) \cdot (r_{i,l} - r_{i-1,l})},\ i = 1, \ldots, m$$

wobei die absolute Häufigkeit $h_{C_q}([r_{i-1,l}, r_{i,l}))$ definiert ist als

$$h_{C_q}([r_{i-1,l}, r_{i,l})) := |\{X_j \in C_q | r_{i-1,l} \leq x_{jl} < r_{i,l}\}|,\ i = 1, \ldots, m$$

Wegen

$$\begin{aligned}
\int_{-\infty}^{\infty} H_{C_q}^{l}(t)dt &= \int_{r_{min,l}}^{r_{max,l}} H_{C_q}^{l}(t)dt \\
&= \int_{r_{min,l}}^{r_{1,l}} H_{C_q}^{1,l}(t)dt + \int_{r_{1,l}}^{r_{2,l}} H_{C_q}^{2,l}(t)dt + \ldots + \int_{r_{m-1,l}}^{r_{max,l}} H_{C_q}^{m,l}(t)dt \\
&= \frac{h_{C_q}([r_{min,l}, r_{1,l})) + h_{C_q}([r_{1,l}, r_{2,l})) + \ldots + h_{C_q}([r_{m-1,l}, r_{max,l}))}{total(A_l, C_q)} = 1
\end{aligned}$$

ist $H_{C_q}^{l}(t)$ eine stückweise stetige Dichtefunktion, sodass man (3.8) folgendermaßen

umformen kann:

$$
\begin{aligned}
(3.8) &\approx total(A_l, C_q) \cdot \int_{-\infty}^{\infty} s^l(x_{jl}, t) \cdot H^l_{C_q}(t)dt \\
&= total(A_l, C_q) \cdot \left(\int_{r_{min,l}}^{r_{1,l}} s^l(x_{jl}, t) \cdot H^l_{C_q}(t)dt + \ldots + \int_{r_{m-1,l}}^{r_{max,l}} s^l(x_{jl}, t) \cdot H^l_{C_q}(t)dt \right) \\
&= total(A_l, C_q) \cdot \sum_{i=1}^{m} H^{i,l}_{C_q} \cdot \int_{r_{i-1,l}}^{r_{i,l}} s^l(x_{jl}, t)dt \\
&= \sum_{i=1}^{m} \frac{h_{C_q}([r_{i-1,l}, r_{i,l}))}{r_{i,l} - r_{i-1,l}} \cdot \int_{r_{i-1,l}}^{r_{i,l}} \exp\left(-\ln 2 \cdot \left(\frac{|x_{jl} - t|}{\delta_l} \right)^2 \right) dt
\end{aligned}
\tag{3.9}
$$

Durch weitere Umformungen kann man das letzte Integral auf die Form

$$P(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} \exp\left(-\frac{1}{2}t^2 \right) dt$$

bringen, welches ein tabelliertes Integral ist (z.B. in [AS68], S.966-972). Man kann dann bei der Initialisierungsphase des Algorithmus entweder die Tabellierungen in den Hauptspeicher laden oder das Integral mit Hilfe einer geeigneten Näherungsformel berechnen, wobei der Fehler kleiner ist als $2,5 \cdot 10^{-4}$ [AS68].
Speichert man also für die stetigen Attribute die absoluten Häufigkeiten $h_{C_q}([r_{i-1,l}, r_{i,l}))$, $i = 1, \ldots, m$, so kann man die Summe (3.6) mit einem Zeitaufwand von $\mathcal{O}(m)$ näherungsweise berechnen. m ist stets sehr klein, sodass der Aufwand als konstant betrachtet werden kann.

Diskret numerische Attribute

Der Fall der diskret numerischen Attribute lässt sich unter Verwendung von *Deltafunktionen* auf den Fall der stetigen Attribute zurückführen.

Definition 3.2.1 (Deltafunktion) *Eine uneigentliche Funktion $\delta : \mathbb{R} \to \mathbb{R}$ heißt* Deltafunktion, *wenn für ein festes $a \in \mathbb{R}$ gilt, dass $\delta(x - a) = 0$ für $x \neq a$ und wenn die Funktion an der Stelle $x = a$ in der Weise singulär ist, dass $\int_{-\infty}^{\infty} \delta(x - a)dx = 1$ gilt.*

Seien $\alpha_{l1}, \ldots, \alpha_{la_l}$ die a_l diskret numerischen Werte, die das (diskret numerische) Attribut A_l auf dem Cluster C_q annimmt. Dann besitzt A_l die so genannte *Zähldichte*

$$f^l_{C_q}(t) = \begin{cases} p_{C_q}(\alpha_{li}), & \text{falls } t = \alpha_{li},\ i = 1, \ldots, n_l \\ 0 & \text{sonst} \end{cases}$$

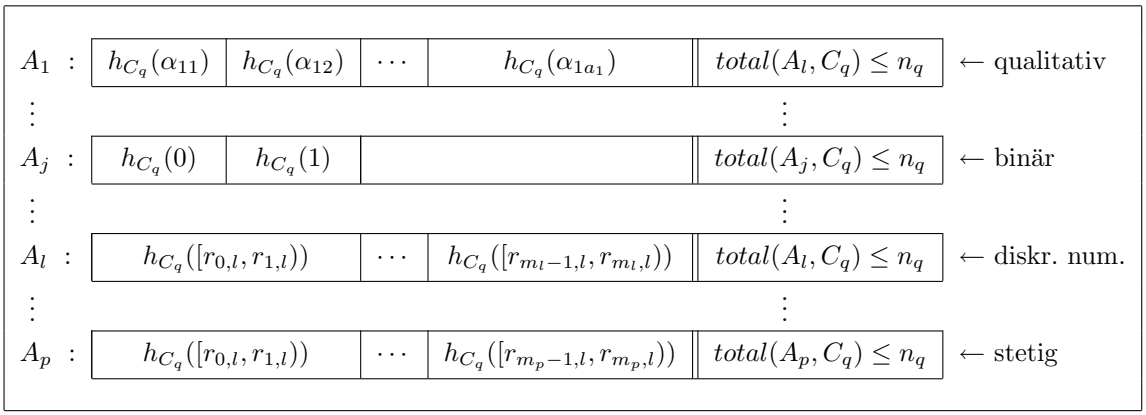


Abbildung 3.4: Cluster C_q mit Statistik Σ_{C_q}

und es gilt $\sum_{i=1}^{n_l} p_{C_q}(\alpha_{li}) = 1$. Wegen letzterem ist die Bedingung

$$\int_{-\infty}^{\infty} \sum_{i=1}^{n_l} p_{C_q}(\alpha_{li})\delta(t - \alpha_{li})dt = 1,$$

erfüllt, sodass folgende Darstellung gültig ist:

$$f_{C_q}^l(t) = \sum_{i=1}^{n_l} p_{C_q}(\alpha_{li})\delta(t - \alpha_{li}).$$

Nun kann man analog zu den stetigen Attributen vorgehen und die obige Zähldichte mit Hilfe einer Histogramm-Funktion schätzen. Analog zu oben kann man dann auch die Summe (3.6) für diskret numerische Attribute mit konstantem Zeitaufwand approximieren. Auch hierfür ist keine mengenbasierte Darstellung der Cluster nötig, sondern es genügt, die absoluten Häufigkeiten der Werte in den einzelnen Buckets zu speichern.
Insgesamt sieht die Datenstruktur eines Clusters C_q also aus wie in Abbildung 3.4.

Zusammenfassung

Zu Beginn des Abschnitts wurde festgestellt, dass der zeitaufwändige Teil bei der Zuordnung eines Datensatzes X_j zu einem Cluster in der vorab nötigen Berechnung der Ähnlichkeiten des Datensatzes zu allen Clustern liegt, also in der Berechnung der Summen $\sum_{i \in C_q} s_{ij}$ für $q = 1, \ldots, k$.. Dieses Problem lässt sich auf das Problem der Berechnung sämtlicher Ähnlichkeiten in einem Attribut ($\sum_{i \in C_q} s_{ij}^l$) zurückführen (siehe Gleichung (3.5)). Für diese Summe wurde anschließend gezeigt, dass sie sich bei Verwendung einer speziellen Datenstruktur für qualitative Attribute exakt und für quantitative Attribute näherungsweise in konstanter Zeit berechnen lässt. Berechnet man die Ähnlichkeit eines Datensatzes zu einem Cluster, so muss diese Summe p-mal berechnet werden, wobei p die Anzahl der Attribute ist. Bevor ein Datensatz einem Cluster zugeordnet werden kann, muss er mit jedem Cluster verglichen werden,

was den Faktor $K :=$ maximale Cluster-Anzahl hinzubringt. Schließlich muss jeder Datensatz mindestens einmal betrachtet werden, um ihn einem Cluster zuordnen zu können. Insgesamt ergibt das eine Laufzeit von $\mathcal{O}(p \cdot K \cdot n)$. Da jedoch p und K immer sehr kleine Werte sind, insbesondere in Relation zu n (also $p \ll n,\ K \ll n$), erhält man eine Laufzeit, die linear in der Anzahl der Datensätze ist.

Kapitel 4

Optimierung der Laufzeit

4.1 Grundidee

Cluster-Analyse-Verfahren, bei denen die Cluster als Menge der in ihnen enthaltenen Datensätze repräsentiert werden, haben in der Regel eine quadratische Laufzeit, da zur Berechnung der Ähnlichkeit eines Datensatzes zu einem Cluster der Datensatz mit jedem der Datensätze im Cluster verglichen werden muss. Um einen Datensatz richtig zuordnen zu können, muss er mit jedem Cluster verglichen werden, des weiteren muss jeder Datensatz mindestens einmal betrachtet werden, um ihn zuordnen zu können. Insgesamt ist also mindestens ein Durchlauf durch alle Daten notwendig und bei jedem Durchlauf wird jeder Datensatz mit allen anderen verglichen, was zu einer Laufzeit von $\mathcal{O}(n^2)$ führt. Im vorigen Kapitel wurde gezeigt, dass man die Datensatz-Cluster-Vergleiche in konstanter Zeit durchführen kann, wenn man die Cluster nicht mengenbasiert speichert, sondern als Statistik über die im Cluster vorkommenden Attributwerte. Auf diese Weise erzielt man eine lineare Laufzeit. Da jeder Datensatz mindestens einmal betrachtet werden muss, um ihn einem Cluster zuordnen zu können, ist es wohl nicht möglich, eine Laufzeit in einer „besseren“ Komplexitätsklasse, etwa $\mathcal{O}(\log n)$, zu erzielen. Man kann aber eine bessere absolute Laufzeit erzielen, wenn man erreicht, dass ein einzelner Datensatz nicht mehr mit allen Clustern verglichen werden muss, wenn also die Schleife über alle Cluster früher abgebrochen werden kann. Ein Abbruch dieser Schleife ist dann möglich, wenn man weiß, dass der Wert des Optimalitätskriteriums nicht größer wird, wenn der aktuelle Datensatz in eines der noch nicht betrachteten Cluster eingefügt wird. Doch wie erhält man ein solches Abbruchkriterium?
In Kapitel 3.2 wurde gezeigt, dass man durch die Verwendung von Update-Formeln beim Einfügen eines Datensatzes X_j in ein Cluster C_q nicht mehr sämtliche Ähnlichkeiten $s_{ij}, i \neq j$, berechnen muss, sondern lediglich die Ähnlichkeiten zu den anderen Datensätzen im Cluster C_q. Möchte man das „beste“ Cluster für X_j finden, so muss man nach dem Cluster suchen, bei dem der Wert der Update-Formel

$$\Delta z := 2 \left(\left(\sum_{\substack{i \in C_q \\ i \neq j}} s_{ij} \right) - \alpha \cdot n_q \right) \ , \ q = 1, \ldots, k := \textit{aktuelle Clusteranzahl} \qquad (4.1)$$

maximal wird. Wenn es gelingt, diesen Wert für jedes Cluster unabhängig von dem aktuell betrachteten Datensatz X_j nach oben abzuschätzen, hat man das gewünschte Abbruchkriterium. Das weitere Vorgehen sieht dann folgendermaßen aus: man sortiert die Cluster entsprechend dieser oberen Schranke in absteigender Reihenfolge und jeder Datensatz wird zunächst mit dem Cluster mit der höchsten Schranke verglichen[1]. Ist der Wert, den die Update-Formel liefert, größer als die obere Schanke des Clusters mit der zweithöchsten Schranke, so kann die Schleife abgebrochen werden, da man dann weiß, dass kein anderes Cluster einen höheren Wert der Update-Formel liefern wird. Andernfalls wird so lange fortgefahren, bis die obere Schranke eines Clusters (in der sortierten Cluster-Folge) kleiner ist als das bisher gefundene Maximum der Update-Formel.

4.2 Methoden zur Bestimmung einer oberen Schranke

Um ein Abbruch-Kriterium für die Schleife über alle Cluster zu erhalten muss man die Summe (4.1) möglichst gut nach oben abschätzen. Die Clustergröße n_q braucht nicht abgeschätzt zu werden, da sie direkt aus der Cluster-Statistik abgelesen werden kann. Gesucht sind also Abschätzungen der Summen

$$\sum_{\substack{i \in C_q \\ i \neq j}} s_{ij}, \ q = 1, \ldots, k$$

für die k aktuellen Cluster unabhängig vom aktuell betrachteten Datensatz X_j. Wegen

$$\sum_{\substack{i \in C_q \\ i \neq j}} s_{ij} \overset{(3.2)}{=} \sum_{\substack{i \in C_q \\ i \neq j}} \frac{1}{\sum_{l=1}^{p} w_l} \sum_{l=1}^{p} w_l \cdot s_{ij}^l = \frac{1}{\sum_{l=1}^{p} w_l} \sum_{l=1}^{p} w_l \cdot \sum_{\substack{i \in C_q \\ i \neq j}} s_{ij}^l$$

lässt sich die Abschätzung der obigen Summe herunterbrechen auf die Attribut-Ebene, sodass also der entscheidende Faktor die Abschätzung der Summe

$$\sum_{\substack{i \in C_q \\ i \neq j}} s_{ij}^l \tag{4.2}$$

der Ähnlichkeiten bezüglich der einzelnen Attribute ist.
Das Ähnlichkeitsmaß ist für die verschiedenen Attributtypen unterschiedlich definiert, darum ist es sinnvoll, diese Fälle getrennt voneinander zu betrachten. Bevor auf die Abschätzung für die einzelnen Attributtypen näher eingegangen wird, soll zunächst noch eine vom Attributtyp unabhängige Möglichkeit der Abschätzung angegeben werden:

[1] „Vergleichen“ bedeutet in diesem Zusammenhang das gleiche wie „die Ähnlichkeit berechnen“, da man das probeweise Einfügen eines Datensatzes in ein Cluster auch als Vergleich des Datensatzes mit dem Cluster interpretieren kann.

4.2.1 Abschätzung für alle Attributtypen

Sei A_l ein beliebiges Attribut und X_j ein Datensatz, dessen Ähnlichkeit zum Cluster C_q berechnet werden soll (wobei $X_j \notin C_q$). Da man den Attributtyp nicht kennt, muss die Abschätzung unabhängig vom zugehörigen Ähnlichkeitsmaß sein. Eine erste grobe Abschätzung liefert die Clustergröße: Wenn man annimmt, dass X_j im Attribut A_l mit allen Datensätzen im Cluster übereinstimmt, so kann man (4.2) durch die Clustergröße n_q abschätzen, da in diesem Fall jeder Datensatz im Cluster den Beitrag eins zu der Summe leistet.
Um eine bessere Abschätzung von (4.2) machen zu können, ist es jedoch notwendig, für die verschiedenen Attributtypen unterschiedliche Abschätzungen zu machen.

4.2.2 Kategorische Attribute

Sei A_l ein kategorisches Attribut und X_j ein Datensatz, dessen Ähnlichkeit zum Cluster C_q berechnet werden soll (wobei $X_j \notin C_q$). Das für kategorische Attribute verwendete Ähnlichkeitsmaß (siehe Kap.3.1.2)

$$s^l(X_i, X_j) = s_{ij}^l := \begin{cases} 1, & falls\ x_{il} = x_{jl} \\ 0, & falls\ x_{il} \neq x_{jl} \end{cases}$$

erlaubt eine gute Abschätzung der Summe (4.2) für kategorische Attribute. Grund hierfür ist die Eigenschaft, dass zwei Attributwerte nur dann eine von null verschiedene Ähnlichkeit haben, wenn sie übereinstimmen. Eine sehr gute Abschätzung liefert folgender Ansatz: Wenn das Attribut A_l die Werte $\alpha_{l1}, \ldots, \alpha_{la_l}$ hat, so kann x_{jl} nur mit einem dieser Werte übereinstimmen. Im schlimmsten Fall ist das der Wert, der in Cluster C_q am häufigsten vorkommt, wenn also

$$x_{jl} = \alpha_l^{max},\ \text{wo}\ \alpha_l^{max} \in \{\alpha_{l1}, \ldots, \alpha_{la_l}\}\ \text{mit}\ h_{C_q}(\alpha_l^{max}) = \max_{k=1}^{a_l}\{h_{C_q}(\alpha_{lk})\}$$

gilt. Da X_j mit allen Datensätzen $X_i \in C_q$ mit $x_{il} = \alpha_l^{max}$ die Ähnlichkeit eins und mit allen anderen Datensätzen im Cluster die Ähnlichkeit null hat (bezüglich des l-ten Attributs), kann man (4.2) durch die Anzahl der Datensätze in C_q mit dem häufigsten Attributwert abschätzen:

$$\sum_{i \in C_q} s_{ij}^l \leq h_{C_q}(\alpha_l^{max})$$

Aufgrund der Datenstruktur der Cluster (siehe Kap.3.2.2) ist es nicht nötig, diesen Wert zu berechnen, er kann einfach aus der Clusterstatistik Σ_{C_q} abgelesen werden.
Sollen bei einem kategorischen Attribut zwei oder mehr Alternativen eine von null verschiedene Ähnlichkeit haben, so hat der Benutzer die Möglichkeit, eine Ähnlichkeitmatrix für dieses Attribut zu definieren. Ist dies der Fall, so kann man diese Methode der Abschätzung nicht anwenden. Man kann dann jedoch die im Folgenden beschriebenen Abschätzungen für diskret numerische Attribute verwenden.

4.2.3 Diskret numerische Attribute

Sei A_l ein diskret numerisches Attribut und X_j ein Datensatz, dessen Ähnlichkeit zum Cluster C_q berechnet werden soll (wobei $X_j \notin C_q$). Als Ähnlichkeitsmaß für diesen Attributtyp wurde

$$s_{ij}^l := \exp\left(-\ln 2 \cdot \left(\frac{d_{ij}^l}{\delta_l}\right)^2\right) \tag{4.3}$$

gewählt (siehe Kap.3.1.2). Hierbei ist eine gute Abschätzung aufwändiger als im kategorischen Fall, da berücksichtigt werden muss, dass verschiedene Attributwerte eine Ähnlichkeit größer null besitzen können. Unterschiedliche Abschätzungen sind denkbar:

1. Seien $\alpha_{l1}, \ldots, \alpha_{la_l}$ die verschiedenen Werte, die das Attribut A_l annimmt. Die Ähnlichkeit von x_{jl} zum Cluster C_q wird dann am größten, wenn der häufigste Wert α_l^{max}die größte Ähnlichkeit zu x_{jl} hat, der zweithäufigste Wert die zweitgrößte Ähnlichkeit usw. Sei

   $$\tau : \{1, \ldots, a_l\} \to \{1, \ldots, a_l\} \text{ eine Permutation mit}$$
   $$h_{C_q}(\alpha_{l\tau(1)}) \geq h_{C_q}(\alpha_{l\tau(2)}) \geq \cdots \geq h_{C_q}(\alpha_{l\tau(a_l)})$$

   Wenn man die (symmetrische) Ähnlichkeitsmatrix so betrachtet, dass in den Zeilen die Werte stehen, die x_{jl} annehmen kann und in den Spalten die Werte der Datensätze im Cluster,

		C_q			
		α_{l1}	α_{l2}	$\cdots$	α_{la_l}
	α_{l1}	1	s_{12}^l	$\cdots$	$s_{1a_l}^l$
	α_{l2}	s_{21}^l	1	$\cdots$	$s_{2a_l}^l$
x_{jl}	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
	α_{la_l}	$s_{a_l1}^l$	$s_{a_l2}^l$	$\cdots$	1

   so muss man für die Berechnung der oberen Schranke zeilenweise durch die Matrix gehen und für jede Zeile t die Summe

   $$\sum_{k=1}^{a_l} s_{t\sigma_t(k)}^l \cdot h_{C_q}(\alpha_{l\tau(k)}), \quad t = 1, \ldots, a_l$$

   berechnen, wobei

   $$\sigma_t : \{1, \ldots, a_l\} \to \{1, \ldots, a_l\} \text{ eine Permutation ist mit}$$
   $$1 = s_{t\sigma_t(1)}^l \geq s_{t\sigma_t(2)}^l \geq \cdots \geq s_{t\sigma_t(a_l)}^l$$

   und anschließend das Maximum über die so errechneten Werte bilden. Als obere Schranke für (4.2) erhält man somit:

$$\sum_{i \in C_q} s_{ij}^l \leq \max_{t=1}^{a_l} \left\{ \sum_{k=1}^{a_l} s_{t\sigma_t(k)}^l \cdot h_{C_q}(\alpha_{l\tau(k)}) \right\} = \max_{t=1}^{a_l} \left\{ \sum_{k=1}^{a_l} s^l(\alpha_{lt}, \alpha_{l\sigma_t(k)}) \cdot h_{C_q}(\alpha_{l\tau(k)}) \right\}$$

Dies soll an einem Beispiel veranschaulicht werden:

Sei A_l ein diskret numerisches Attribut mit den 4 Alternativen $\alpha_{l1}, \ldots, \alpha_{l4}$ und der Ähnlichkeitsmatrix

$$S = \begin{pmatrix} 1 & \frac{3}{5} & \frac{1}{2} & \frac{2}{5} \\ \frac{3}{5} & 1 & \frac{4}{5} & \frac{1}{10} \\ \frac{1}{2} & \frac{4}{5} & 1 & \frac{1}{5} \\ \frac{2}{5} & \frac{1}{10} & \frac{1}{5} & 1 \end{pmatrix}$$

A_l habe im Cluster C_q die Häufigkeiten $h_{C_q}(\alpha_{l1}) = 50$, $h_{C_q}(\alpha_{l2}) = 200$, $h_{C_q}(\alpha_{l3}) = 5$, $h_{C_q}(\alpha_{l4}) = 70$. Um die obere Schranke für die maximale Ähnlichkeit eines Datensatzes mit dem Cluster C_q (im l-ten Attribut) zu berechnen, sind folgende Schritte zu tun:

- Ordne die Häufigkeiten in absteigender Reihenfolge: $200 \geq 70 \geq 50 \geq 2$.
- Gehe zeilenweise durch S und berechne für jede Zeile (größte Ähnlichkeit · größte Häufigkeit) + (zweitgrößte Ähnlichkeit · zweitgrößte Häufigkeit) +...:
  $1 \cdot 200 + \frac{3}{5} \cdot 70 + \frac{1}{2} \cdot 50 + \frac{2}{5} \cdot 5 = 269$
  $1 \cdot 200 + \frac{4}{5} \cdot 70 + \frac{3}{5} \cdot 50 + \frac{1}{10} \cdot 5 = 286,5$
  $1 \cdot 200 + \frac{4}{5} \cdot 70 + \frac{1}{2} \cdot 50 + \frac{1}{5} \cdot 5 = 282$
  $1 \cdot 200 + \frac{2}{5} \cdot 70 + \frac{1}{3} \cdot 50 + \frac{1}{5} \cdot 5 = 259$
- Maximiere über die eben berechneten Werte:

$$\sum_{i \in C_q} s_{ij}^l \leq \max\{269,\ 286,5,\ 282,\ 259\} = 286,5$$

Als obere Schranke für (4.2) erhält man in diesem Beispiel also 286,5.

2. Eine bessere Abschätzung erhält man, wenn man die tatsächlichen Ähnlichkeiten zur Bestimmung der oberen Schranke verwendet. Hierzu lässt man x_{jl} alle Werte $\alpha_{l1}, \ldots, \alpha_{la_l}$ des Attributs A_l durchlaufen, berechnet für jeden Wert die Ähnlichkeit zu C_q und maximiert über diese Werte. Wie bei dem unter 2. beschriebenen Verfahren geht man dabei zeilenweise durch die Ähnlichkeitsmatrix und berechnet für jede Zeile t den Wert

$$\sum_{k=1}^{a_l} s_{tk}^l \cdot h_{C_q}(\alpha_{lk}),\ t = 1, \ldots, a_l$$

sodass man als obere Schranke für (4.2)

$$\sum_{i \in C_q} s_{ij}^l \leq \max_{t=1}^{a_l} \left\{ \sum_{k=1}^{a_l} s_{tk}^l \cdot h_{C_q}(\alpha_{lk}) \right\} = \max_{t=1}^{a_l} \left\{ \sum_{k=1}^{a_l} s^l(\alpha_{lt}, \alpha_{lk}) \cdot h_{C_q}(\alpha_{lk}) \right\}$$

erhält. Gegenüber dem obigen Verfahren hat man dadurch nicht nur eine Qualitätsverbesserung erzielt, sondern auch eine Laufzeitverbesserung, da die

Häufigkeiten und Ähnlichkeiten vorab nicht sortiert werden müssen. Zur Veranschaulichung sei das obige Beispiel noch einmal aufgegriffen. Bei dieser Methode geht man folgendermaßen vor:

- Berechne für jede der Alternativen $\alpha_{l1}, \ldots, \alpha_{l4}$ von A_l die Ähnlichkeit zum Cluster C_q:
  $s(\alpha_{l1}, C_q) = 1 \cdot 50 + \frac{3}{5} \cdot 200 + \frac{1}{2} \cdot 5 + \frac{2}{5} \cdot 70 = 200,5$
  $s(\alpha_{l2}, C_q) = \frac{3}{5} \cdot 50 + 1 \cdot 200 + \frac{4}{5} \cdot 5 + \frac{1}{10} \cdot 70 = 241$
  $s(\alpha_{l3}, C_q) = \frac{1}{2} \cdot 50 + \frac{4}{5} \cdot 200 + 1 \cdot 5 + \frac{1}{5} \cdot 70 = 204$
  $s(\alpha_{l4}, C_q) = \frac{2}{5} \cdot 50 + \frac{1}{10} \cdot 200 + \frac{1}{5} \cdot 5 + 1 \cdot 70 = 111$
- Maximiere über diese Ähnlichkeiten:

$$\sum_{i \in C_q} s_{ij}^l \leq \max\{200,5;\ 241,\ 204,\ 111\} = 241$$

Als obere Schranke für (4.2) erhält man mit dieser Methode also 241. Das Ergebnis ist besser als beim vorigen Verfahren bei gleichzeitiger Verringerung des Rechenaufwands.

Dieses Verfahren hat den weiteren Vorteil, dass man bei der Implementierung auf bereits implementierte Methoden zurückgreifen kann, da die Berechnung der Ähnlichkeit eines Datensatzes zu einem Cluster auch für die Berechnung des Optimalitätskriteriums benötigt wird.

4.2.4 Stetige Attribute

Sei A_l ein stetiges Attribut und X_j ein Datensatz, dessen Ähnlichkeit zum Cluster C_q berechnet werden soll (wobei $X_j \notin C_q$). Als Ähnlichkeitsmaß für diesen Attributtyp wurde wie bei den diskret numerischen Attributen das Maß (4.3) gewählt. Auch hier gibt es verschiedene Möglichkeiten zur Berechnung einer oberen Schranke:

1. Anstatt alle Datensätze im Cluster zu zählen, wie das bei der Abschätzung durch die Clustergröße der Fall ist, zählt man nur diejenigen Datensätze, die (im l-ten Attribut) zu X_j eine Ähnlichkeit haben, die größer ist als die Ähnlichkeitsschranke α ($0 < \alpha < 1$). Denn da s_{ij} ein aus den s_{ij}^l ($l = 1, \ldots, p$) aggregiertes Ähnlichkeitsmaß ist, gehen bei der Berechnung des Optimalitätskriteriums
$$z(C) = \sum_{\substack{i,j \text{ im gleichen} \\ \text{Cluster, } i \neq j}} (s_{ij} - \alpha)$$
alle Ähnlichkeiten, die kleiner sind als α, negativ ein und können bei der Abschätzung mit 0 abgeschätzt werden. Die Werte, die A_l annimmt, werden in Buckets unterteilt (siehe Kap. 3.2.2), sodass man alle Buckets berücksichtigen muss, in denen Datensätze liegen, die zu x_{jl} eine Ähnlichkeit größer α haben. Hierbei muss beachtet werden, dass der „Bucket-Bereich“ um x_{jl} groß genug gewählt wird, um auch den Fall abzudecken, dass x_{jl} am Rand seines Buckets liegt:

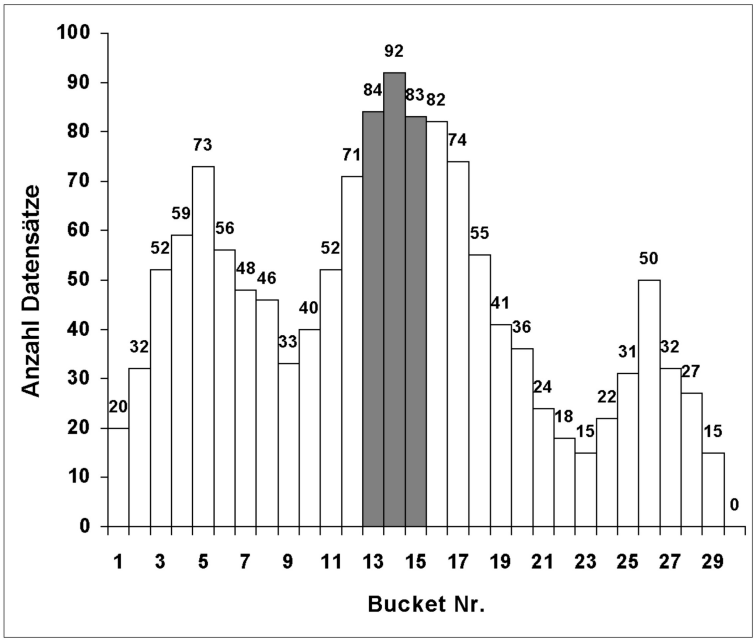


Abbildung 4.1: Beispiel für die Bestimmung des maximalen Bucket-Bereichs

Sei $d \in \mathbb{R}$ die Abstandseinheit, innerhalb der zwei Werte von A_l noch als ähnlich gelten (siehe Kapitel 3.1.2). Sei b_s das Bucket, das x_{jl} enthält, w die Bucket-Breite[2] und m die Anzahl der Buckets. Dann liegen links und rechts von b_s jeweils höchstens $\lceil \frac{d}{w} \rceil$ Buckets mit Werten, die zu x_{jl} eine Ähnlichkeit größer α haben. Insgesamt muss also ein Bucket-Bereich von $r := 2 \cdot \lceil \frac{d}{w} \rceil + 1$ vielen Buckets berücksichtigt werden. Nimmt man an, dass alle Datensätze in diesem Bereich maximale Ähnlichkeit zu x_{jl} haben, so ist für festes X_j

$$\sum_{k=s-\lceil \frac{d}{w} \rceil}^{s+\lceil \frac{d}{w} \rceil} h_{C_q}(b_k)$$

eine obere Schranke für (4.2). Da die obere Schranke von X_j unabhängig sein soll, muss unter den $(m-r)$–vielen Bucket-Bereichen jener bestimmt werden, für den dieser Wert am größten wird. Als obere Schranke für (4.2) ergibt sich somit

$$\sum_{i \in C_q} s_{ij}^l \leq \max_{k=0}^{m-r} \left\{ \sum_{t=k}^{k+r-1} h_{C_q}(b_t) \right\}$$

Wenn also ein stetiges Attribut A_l in einem Cluster C_q beispielsweise eine Häufigkeitsverteilung wie in Abbildung 4.1 hat, die Abstandseinheit 0,5 und die Bucketbreite 1 beträgt, so wird Folgendes berechnet:

[2]Beim *Intelligent Miner for Data* werden automatisch gleichgroße Buckets gebildet. Der Benutzer kann aber auch unterschiedlich große Buckets festlegen. Da dies in der Praxis äußerst selten vorkommt, wird hier nur der Fall gleichgroßer Buckets betrachtet.

- Bestimme die Größe des zu betrachtenden Bucket-Bereichs: $r = 2 \cdot \lceil \frac{1}{2} \rceil + 1 = 3$.
- Berechne für alle Bucket-Bereiche die Summe der Häufigkeiten in den zugehörigen Buckets:

Bucket-Bereich	1	2	3	4	5	6	7	8	9	10
Wert	104	143	184	188	177	150	127	119	125	163
Bucket-Bereich	11	12	13	14	15	16	17	18	19	20
Wert	207	247	259	257	239	211	170	132	101	78
Bucket-Bereich	21	22	23	24	25	26	27	28		
Wert	57	55	68	103	113	109	74	42		

- Maximiere über die eben berechneten Werte:

$$\max\{104, 143, 184, \ldots, 109, 74, 42\} = 259$$

Als obere Schranke für (4.2) erhält man somit den Wert 259.

2. Wenn man analog zu den diskret numerischen Attributen auch bei den stetigen Attributen eine gute Abschätzung erzielen will, indem man das Maximum der tatsächlichen Ähnlichkeiten verwendet, so stßt man auf das Problem, dass die Anzahl der auftretenden Alternativen oft genauso hoch ist wie die Anzahl der Datenstze. Dadurch wird die Berechnung der oberen Schranke zu zeitaufwndig. Man kann das Maximum

$$\max_x \left\{ \sum_{i \in C_q} s^l(x, x_{il}) \right\}$$

jedoch näherungsweise bestimmen. Gleich wie beim vorigen Ansatz wird der Bucket-Bereich $(b_k \cup \ldots \cup b_{k+r-1})$ bestimmt, der die meisten Datensätze enthält. Nun macht man zwei Annahmen:

- Das Maximum liegt in $(b_k \cup \ldots \cup b_{k+r-1})$.
  Damit ein Wert x eine sehr große Ähnlichkeit zu C_q erzielt, müssen möglichst viele in C_q auftretende Werte möglichst nahe bei x liegen und möglichst wenig Werte dürfen weit von x entfernt sein. Die Werte in $(b_k \cup \ldots \cup b_{k+r-1})$ haben genau diese Eigenschaft. Sind die Häufigkeiten derart verteilt, dass die Dichtefunktion monoton oder zweiseitig monoton mit einem Extremwert (wie beispielsweise bei der Normalverteilung) verläuft, so liegt der Wert mit der maximalen Ähnlichkeit sicher in diesem Bereich. Sind die Häufigkeiten ungünstig verteilt, kann es passieren, dass er außerhalb dieses Bereichs liegt. Doch auch in diesem Fall liegen in $(b_k \cup \ldots \cup b_{k+r-1})$ Werte mit einer sehr hohen Ähnlichkeit zu C_q, sodass das dort gefundene Maximum nahe beim tatsächlichen liegt. Abbildung (4.2) zeigt den Fall einer ungünstigen Verteilung: wenn die Abstandseinheit $d = 1$ ist, so beträgt bei einer Bucket-Breite von 1 der zu betrachtende Bucket-Bereich $r = 3$ Buckets. Zur Vereinfachung der Rechnung sei die

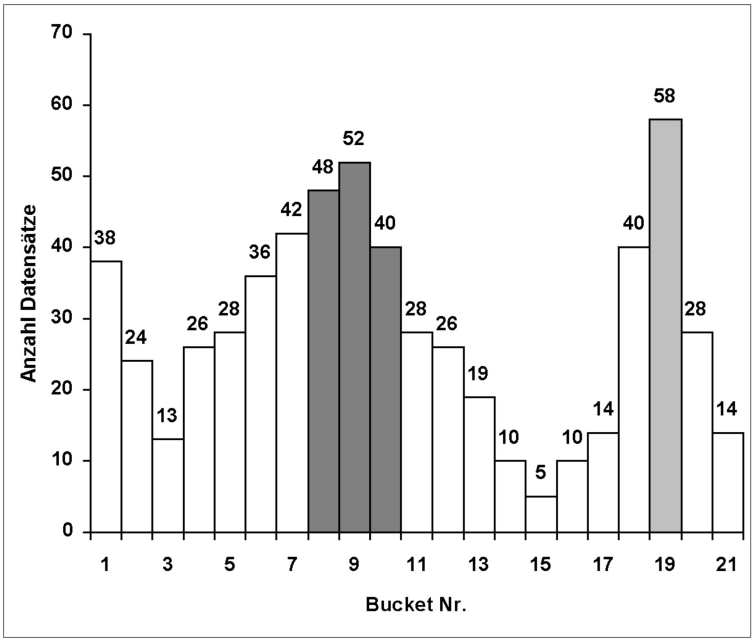


Abbildung 4.2: Ungünstige Verteilung der Häufigkeiten

Ähnlichkeit zweier Werte x, y aus dem Wertebereich des Attributs nicht wie in (3.3) definiert, sondern folgendermaßen:

$$|x-y| \leq \tfrac{d}{2} \Rightarrow s(x,y) = 1$$
$$\tfrac{d}{2} < |x-y| \leq d \Rightarrow s(x,y) = \tfrac{1}{2}$$
$$d < |x-y| \Rightarrow s(x,y) = 0$$

Das Maximum wird in den Buckets 8–10 gesucht, da diese zusammen die größte Häufigkeit haben. Geht man aber von einer Gleichverteilung der Werte innerhalb eines Buckets aus (siehe nächste Annahme), so liegt das tatsächliche Maximum in Bucket Nr.19, da ein Wert in der Mitte dieses Buckets eine Ähnlichkeit von $1 \cdot 58 + \frac{1}{2}(\frac{1}{2} \cdot 40 + \frac{1}{2} \cdot 28) = 75$ zum Cluster hat, wohingegen ein Wert in Bucket Nr.9 höchstens eine Ähnlichkeit von $1 \cdot 52 + \frac{1}{2}(\frac{1}{2} \cdot 48 + \frac{1}{2} \cdot 40) = 74$ erzielt.

- Die Werte innerhalb eines Buckets b_k sind gleichverteilt. Eigentlich braucht diese Annahme an dieser Stelle nicht explizit gemacht werden, da sie bereits benötigt wurde, um die Dichtefunktion eines stetigen Attributs mit der Histogramm-Funktion approximieren zu können, und somit implizit bereits gilt. Da sie aber für die folgende Abschätzung benötigt wird, sei sie hier noch einmal erwähnt.

  Sind die Werte innerhalb eines Buckets gleichverteilt, so liegt der Wert, der die größte Ähnlichkeit zum Cluster C_q hat, in der Mitte oder am Rand des Buckets. Denn hat eines der Nachbar-Buckets eine größere Häufigkeit, so liegen in der Nähe des entsprechenden Randwertes von b_k mehr Werte als in der Umgebung eines anderen Wertes von b_k (außer das andere Nachbar-

Bucket hat eine noch größere Häufigkeit, aber in diesem Fall betrachtet man den anderen Randwert von b_k). Haben beide Nachbar-Buckets eine kleinere Häufigkeit als b_k, so ist (bei Gleichverteilung der Werte innerhalb der Buckets) ein Wert mit maximaler Ähnlichkeit zu C_q in der Bucket-Mitte zu finden: alle anderen Werte haben die gleiche oder eine kleinere Ähnlichkeit zu C_q, da in ihrer Umgebung gleich viele oder weniger Werte liegen. Sind die Werte innerhalb der Buckets sehr ungünstig verteilt, etwa derart, dass in der Nähe der Ränder und der Mitte sehr wenige Werte liegen, so wird das Ähnlichkeitsmaximum von einem anderen Wert im Bucket erreicht.

Gelten diese Annahmen, so kann man die maximale tatsächliche Ähnlichkeit eines Wertes aus dem Wertebereich von A_l zu dem Cluster C_q dadurch bestimmen, dass man für alle Rand- und Mittelwerte der Buckets aus dem Bucket-Bereich $(b_k \cup \ldots \cup b_{k+r-1})$ die tatsächliche Ähnlichkeit zu C_q berechnet und anschließend das Maximum dieser Werte bildet. Als obere Schranke für (4.2) erhält man somit:

$$\sum_{i \in C_q} s_{ij}^l \leq \max_{t=k}^{k+r-1} \left\{ \max \left\{ \sum_{i \in C_q} s^l(\bar{b}_{t-1}, x_{il}), \sum_{i \in C_q} s^l(\bar{b}_{t-1} + \frac{w}{2}, x_{il}) \right\} \right\}$$

wobei

$$k \in \mathbb{N} \text{ mit } \sum_{p=k}^{k+r-1} h_{C_q}(b_p) = \max_{s=0}^{m-r} \left\{ \sum_{p=s}^{s+r-1} h_{C_q}(b_p) \right\}$$
$$b_i = [\bar{b}_{i-1}, \bar{b}_i) \text{ für } i \in \{1, \ldots, m\},\ w = \text{Bucketbreite}$$

3. Man kann in der vorigen Abschätzung auf die erste Annahme verzichten, wenn man sich nicht auf den maximalen Bucket-Bereich beschränkt, sondern bei allen Buckets die Ähnlichkeit der Rand- und Mittelwerte zu dem Cluster C_q berechnet und über diese Ähnlichkeiten maximiert. Man muß also einmal über alle Buckets gehen, was aber gegenüber der vorigen Methode keinen Mehraufwand bedeutet, da man dies für die Bestimmung des „maximalen Bucket-Bereichs“ ebenfalls tun muß. Die Annahme über die Gleichverteilung der Werte innerhalb der Buckets benötigt man weiterhin. Als Abschätzung erhält man somit:

$$\sum_{i \in C_q} s_{ij}^l \leq \max_{t=1}^{m} \left\{ \max \left\{ \sum_{i \in C_q} s^l(\bar{b}_{t-1}, x_{il}), \sum_{i \in C_q} s^l(\bar{b}_{t-1} + \frac{w}{2}, x_{il}) \right\} \right\}$$

mit

$$b_i = [\bar{b}_{i-1}, \bar{b}_i) \text{ für } i \in \{1, \ldots, m\},\ w = \text{Bucketbreite}$$

4.3 Korrektheit

4.3.1 Alle Attributtypen

Abschätzung mit der Clustergröße als obere Schranke:
Sei $C_q,\ q \in \{1,\ldots,k\}$, ein Cluster der aktuellen Clusterung, $X_j \notin C_q$ ein beliebiger Datensatz und $A_l,\ l \in \{1,\ldots,p\}$, ein beliebiges Attribut. Dann ist

$$\sum_{i \in C_q} s_{ij}^l \leq \sum_{i \in C_q} 1 = n_q.$$

4.3.2 Kategorische Attribute

Sei $C_q,\ q \in \{1,\ldots,k\}$, ein Cluster der aktuellen Clusterung, $X_j \notin C_q$ ein beliebiger Datensatz und $A_l,\ l \in \{1,\ldots,p\}$, ein kategorisches Attribut mit den Alternativen $\alpha_{l1},\ldots,\alpha_{la_l}$. Sei Œ $x_{jl} = \alpha_{l1}$.
Abschätzung mit $\max_{k=1}^{a_l}\{h_{C_q}(\alpha_{lk})\}$:

$$\sum_{i \in C_q} s_{ij}^l = \sum_{i \in C_q} s^l(x_{il}, x_{jl}) = \sum_{i \in C_q} s^l(x_{il}, \alpha_{l1}) = h_{C_q}(\alpha_{l1}) \leq \max_{k=1}^{a_l}\{h_{C_q}(\alpha_{lk})\}.$$

4.3.3 Diskret numerische Attribute

Sei $C_q,\ q \in \{1,\ldots,k\}$, ein Cluster der aktuellen Clusterung, $X_j \notin C_q$ ein beliebiger Datensatz und $A_l,\ l \in \{1,\ldots,p\}$, ein diskret numerisches Attribut mit den Alternativen $\alpha_{l1},\ldots,\alpha_{la_l}$.

- Abschätzung mit $\max_{t=1}^{a_l}\left\{\sum_{k=1}^{a_l} s^l(\alpha_{lt}, \alpha_{lk}) \cdot h_{C_q}(\alpha_{lk})\right\}$:

$$\sum_{i \in C_q} s_{ij}^l = \sum_{k=1}^{a_l} s^l(x_{jl}, \alpha_{lk}) \cdot h_{C_q}(\alpha_{lk}) \leq \max_{t=1}^{a_l} \left\{ \sum_{k=1}^{a_l} s^l(\alpha_{lt}, \alpha_{lk}) \cdot h_{C_q}(\alpha_{lk}) \right\} \quad (4.4)$$

- Abschätzung mit $\max_{t=1}^{a_l}\left\{\sum_{k=1}^{a_l} s^l(\alpha_{lt}, \alpha_{l\sigma_t(k)}) \cdot h_{C_q}(\alpha_{l\tau(k)})\right\}$:
  Sei

  $$\tau : \{1,\ldots,a_l\} \to \{1,\ldots,a_l\} \text{ eine Permutation mit}$$
  $$h_{C_q}(\alpha_{l\tau(1)}) \geq h_{C_q}(\alpha_{l\tau(2)}) \geq \cdots \geq h_{C_q}(\alpha_{l\tau(a_l)})$$

  und

  $$\sigma_t : \{1,\ldots,a_l\} \to \{1,\ldots,a_l\} \text{ eine Permutation mit}$$
  $$1 = s_{t\sigma_t(1)} \geq s_{t\sigma_t(2)} \geq \cdots \geq s_{t\sigma_t(a_l)}.$$

  Dann lässt sich (4.4) fortsetzen:

$$\max_{t=1}^{a_l} \left\{ \sum_{k=1}^{a_l} s^l(\alpha_{lt}, \alpha_{lk}) \cdot h_{C_q}(\alpha_{lk}) \right\} \leq \max_{t=1}^{a_l} \left\{ \sum_{k=1}^{a_l} s^l(\alpha_{lt}, \alpha_{l\sigma_t(k)}) \cdot h_{C_q}(\alpha_{l\tau(k)}) \right\}$$

4.3.4 Stetige Attribute

Sei $C_q,\ q \in \{1,\ldots,k\}$, ein Cluster der aktuellen Clusterung, $X_j \notin C_q$ ein beliebiger Datensatz und $A_l,\ l \in \{1,\ldots,p\}$, ein stetiges Attribut. Sei w die Bucketbreite, m die Anzahl der Buckets $b_i = [\bar{b}_{i-1}, \bar{b}_i),\ i \in \{1,\ldots,m\}$, und b_s das Bucket, in dem x_{jl} enthalten ist.

- Abschätzung durch $\max_{k=0}^{m-r}\left\{\sum_{t=k}^{k+r-1} h_{C_q}(b_t)\right\}$

  Sei d die Abstandseinheit für A_l und $r = 2 \cdot \lceil \frac{d}{w} \rceil + 1$ die Größe des zu betrachtenden Bucket-Bereichs. Wie in Kapitel 4.2.4 bereits bemerkt wurde, können die Ähnlichkeiten von X_j zu Datensätzen, die weiter als d von x_{jl} entfernt sind, mit 0 abgeschätzt werden. Man erhält somit

$$\begin{aligned}
\sum_{i \in C_q} s_{ij}^l &= \frac{1}{w} \sum_{i=1}^{m} h_{C_q}(b_i) \cdot \int_{\bar{b}_{i-1}}^{\bar{b}_i} \exp\left(-\ln 2 \left(\frac{|x_{jl}-t|}{\delta_l}\right)^2\right) dt \\
&\leq \frac{1}{w} \sum_{i=s-\lceil \frac{d}{w} \rceil}^{s+\lceil \frac{d}{w} \rceil} h_{C_q}(b_i) \cdot \int_{\bar{b}_{i-1}}^{\bar{b}_i} \exp\left(-\ln 2 \left(\frac{|x_{jl}-t|}{\delta_l}\right)^2\right) dt \\
&\leq \frac{1}{w} \sum_{i=s-\lceil \frac{d}{w} \rceil}^{s+\lceil \frac{d}{w} \rceil} h_{C_q}(b_i) \cdot (w \cdot 1) \\
&= \sum_{i=s-\lceil \frac{d}{w} \rceil}^{s+\lceil \frac{d}{w} \rceil} h_{C_q}(b_i) \leq \max_{k=0}^{m-r} \left\{ \sum_{t=k}^{k+r-1} h_{C_q}(b_t) \right\}
\end{aligned}$$

- Die Abschätzung, bei der das Maximum der tatsächlichen Ähnlichkeiten der Bucket-Ränder und -Mitten des maximalen Bucket-Bereichs als obere Schranke verwendet werden (Vorschlag Nr. 2), soll an dieser Stelle nicht weiter betrachtet werden. Grund hierfür ist die Tatsache, dass sie für die Praxis uninteressant ist, da der nachfolgende Vorschlag (Nr. 3) bei gleichem Rechenaufwand (man muss einmal über alle Buckets gehen) ein eventuell besseres Ergebnis liefert, da keine Buckets ausgeschlossen werden. Des weiteren birgt der dritte Vorschlag weniger potentielle Quellen für Implementationsfehler.

- Abschätzung mit $\max_{t=1}^{m}\left\{\max\left\{\sum_{i \in C_q} s^l(\bar{b}_{t-1}, x_{il}), \sum_{i \in C_q} s^l(\bar{b}_{t-1} + \frac{w}{2}, x_{il})\right\}\right\}$

  Bei dieser Abschätzung kann ein Fehler auftreten, d.h. der berechnete Wert ist eventuell kleiner als das tatsächliche Ähnlichkeitsmaximum. Man kann jedoch zeigen, dass die Größe des Fehlers mit kleiner werdender Bucketbreite gegen null geht.

  Sei x ein Wert aus dem Wertebereich von A_l mit maximaler Ähnlichkeit zu C_q und $y \in \{\bar{b}_0, \bar{b}_0 + \frac{w}{2}, \bar{b}_1, \ldots, \bar{b}_m\}$ der vom Algorithmus gefundene Wert mit maximaler Ähnlichkeit zu C_q. Seien Œ x, y im gleichen Bucket b_s (Sonst sei $\tilde{y} \in \{\bar{b}_0, \bar{b}_0 + \frac{w}{2}, \bar{b}_1, \ldots, \bar{b}_m\}$ mit $s^l(\tilde{y}, C_q) \geq s^l(y, C_q)$. Ein Fehler tritt nur auf,

wenn $s^l(\tilde{y}, C_q) < s^l(x, C_q)$. Für den Fehler ϵ gilt dann $\epsilon = s^l(x, C_q) - s^l(\tilde{y}, C_q) \leq s^l(x, C_q) - s^l(y, C_q))$.

Setzt man die Definition (3.3) des für stetige Attribute verwendeten Ähnlichkeitsmaßes ein, so ergibt sich für den Fehler

$$\epsilon \leq \frac{1}{w} \sum_{i=1}^{m} h_{C_q}(b_i) \cdot \underbrace{\int_{\bar{b}_{i-1}}^{\bar{b}_i} \exp\left(-\ln 2 \left(\frac{|x-t|}{\delta_l}\right)^2\right) dt}_{=:A}$$
$$- \frac{1}{w} \sum_{i=1}^{m} h_{C_q}(b_i) \cdot \underbrace{\int_{\bar{b}_{i-1}}^{\bar{b}_i} \exp\left(-\ln 2 \left(\frac{|y-t|}{\delta_l}\right)^2\right) dt}_{=:B}$$

Die beiden Exponentialfunktionen nehmen ihr Maximum bei x bzw. y an, sind links davon monoton wachsend und rechts davon monoton fallend. Somit kann man die Terme A und B folgendermaßen abschätzen:

– für $1 \leq i < s$:

$$A \leq w \cdot \exp\left(-\frac{\ln 2}{\delta_l^2}(x - \bar{b}_i)^2\right)$$
$$B \geq w \cdot \exp\left(-\frac{\ln 2}{\delta_l^2}(y - \bar{b}_{i-1})^2\right)$$

– für $i = s$:

$$A \leq w \cdot \exp\left(-\frac{\ln 2}{\delta_l^2}(x - x)^2\right) = w$$
$$B \geq w \cdot \exp\left(-\frac{\ln 2}{\delta_l^2}(y - \bar{b}_s)^2\right)$$

– für $s < i \leq m$:

$$A \leq w \cdot \exp\left(-\frac{\ln 2}{\delta_l^2}(x - \bar{b}_{i-1})^2\right)$$
$$B \geq w \cdot \exp\left(-\frac{\ln 2}{\delta_l^2}(y - \bar{b}_i)^2\right)$$

Für die Fehlerabschätzung ergibt sich damit

$$\begin{aligned}
\epsilon \leq\ & \frac{1}{w}\sum_{i=1}^{s-1} h_{C_q}(b_i)\left[w\cdot\exp\left(-\frac{\ln 2}{\delta_l^2}(x-\bar{b}_i)^2\right) - w\cdot\exp\left(-\frac{\ln 2}{\delta_l^2}(y-\bar{b}_{i-1})^2\right)\right] \\
& + \frac{h_{C_q}(b_i)}{w}\left[w - w\cdot\exp\left(-\frac{\ln 2}{\delta_l^2}(y-\bar{b}_s)^2\right)\right] \\
& + \frac{1}{w}\sum_{i=s+1}^{m} h_{C_q}(b_i)\left[w\cdot\exp\left(-\frac{\ln 2}{\delta_l^2}(x-\bar{b}_{i-1})^2\right) - w\cdot\exp\left(-\frac{\ln 2}{\delta_l^2}(y-\bar{b}_i)^2\right)\right] \\
\\
\leq\ & \sum_{i=1}^{s-1} h_{C_q}(b_i)\left[\exp\left(-\frac{\ln 2}{\delta_l^2}((s-i-1)w)^2\right) - \exp\left(-\frac{\ln 2}{\delta_l^2}((s-i+1)w)^2\right)\right] \\
& + h_{C_q}(b_i)\left[1-\exp\left(-\frac{\ln 2}{\delta_l^2}w^2\right)\right] \\
& + \sum_{i=s+1}^{m} h_{C_q}(b_i)\left[\exp\left(-\frac{\ln 2}{\delta_l^2}((i-s-1)w)^2\right) - \exp\left(-\frac{\ln 2}{\delta_l^2}((i-s+1)w)^2\right)\right] \\
\\
\leq\ & \sum_{i=1}^{s-1} h_{C_q}(b_i)\left[\left(\exp\left(-\ln 2\left(\frac{s-i-1}{\delta_l}\right)^2\right)\right)^{w^2} - \left(\exp\left(-\ln 2\left(\frac{s-i+1}{\delta_l}\right)^2\right)\right)^{w^2}\right] \\
& + h_{C_q}(b_i)\left[1-\left(\exp\left(-\frac{\ln 2}{\delta_l^2}\right)\right)^{w^2}\right] \\
& + \sum_{i=s+1}^{m} h_{C_q}(b_i)\left[\left(\exp\left(-\ln 2\left(\frac{i-s-1}{\delta_l}\right)^2\right)\right)^{w^2} - \left(\exp\left(-\ln 2\left(\frac{i-s+1}{\delta_l}\right)^2\right)\right)^{w^2}\right]
\end{aligned}$$

Für $w \to 0$ geht der Ausdruck gegen 0. Über die Bucketbreite kann also die Fehlergröße reguliert werden.

4.4 Zur Implementierung

Teil dieser Diplomarbeit ist die Implementierung der in den vorigen Abschnitten vorgestellten Verbesserungsvorschläge für den Demographic Clustering Algorithmus. Hierbei wurde von den verschiedenen Möglichkeiten der Bestimmung einer oberen Schranke jeweils folgende realisiert:

- Für kategorische Attribute

  Hier wurde Vorschlag Nr.2 aus Kapitel 4.2.2 gewählt, also die Abschätzung durch die maximale im Cluster auftretende Häufigkeit eines Attributwertes.

- Für diskret numerische Attribute

  Möchte man eine bessere obere Schranke als die Clustergröße, so ist klar, dass von den Vorschlägen für diskret numerische Attribute der dritte implementiert

werden sollte, also die Abschätzung mittels des Maximums der (tatsächlichen) Ähnlichkeiten der einzelnen Attributwerte zum betrachteten Cluster. Zum einen erhält man gegenüber dem zweiten Vorschlag eine bessere Abschätzung, zum anderen spart man Rechenaufwand, da die Häufigkeiten der Alternativen sowie die Zeilen der Ähnlichkeitsmatrix nicht sortiert werden müssen.

- Für stetige Attribute

  Bei den stetigen Attributen wurde die vierte Möglichkeit implementiert, also die Bestimmung der maximalen Ähnlichkeit aller Bucket-Ränder und -Mitten zum Cluster als obere Schranke.

Die Implementierung wurde für das *IBM DB2 Intelligent Miner Scoring* - Tool realisiert, welches einen Teil der Mining-Algorithmen des *Intelligent Miner for Data* für die direkte Anwendung auf bereits bestehende Mining-Modelle aus einer DB2-Datenbank heraus zur Verfügung stellt. Für das demografische Clustern heißt dies konkret: Mit Hilfe des *IM Scoring* kann man Daten in der Datenbank gegen ein bereits erstelltes Cluster-Modell „scoren“, was bedeutet, dass die Daten unter Verwendung des Demographic Clustering Algorithmus den Clustern zugeordnet werden. Hierbei werden jedoch nicht die in Kapitel 3.1.2 vorgestellten Maße verwendet, sondern davon abgeleitete *Score-Maße*, was im Folgenden genauer erläutert werden soll:

4.4.1 Das Score-Maß

Das Score-Maß ist ein auf einem Ähnlichkeitsmaß basierendes Maß, das man erhält, indem man den Wertebereich des zugrunde liegenden Ähnlichkeitsmaßes derart auf ein Intervall $I \subseteq [-1, 1]$ abbildet, dass die Ähnlichkeitsschranke auf null abgebildet wird.

Definition 4.4.1 (Inneres und äußeres Score-Maß) *Sei s_{ij}^l ein Ähnlichkeitsmaß gemäß Definition (2.3.1) und $\alpha \in [0, 1]$ eine Ähnlichkeitsschranke. Dann heißt die Transformation*

$$score_l(x_{il}, x_{jl}) := (s_{ij}^l - \alpha) \in [-\alpha, 1 - \alpha]$$

inneres Score-Maß *oder kurz* innerer Score *des l-ten Merkmals.*
Aggregiert man analog zu Ähnlichkeitsmaßen die gewichteten inneren Score-Maße, so erhält man das äußere Score-Maß *(bzw. den* äußeren Score*)*

$$score(X_i, X_j) := \sum_{k=1}^{p} w_l \cdot score_l(x_{il}, x_{jl}) \in [-(w_1 + \ldots + w_l)\alpha, (w_1 + \ldots + w_l)(1 - \alpha)].$$

Beim *Intelligent Miner for Data* (sowie beim *IM Scoring*) werden die auf den in Kapitel 3.1.2 eingeführten Ähnlichkeitsmaßen basierenden inneren Score-Maße und das daraus aggregierte gewichtete äußere Score-Maß verwendet.
Der Vorteil bei der Verwendung von Scores liegt darin, dass sich die Zuordnung eines Datensatzes zu einem Cluster anschaulich interpretieren lässt. Denn betrachtet man das dem Optimalitätskriterium zugrunde liegende Condorcet-Abstimmungsverhalten

(siehe Kap. 3.1.1), so lässt sich ein positiver äußerer Score als Zustimmung und ein negativer äußerer Score als Ablehnung der gegebenen Partition interpretieren (bisher war die Ähnlichkeitsschranke α die „Grenze“ zwischen Zustimmung und Ablehnung). Möchte man also einen Datensatz in ein Cluster einfügen und der Score wird beispielsweise für jedes Cluster negativ, so bedeutet dies, dass der Datensatz in keines der Cluster eingefügt werden soll. In diesem Fall wird mit dem Datensatz ein neues Cluster gegründet (außer die benutzerdefinierte maximale Clusterzahl ist bereits erreicht, dann wird der Datensatz dem Cluster mit dem höchsten Score zugeordnet). Das Optimalitätskriterium

$$z(C) = \sum_{\substack{i,j \text{ im gleichen} \\ \text{Cluster, } i \neq j}} (s_{ij} - \alpha),$$

muss für die Verwendung von Scores nur minimal modifiziert werden. Multipliziert man den Term mit der Konstanten $(w_1 + \ldots + w_l)$, so ändert sich an der Lage des Maximums nicht und man erhält:

$$\begin{aligned}
\tilde{z}(C) &= (w_1 + \ldots + w_l) \sum_{\substack{i,j \text{ im gleichen} \\ \text{Cluster, } i \neq j}} (s_{ij} - \alpha) \\
&\overset{(3.2)}{=} (w_1 + \ldots + w_l) \sum_{\substack{i,j \text{ im gleichen} \\ \text{Cluster, } i \neq j}} \left(\frac{1}{\sum_{l=1}^{p} w_l} \sum_{l=1}^{p} w_l \cdot s_{ij}^l - \alpha \right) \\
&= \sum_{\substack{i,j \text{ im gleichen} \\ \text{Cluster, } i \neq j}} \left(\left(\sum_{l=1}^{p} w_l \cdot s_{ij}^l \right) - (w_1 + \ldots + w_l)\alpha \right) \\
&= \sum_{\substack{i,j \text{ im gleichen} \\ \text{Cluster, } i \neq j}} \sum_{l=1}^{p} w_l \cdot (s_{ij}^l - \alpha) = \boxed{\sum_{\substack{i,j \text{ im gleichen} \\ \text{Cluster, } i \neq j}} score(X_i, X_j)}
\end{aligned}$$

Dies ist das Optimalitätskriterium, wie es bei der Implementierung des Demographic Clustering Algorithmus realisiert ist. Wie sich leicht nachrechnen lässt, ändern sich die Update-Formeln folgendermaßen:

- beim Einfügen eines Datensatzes X_j in ein Cluster C_q

$$\Delta\tilde{z} = 2 \cdot \sum_{i \in C_q} score(X_i, X_j)$$

- beim Entfernen eines Datensatzes X_j aus einem Cluster C_q

$$\Delta\tilde{z} = -2 \cdot \sum_{i \in C_q} score(X_i, X_j)$$

Wie auch beim Ähnlichkeitsmaß (3.2) ist auch bei der Verwendung des Score-Maßes die Berechnung der Summen $\sum_{i \in C_q} score(X_i, X_j),\ q = 1, \cdots, k$, notwendig. Wegen

$$\sum_{i \in C_q} score_l(x_{il}, x_{jl}) = \sum_{i \in C_q} \left(s_{ij}^l - \alpha \right) = \left(\sum_{i \in C_q} s_{ij}^l \right) - n_q \cdot \alpha$$

lassen sich jedoch die Überlegungen zur effizienten Berechnung dieser Werte mittels der im vorigen Kapitel eingeführten Datenstruktur auf das Score-Maß übertragen.

4.4.2 Implementierung

Im Folgenden wird gezeigt, an welchen Programmteilen welche Änderungen vorgenommen wurden, um die oben erläuterte Verbesserungsidee umzusetzen. Dabei werden die neuen bzw. die veränderten Teile in Pseudo-Code angeführt. Auslassungspunkte kennzeichnen für das Verständnis nicht notwendige Programmteile.
Wichtig bei der Implementierung war, dass die wesentlichen Kontrollstrukturen beibehalten wurden. Die nötigen Berechnungen werden alle vorab bei der Initialisierung gemacht, sodass sich der eigentliche Programmablauf nur wenig ändert. Insbesondere an der „kritischen Stelle“, nämlich in der Schleife über alle Datensätze, sind somit keine weiteren Berechnungen nötig. Dies ist entscheidend, da diese Schleife sehr oft durchlaufen wird und darum zusätzliche Berechnungen in dieser Schleife zu einem enormen Mehraufwand führen würden, was das Programm trotz der gesparten Datensatz-Cluster-Vergleiche letztenendes verlangsamen würde. Die zur Bestimmung der oberen Schranke benötigten Berechnungen müssen nicht für jeden Datensatz, sondern nur für jedes Cluster durchgeführt werden, sodass der dafür nötige Aufwand vernachlässigbar gering ist, da Anzahl Cluster $\ll$ Anzahl Datensätze.

Initialisierung

Bevor Daten gegen ein Cluster-Modell gescort werden können, muss das Modell aus der Datenbank geladen werden.[3] Das Modell enthält alle für das Scoren notwendigen Informationen, wie zum Beispiel die Cluster-Statistiken, die zum Clustern verwendeten Attribute, die Gewichte oder die Ähnlichkeitsschranke. Mit diesen Daten wird zu Beginn des Scorings der Konstruktor für die Cluster aufgerufen. Wie in Kapitel 3.2.2 erläutert, werden die Cluster nicht mengentheoretisch gespeichert, sondern als Statistiken über die Häufigkeiten der einzelnen Attribut-Werte bzw. der Werte in den Buckets. Bei der Initialisierung der Statistiken können in der Schleife über alle Attribute die oberen Schranken für alle Attribute problemlos mitberechnet werden:

```
/* CLUSTER CONSTRUCTOR
...
maxScoreBound = 0

for all categorical fields F do
    intialize the cluster statistics for F
    ...
    get the similarity threshold sim
    get the number n of records in the cluster
    determine the maximum frequency MaxFreq of the field values of F
    maxScoreBound += MaxFreq -(sim * n)
```

[3] Jedes Modell, das im XML-Format gespeichert wurde, kann mittels eines einfachen Skripts in der DB2 Datenbank abgelegt werden.

```
end for

for all discrete numeric fields F do
    intialize the cluster statistics for F
    ...
    for all field values v of F do
        score v against this cluster
        store the current best score curBest
    end for

    maxScoreBound += curBest
end for

for all continuous fields F do
    intialize the cluster statistics for F
    ...
    score the values of the bucket limits and the bucket middles
        against this cluster
    determine the maximum maxScore of these scores

    maxScoreBound += maxScore
end for

maxScoreBound = 2 * maxScoreBound
...
```

Sortieren

Damit beim Scoren der Daten die Schleife über alle Cluster früher abgebrochen werden kann, müssen die Cluster anhand der oberen Schranke in absteigender Reihenfolge sortiert werden. Als Sortieralgorithmus wurde hierbei QuickSort gewählt.

Scoring

Beim Scoren der Daten werden in einer äußeren Schleife alle zu scorenden Datensätze abgearbeitet. In einer inneren Schleife werden die Cluster abgearbeitet. Bisher war dies eine Schleife über alle Cluster:

```
...
for all records X do
    ...
    for all clusters C do
        calculate score(X,C)
        ...
    end for
end for
...
```

Diese wurde ersetzt durch eine Schleife mit Abbruchbedingung: die Schleife wird abgebrochen, wenn der Score für das aktuell betrachtete Cluster größer ist als die obere Schranke des nächsten Clusters in der sortierten Clusterfolge:

```
...
for all records X do
    ...
    bestCluster = FALSE
    lastCluster = FALSE
    while (!bestCluster && !lastCluster)
        get the next cluster C in the sorted cluster array
        calculate score(X,C)
        ...
        if C is not the last cluster
            get the bound B for the score for the next cluster
                in the sorted cluster array
            if (score(X,C)>B)
                bestCluster = TRUE
        else
            lastCluster = TRUE
    end while
end for
...
```

An der Struktur des eigentlichen Programms wurde also nur wenig geändert. Wie eingangs schon erläutert, ist vor allem die Tatsache, dass pro Datensatz keine zusätzlichen Berechnungen gemacht werden müssen, entscheidend für eine tatsächliche Laufzeit-Ersparnis.

4.4.3 Gewichte

Die bisherigen Veränderungen berücksichtigen nicht den Fall gewichteter Attribute und Attributwerte. Diese Fälle lassen sich durch kleine zusätzliche Veränderungen abdecken:

Attribut-Gewichte

Da manche Attribute wichtiger sind als andere, ist es sinnvoll, dies bei der Erstellung eines Cluster-Modells zu berücksichtigen. Beim *IM for Data* hat der Benutzer die Möglichkeit, den einzelnen Attributen unterschiedliche Gewichte zu geben. Bei der Berechnung der oberen Schranke für den Score muss dies natürlich berücksichtigt werden, da ein Gewicht größer eins einen positiven Score noch vergrößert. Für diskret numerische und stetige Attribute braucht man hierfür nichts weiter zu tun, da die Berechnung der oberen Schranke mit Hilfe der tatsächlichen Scores erfolgt und in der Score-Methode, die hierzu aufgerufen wird, der Fall gewichteter Attribute berücksichtigt wird. Bei kategorischen Attributen wird man den Attribut-Gewichten durch eine Fallunterscheidung im Cluster-Konstruktor gerecht:

```
...
for all categorical fields F do
    ...
    get the highest frequency of the field values of F
    ...
    if (weighting)
        calculate MaxScoreBound with respect to the field weight
    else
        calculate MaxScoreBound with field weight = 1
end for
...
```

Da der Sonderfall ungewichteter Attribute sehr häufig auftritt, ist eine Fallunterscheidung sinnvoll, denn dann spart man sich in diesem Fall das Einlesen der Gewichte.

Wertgewichte

Analog zu den Attributgewichten hat der Benutzer die Möglichkeit, den einzelnen Werten eines Attributs unterschiedliche Gewichte zu geben. Mit der gleichen Argumentation wie oben und da in der Score-Methode Wertgewichte berücksichtigt werden, brauchen lediglich für kategorische Attribute Veränderungen vorgenommen werden. Auch hier ist eine Fallunterscheidung sinnvoll, da der Fall gewichteter Werte selten vorkommt. In diesem Fall wird nicht die maximale Häufigkeit zur Abschätzung verwendet, sondern es wird - wie bei den diskret numerischen Attributen - für jeden Attributwert der tatsächliche Score berechnet und über diese Werte maximiert.

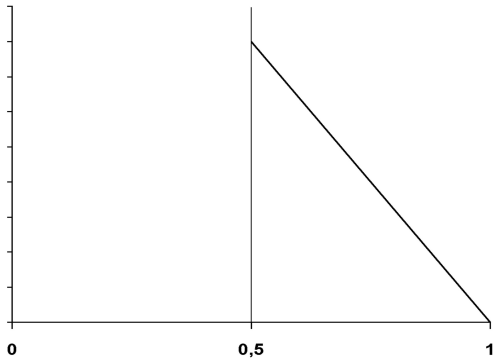
4.5 Laufzeit-Ersparnis

Wie am Anfang dieses Kapitels bereits erläutert, ist es nicht möglich, eine Laufzeit in einer besseren Komplexitätsklasse als $\mathcal{O}(n)$ zu erhalten. Dies bedeutet, dass man eine Laufzeitverbesserung nicht garantieren kann; im worst case müssen ebensoviele Datensatz-Cluster-Vergleiche gemacht werden wie beim ursprünglichen Algorithmus. Der worst case tritt ein, wenn die obere Schranke für den maximalen Score bei allen Clustern ungefähr gleich hoch ist. Da diese obere Schranke stark von der Clustergröße abhängt, kann man vereinfacht auch sagen, dass im worst case alle Cluster gleich groß sind. In der Praxis tritt dieser Fall äußerst selten auf; hier hat man meistens den Fall weniger großer und vieler kleiner Cluster.

Im Folgenden wird zunächst in einer theoretischen Abschätzung gezeigt, wieviele Vergleiche man unter gewissen (pessimistisch gewählten) Voraussetzungen sparen kann. Anschließend wird für verschiedene Testdaten gezeigt, wie groß die Ersparnis im konkreten Fall ist. Die Verteilungen der Werte in den Testdaten basieren auf Verteilungen von in der Praxis auftretenden Daten (soweit Informationen darüber vorliegen), sodass die Ergebnisse als Anhaltspunkt für den tatsächlichen Nutzen der Optimierung betrachtet werden können.

4.5.1 Theoretische Abschätzung

Um abschätzen zu können, wie groß der zu erwartende Nutzen der Verbesserung sein wird, ist es sinnvoll, zunächst die Ersparnis bei gewissen fest gewählten Parametern zu berechnen. Seien also (wie in den Grundeinstellungen des *Intelligent Miner for Data*) die Ähnlichkeitsschranke $\alpha = 0,5$ und alle Gewichte $w_l = 1, (l = 1, \ldots, p)$. Die Verteilung der Ähnlichkeiten in den Clustern sei folgendermaßen:



Man geht also von sehr homogenen Clustern aus, was zu hohen oberen Schranken für die maximale Ähnlichkeit bzw. den maximalen Score führt. Die Fläche unter der Kurve muss 1 ergeben, sodass man als Dichtefunktion f der obigen Verteilung die Funktion

$$f(x) = -8x + 8$$

erhält. Der Mittelwert μ der Verteilung beträgt dann

$$\mu = \int_{0.5}^{1} x(-8x + 8)dx = \frac{2}{3}$$

Ein Datensatz $X_j \in C_q$ hat also eine durchschnittliche Ähnlichkeit von $\frac{2}{3}$ zu seinem Cluster C_q. Nimmt man diese Ähnlichkeit für jedes Attribut A_l $(l = 1, \ldots, p)$ an, so ergibt sich für den Score

$$\begin{aligned} score(X_j, C_q) &= \sum_{i \in C_q} score(X_i, X_j) \overset{(4.4.1)}{=} \sum_{i \in C_q} \sum_{l=1}^{p} 1 \cdot score_l(x_{il}, x_{jl}) \\ &= \sum_{l=1}^{p} \sum_{i \in C_q} \left(s_{ij}^l - \frac{1}{2} \right) = \sum_{l=1}^{p} \sum_{i \in C_q} \left(\frac{2}{3} - \frac{1}{2} \right) = \frac{p \cdot n_q}{6} \end{aligned}$$

Wählt man für jeden Attributtyp eine andere Berechnungsart für die obere Schranke, so gehen viele zusätzliche Parameter in die Abschätzung ein, die diese komplex und unübersichtlich machen. Darum soll für alle Attributtypen die gleiche Berechnungsart der oberen Schranke gewählt werden; hierbei bietet sich die Abschätzung mittels der Clustergröße an (siehe Kap. 4.2). Für ein Cluster C_q beträgt die obere Schranke

für den Score also $2pn_q$. Somit brauchen für einen Datensatz aus C_q alle Cluster $C_r, r \in \{1, \ldots, k\}$, (wobei k =Anzahl Cluster) mit

$$2p \cdot n_r \leq \frac{p \cdot n_q}{6} \text{ , also } n_r \leq \frac{n_q}{3}$$

nicht betrachtet werden. Nun muß eine Annahme über die Verteilung der Clustergrößen gemacht werden. Zwei Fälle sollen betrachtet werden:

1. Gleichverteilung der Clustergrößen mit konstantem Abstand

   Der Algorithmus sortiert die Cluster anhand der oberen Schranke für den Score, in diesem Fall ist dies äquivalent zu einer Sortierung nach der Größe, da der oberen Schranke die Clustergröße zugrunde liegt. Sei also $d \in \mathbb{N},\ n_1 = d,\ n_2 = 2 \cdot d, \ldots, n_k = k \cdot d$. Die Anzahl R der Datensätze beträgt dann $R = \sum_{i=1}^{k} i \cdot d = \frac{k(k+1)}{2} \cdot d$. Soll ein Datensatz aus Cluster $C_j,\ j \in \{1, \ldots, k\}$, neu zugeordnet werden, so brauchen alle Cluster C_r mit

   $$r \cdot d = n_r \leq \frac{n_j}{3} = \frac{j \cdot d}{3} \text{ , also } r \leq \frac{j}{3}$$

   nicht betrachtet werden. Die Ersparnis bei der Bearbeitung eines Datensatzes aus C_j beträgt somit

   $$E(j) \geq \frac{\left\lfloor \frac{j}{3} \right\rfloor}{k} \text{ ,}$$

   was folgende Gesamtersparnis ergibt:

   $$\begin{aligned} E_{ges} &\geq \sum_{i=1}^{k} \frac{n_i}{R} \cdot \frac{\left\lfloor \frac{i}{3} \right\rfloor}{k} = \frac{1}{Rk} \sum_{i=1}^{k} id \cdot \left\lfloor \frac{i}{3} \right\rfloor \geq \frac{d}{Rk} \sum_{i=1}^{k} i \left(\frac{i}{3} - 1 \right) \\ &= \frac{d}{\frac{k(k+1)d}{2} \cdot k} \left(\frac{1}{3} \sum_{i=1}^{k} i^2 - \sum_{i=1}^{k} i \right) = \frac{2}{k^2(k+1)} \left(\frac{k(k+1)(2k+1)}{3 \cdot 6} - \frac{k(k+1)}{2} \right) \\ &= \frac{2}{9} - \frac{8}{9k} \end{aligned}$$

   Mit wachsender Clusteranzahl wächst die Ersparnis und nähert sich $\frac{2}{9}$ an. Bei 9 Clustern, was der Standardeinstellung des *IM for Data* entspricht, spart man somit über 12% der Vergleiche ein.

2. Exponentiell wachsende Clustergrößen

   Diese Annahme soll den sehr häufig auftretenden Fall weniger großer und vieler kleiner Cluster modellieren. Wie im obigen Fall werden die Cluster der Größe nach sortiert. Sei $d \in \mathbb{N},\ n_1 = 2^{d-1},\ n_2 = 2^{d-2}, \ldots, n_k = 2^{d-k}$. Die Anzahl der Datensätze beträgt dann

   $$R = \sum_{i=1}^{k} 2^{d-i} = 2^d \sum_{i=1}^{k} \frac{1}{2^i} = 2^d \cdot \left(2 - \frac{1}{2^k} \right).$$

Möchte man einen Datensatz aus Cluster $C_j,\ j \in \{1, \ldots, k\}$, neu zuordnen, so brauchen alle Cluster C_r mit

$$2^{d-r} = n_r \leq \frac{n_j}{3} = \frac{2^{d-j}}{3} \text{ , also } r \geq j + log_2(3)$$

nicht betrachtet werden. Da $j \in \mathbb{N}$ und $1 \leq log_2(3) \leq 2$ ist dies gleichbedeutend mit $r \geq j + 2$. Die Ersparnis für C_j beträgt somit

$$E(j) \geq \frac{k - (j+1)}{k} = 1 - \frac{j+1}{k} \ ,$$

was auf folgende Gesamtersparnis führt:

$$\begin{aligned} E_{ges} &\geq \sum_{i=1}^{k} \frac{n_i}{R} \cdot E(i) = \frac{1}{R} \sum_{i=1}^{k} 2^{d-i} \cdot \left(1 - \frac{i+1}{k}\right) \\ &= \frac{2^d}{2^d \cdot (2 - \frac{1}{2^k})} \left(\left(1 - \frac{1}{k}\right) \left(\sum_{i=1}^{k} \frac{1}{2^i} \right) - \frac{1}{k} \sum_{i=1}^{k} 1 \right) \\ &= \frac{1}{2 - \frac{1}{2^k}} \left(\left(1 - \frac{1}{k}\right) \left(2 - \frac{1}{2^k}\right) - 1 \right) \\ &= 1 - \frac{1}{k} - \frac{1}{2 - \frac{1}{2^k}} \overset{k \geq 2}{\geq} 1 - \frac{1}{k} - \frac{1}{2 - \frac{1}{8}} = \frac{7}{15} - \frac{1}{k} \end{aligned}$$

Mit wachsender Clusteranzahl nähert sich die Ersparnis $\frac{7}{15}$ an. Bei 9 Clustern spart man mehr als 35% der Vergleiche ein.

Es wird deutlich, dass man mit der Optimierung eine große Zahl an Datensatz-Cluster-Vergleichen einsparen kann. In der Praxis ist mit noch größeren Einsparungen zu rechnen, da

- die Parameter für die Abschätzungen eher pessimistisch gewählt wurden. So ist es etwa sehr unwahrscheinlich, dass jeder Datensatz in jedem Attribut eine Ähnlichkeit von $\frac{2}{3}$ zu seinem Cluster hat. In der Regel gibt es selbst in sehr homogenen Clustern Datensätze, die in einigen Attributen eine geringe Ähnlichkeit zum Cluster haben. Des weiteren sinkt die Homogenität der Cluster mit sinkender maximaler Clusteranzahl, da bei Erreichen dieser Clusteranzahl Datensätze, die in keines der vorhandenen Cluster passen, trotzdem in eines dieser Cluster eingefügt werden müssen. (Bei einer höheren maximalen Clusteranzahl würde mit einem solchen Datensatz ein neues Cluster gegründet werden).

- die obere Schranke für den maximalen Score noch verkleinert werden kann, wenn man anstatt der Clustergrößen eine andere Größe nimmt (was bei der Implementierung im Rahmen dieser Diplomarbeit der Fall ist, siehe Kap. 4.4.2).

Nach dieser theoretischen Abschätzung soll nun gezeigt werden, wieviel in konkreten Fällen eingespart werden kann.

4.5.2 Testläufe

Um zu sehen, wieviel tatsächlich gespart werden kann, wurde der optimierte Algorithmus mit verschiedenen Daten getestet.
Folgende Datentabellen wurden verwendet:

1. Daten A Umfang: 500 Datensätze
2. Daten B Umfang: 10000 Datensätze
3. Daten C Umfang: 10000 Datensätze
4. Daten D Umfang: 44810 Datensätze
5. Daten E Umfang: 100000 Datensätze
6. Daten F Umfang: 199523 Datensätze

Bei den Tests wurden zunächst für jede Datentabelle mit Hilfe des *Intelligent Miner for Data* verschiedene Modelle (mit unterschiedlichen Parametern) erstellt. Anschließend wurden die Testdaten unter Verwendung von *IM Scoring* gescort, was bedeutet, dass der Demographic Clustering Algorithmus jeden der Datensätze (aus der entsprechenden Datentabelle) dem ihm ähnlichsten Cluster im Modell zuordnet. Dieser Vorgang wurde für jedes Modell zweimal durchgeführt: zunächst mit dem optimierten Algorithmus, anschließend mit dem ursprünglichen Algorithmus, um zu kontrollieren, ob das Clustering-Ergebnis dasselbe ist. Dies war bei allen Tests der Fall, d.h. die Qualität des Ergebnisses bleibt von der Laufzeitoptimierung unberührt.

Ersparnis bei Verwendung der voreingestellten Parameter

Zunächst wurden alle Daten mit den Default-Werten des *IM for Data* getestet, welche folgendermaßen gesetzt sind:

- Ähnlichkeitsschranke $\alpha = 0,5$
- maximale Clusteranzahl = 9
- maximale Anzahl Durchläufe durch alle Datensätze = 2

Zur Clusterbildung bzw. zum Scoren wurden 10 Attribute verwendet, davon 3 kategorische und 7 stetige. Da nicht alle Testdaten diskret numerische Attribute besitzen, wurden auf diese der besseren Vergleichbarkeit wegen verzichtet (darüber hinaus treten in der Praxis wenige diskret numerische Attribute auf, da die meisten numerischen Attribute, wie zum Beispiel Einkommen, Größe oder Alter, stetig sind).
Folgende Ersparnisse wurden erzielt:

Daten A :	58,33% ,	1875	Vergleiche anstatt	4500
Daten B :	74,90% ,	22585	Vergleiche anstatt	90000
Daten C :	77,19% ,	20527	Vergleiche anstatt	90000
Daten D :	57,76% ,	170343	Vergleiche anstatt	403308
Daten E :	74,73% ,	227441	Vergleiche anstatt	900000
Daten F :	61,99% ,	682425	Vergleiche anstatt	1795707

Wie nach den theoretischen Abschätzungen bereits vermutet, kann man in der Praxis noch mehr einsparen: in diesen Beispielen spart man deutlich mehr als die Hälfte der Datensatz-Cluster-Vergleiche ein, in drei Fällen sogar um die 75%. Die Testdaten sind in der Verteilung der Werte den tatsächlich in der Praxis vorkommenden Daten nachempfunden, jedoch unterscheiden sie sich von diesen im Umfang: in großen Unternehmen müssen Millionen von Datensätzen bearbeitet werden, so dass bei 50% Ersparnis eine gewaltige Zahl an Vergleichen und somit viel Rechenaufwand eingespart werden kann.
Betrachtet man diese Ergebnisse, so liegt die Frage nahe, woran es liegt, dass die Ersparnis bei den verschiedenen Testdaten so unterschiedlich ausfällt. Da die Clustergröße den Wert der oberen Schranke für den Score stark beeinflußt, liegt die Vermutung nahe, dass hier ein Zusammenhang besteht. Dies soll im Folgenden genauer untersucht werden.

Korrelation von Clustergröße und Ersparnis

Der Score eines Datensatzes X_j mit einem Cluster C_q setzt sich zusammen aus den Scores mit den einzelnen Datensätzen im Cluster:

$$score(X_j, C_q) = \sum_{i \in C_q} score(X_i, X_j)$$

Mit wachsender Clustergröße wächst also auch der Score eines Datensatzes X_i mit seinem eigenen Cluster (außer der eingefügte Datensatz ist zwar zu den meisten Datensätzen im Cluster sehr ähnlich, nicht aber zu X_i, was dann der Fall sein kann, wenn X_i „am Rand“ des Clusters liegt). Die obere Schranke für den maximalen Score muss ebenfalls nach oben korrigiert werden.
Dies führt zu folgendem Effekt: der Score, den ein Datensatz aus einem großen Cluster mit seinem eigenen Cluster erzielt, ist relativ hoch (verglichen mit dem Score, den ein Datensatz aus einem kleinen Cluster mit seinem eigenen Cluster erzielt), sodass er über der oberen Schranke für den Score eines kleinen Clusters liegt. Hat man also beispielsweise 2 große und 7 kleine Cluster, so ist es sehr wahrscheinlich, dass die Datensätze aus den großen Clustern nicht mehr mit den kleinen Clustern verglichen werden müssen. Da die großen Cluster einen Großteil der Datensätze enthalten, spart man insgesamt viele Vergleiche ein.
Diese Korrelation zwischen der Clustergröße und der Anzahl der nötigen Vergleiche mit diesem Cluster spiegeln auch die Testdaten wieder. Abbildung 4.3 zeigt für jede der Datentabellen ein Diagramm, in welchem für jedes Cluster seine Größe relativ zum Gesamtdatenvolumen (schwarz) sowie der Anteil der Datensätze, die mit diesem Cluster verglichen werden (grau), aufgeführt sind. Man sieht deutlich, dass ein Abfallen in der Clustergrößen-Kurve mit einem Abfallen der nötigen Vergleiche einhergeht. Dies bestätigt die im vorigen Abschnitt gemachte Vermutung, dass die Gesamtersparnis mit der Verteilung der Clustergrößen zusammenhängt: bei den Testdaten mit einer Ersparnis von ca. 75% gibt es nur 2 große und 7 sehr kleine Cluster, wohingegen die Daten mit einer geringeren Ersparnis 3 große Cluster aufweisen. Datentabelle F bildet hierbei eine Ausnahme: die Gesamtersparnis beträgt 62% und es liegen nur 2 große Cluster vor. Eine mögliche Ursache hierfür könnte sein, dass die großen Cluster sehr

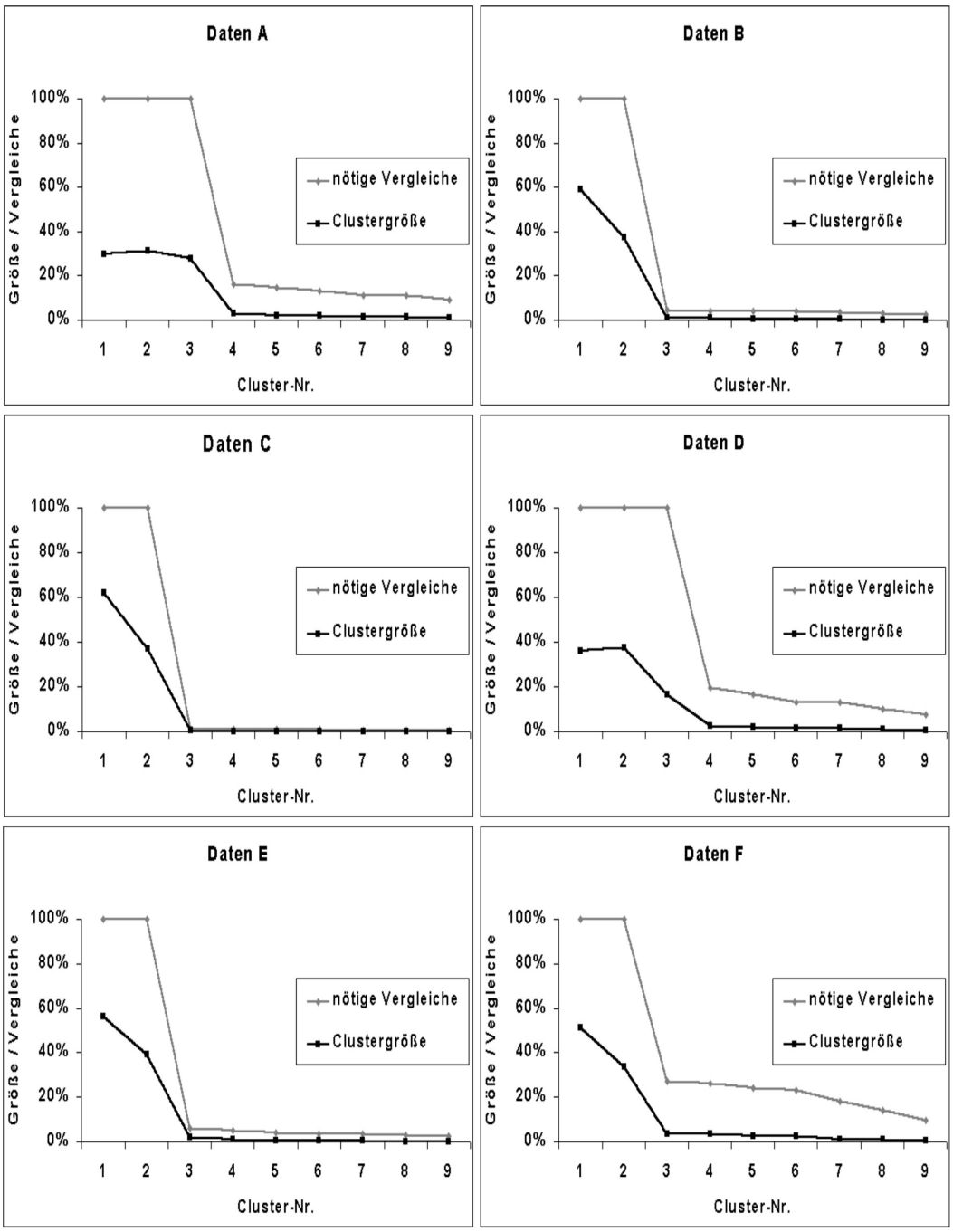


Abbildung 4.3: Korrelation zwischen Clustergröße und der Anzahl nötiger Vergleiche mit dem jeweiligen Cluster

inhomogen und/oder die kleinen Cluster sehr homogen sind. Denn dann kann es sein, dass der Score der Datensätze aus den großen Clustern mit ihrem eigenen Cluster kleiner ist als die obere Schranke für den Score eines kleinen Clusters, sodass die entsprechenden Datensätze auch mit den kleinen Clustern verglichen werden müssen.

Wenn die Verteilung der Clustergrößen so stark mit der Gesamtersparnis korreliert, so stellt sich die Frage, wieviel noch gespart werden kann, wenn die Cluster gleich groß sind, was sich auf die Ersparnis sehr ungünstig auswirken dürfte:

Ersparnis bei annähernd gleich großen Clustern

Hat man starke Größenunterschiede zwischen den Clustern, so werden die meisten Datensätze den großen Clustern zugeordnet. Dies liegt daran, dass bei den großen Clustern meist ein höherer Score erzielt werden kann, auch wenn ein Datensatz eine höhere Ähnlichkeit zu einem kleinen Cluster hat. Hat man also beispielsweise einen Datensatz, der zu einem Cluster der Größe 10000 eine durchschnittliche Ähnlichkeit von 0,6 und zu einem Cluster der Größe 500 eine durchschnittliche Ähnlichkeit von 0,8 hat, so wird er trotzdem dem großen Cluster zugeordnet, weil er mit diesem Cluster einen Score von $10000 \cdot (0,6 - 0,5) = 1000$ erzielt, wohingegen der Score mit dem kleinen Cluster nur $500 \cdot (0,8 - 0,5) = 150$ beträgt. Beim Scoren beginnt man mit den Clustern mit der höchsten oberen Schranke für den maximalen Score (das sind gerade die großen Cluster), sodass also die meisten Datensätze nur mit diesen großen Clustern verglichen werden müssen und dann schon zugeordnet werden können. Sind die Cluster gleich groß, so hat man diesen Effekt nicht. Hier erzielen viele Datensätze den maximalen Score mit einem der hinteren Cluster (in der nach der oberen Schranke für den Score sortierten Clusterfolge), sodass sie also zunächst mit allen vorherigen Clustern verglichen werden müssen (und eventuell auch mit den nachfolgenden, falls deren obere Schranke für den maximalen Score höher ist als der Score mit dem „besten“ Cluster). Dies führt natürlich zu einer geringeren Ersparnis, was sich auch bei den Testdaten widerspiegelt: Abbildung 4.4 zeigt beispielhaft das Ergebnis für zwei der Datentabellen. Die Parameter wurden folgendermaßen verändert: die Ähnlichkeitsschranke wurde auf 0,9 gesetzt; bei Datentabelle B wurden 6 kategorische und 30 stetige Attribute, bei Datentabelle F wurden 14 kategorische, 1 diskret numerisches und 3 stetige Attribute verwendet. Auswahlkriterium für die Attribute war eine Gleichverteilung der Werte über den jeweiligen Wertebereich, denn dies trägt dazu bei, die Größenunterschiede zwischen den Clustern gering zu halten.
Die Gesamtersparnis beträgt bei

Datentabelle B	27,50 %
Datentabelle F	28,33 %.

Dies ist deutlich weniger als in den vorigen Testläufen, man spart aber doch immerhin noch deutlich mehr als ein Viertel der Vergleiche ein. Andere Tests mit (annähernd) gleich großen Clustern lieferten ähnliche Ergebnisse.

Die geringen Größenunterschiede wurden unter anderem durch Erhöhen der Ähnlichkeitsschranke erzielt. Darum soll als nächstes untersucht werden, wie sich eine Modifikation der Ähnlichkeitsschranke auf die Ersparnis auswirkt.

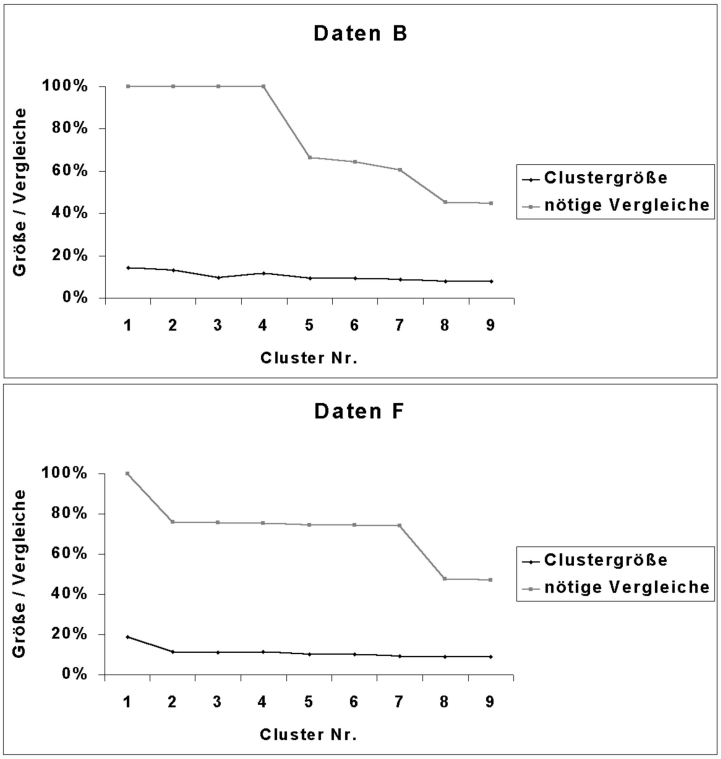


Abbildung 4.4: Ersparnis pro Cluster bei annähernd gleich großen Clustern

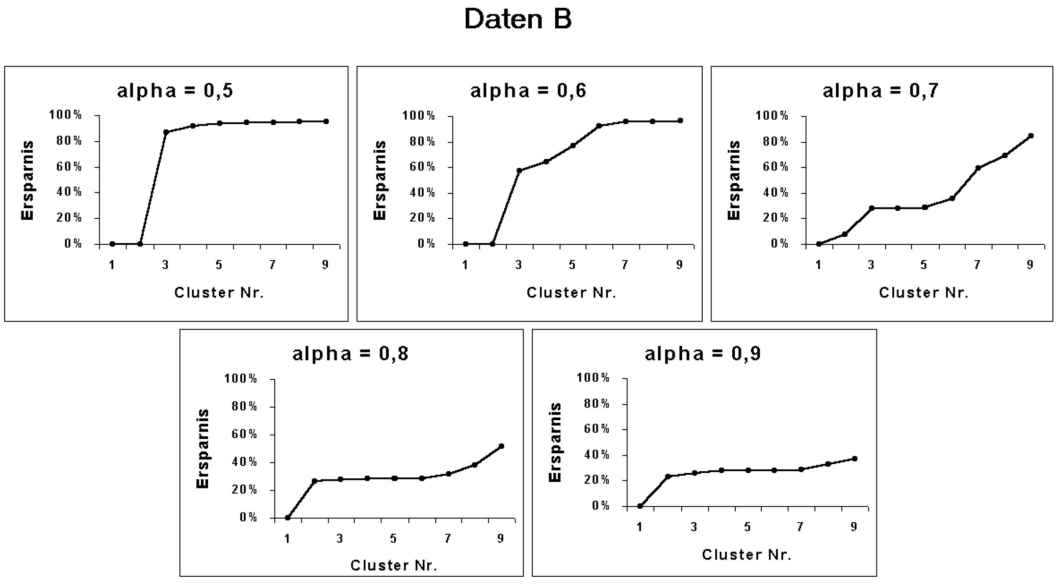


Abbildung 4.5: Ersparnis pro Cluster bei verschiedenen Ähnlichkeitsschranken (Datentabelle B)

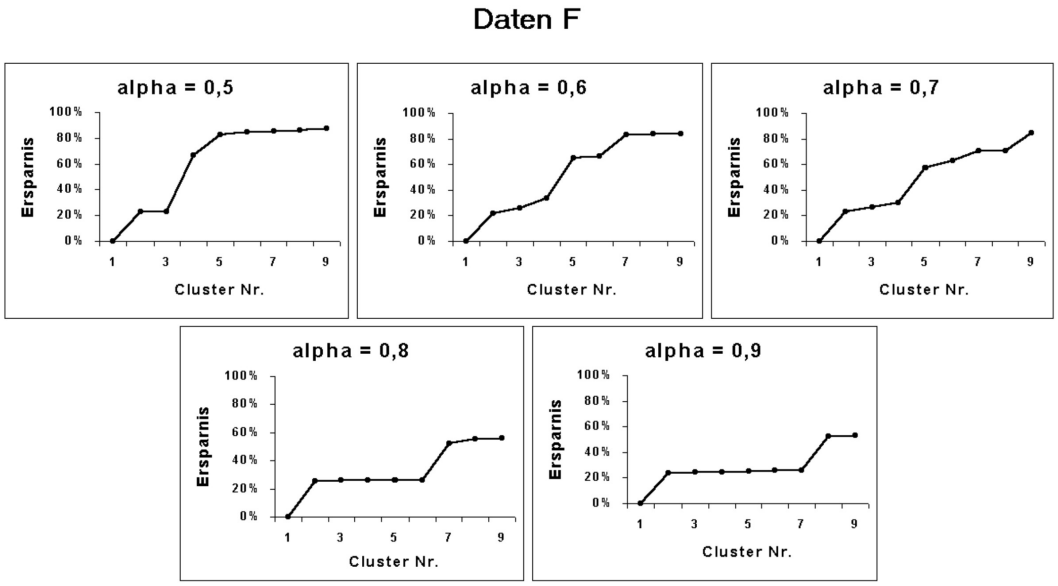


Abbildung 4.6: Ersparnis pro Cluster bei verschiedenen Ähnlichkeitsschranken (Datentabelle F)

Einfluß der Ähnlichkeitsschranke α

Erhöht man die Ähnlichkeitsschranke, so bedeutet dies, dass zwei Datensätze eine höhere Ähnlichkeit haben müssen, um als ähnlich zu gelten, d.h. um einen nichtnegativen Score zu erzielen. Der im vorigen Abschnitt beschriebene Effekt der „stärkeren Anziehungskraft“ großer Cluster wird dadurch abgeschwächt. Zur Veranschaulichung soll das Beispiel des vorigen Abschnitts noch einmal aufgegriffen werden: Ein Datensatz X_j habe eine durchschnittliche Ähnlichkeit von 0,6 zu einem Cluster C_q mit $n_q = 10000$ und eine durchschnittliche Ähnlichkeit von 0,8 zu einem Cluster C_r mit $n_r = 500$. Bei einer Ähnlichkeitsschranke von 0,5 wird X_j dem Cluster C_q zugeordnet. Erhöht man die Schranke aber um 0,1, so ändert sich dies. Denn dann beträgt der Score für das große Cluster $10000 \cdot (0,6 - 0,6) = 0$, der Score für das kleine Cluster jedoch $500 \cdot (0,8 - 0,6) = 100$, sodass der Datensatz dem wesentlich kleineren Cluster C_r zugeordnet wird. Eine weitere Erhöhung von α vergrößert die Differenz in den Scores noch:

α	$score(X_j, C_q)$	$score(X_j, C_r)$
0,5	1000	150
0,6	0	100
0,7	-1000	50
0,8	-2000	0
0,9	-3000	-50

Die großen Cluster verlieren also an „Anziehungskraft“, sodass sich die Größenunterschiede zwischen den Clustern verringern. Wie im vorigen Abschnitt gesehen, führt dies zu einer geringeren Ersparnis. Die Abbildungen 4.5 und 4.6 zeigen für die Da-

tentabellen B und F die Ersparniskurven für verschiedene Ähnlichkeitsschranken. Werte kleiner als 0,5 wurden nicht betrachtet, da diese zu sehr inhomogenen Clustern führen, welche wenig aussagekräftig sind. Bei Datentabelle B wurden 5 kategorische und 7 stetige Attribute zum Clustern verwendet, bei Datentabelle F 14 kategorische, 1 diskret numerisches und 3 stetige Attribute. Bei der Auswahl der Attribute sollten einerseits möglichst viele der zur Verfügung stehenden Attribute verwendet werden; andererseits sollte eine „extreme“ Clusterstruktur, also z.B. ein Riesen-Cluster mit einer Größe von mehr als 50 % der gesamten Datenmenge, vermieden werden. Die gemachte Auswahl kann als Kompromiss dieser beiden Ziele verstanden werden. Für die maximale Clusterzahl und die maximale Anzahl an Durchläufen durch die Daten wurden die Defaultwerte 9 und 2 gewählt.
Die Gesamtersparnis beträgt:

Ähnlichkeitsschranke	Ersparnis	
	Datentabelle B	Datentabelle F
0,5	72,37 %	59,93 %
0,6	64,41 %	51,55 %
0,7	37,75 %	47,31 %
0,8	28,99 %	32,59 %
0,9	25,87 %	28,33 %

An der Gesamtersparnis, wie auch an den Kurven in den Abbildungen 4.5 und 4.6, kann man erkennen, dass eine Erhöhung der Ähnlichkeitsschranke eine Verringerung der Ersparnis zur Folge hat. Zwar kann man bei den ersten Clustern (in der nach der oberen Schranke für den maximalen Score sortierten Cluster-Folge) eventuell Vergleiche einsparen, da diese Cluster kleiner werden (siehe z.B. Cluster Nr. 2 bei Datentabelle B), dafür müssen wesentlich mehr Datensätze mit den hinteren Clustern verglichen werden, sodass insgesamt die Ersparnis kleiner wird. Doch auch bei einer Ähnlichkeitsschranke von 0,9 läßt sich immer noch mehr als ein Viertel der Vergleiche einsparen. Tests mit anderen Datentabellen führten zu ähnlichen Ergebnissen.

Einfluß der maximalen Clusteranzahl

Als letztes soll nun noch untersucht werden, wie sich eine Veränderung der maximalen Clusteranzahl auf die Ersparnis auswirkt. Die Parameter wurden gleich gewählt wie für die Tests im vorigen Abschnitt, mit dem Unterschied, dass die Ähnlichkeitsschranke nicht variiert, sondern auf 0,5 festgesetzt wurde. Wegen der besseren Vergleichbarkeit werden wieder die Ergebnisse für die Datentabellen B und F angeführt, welche in den Abbildungen 4.7 und 4.8 zu sehen sind.

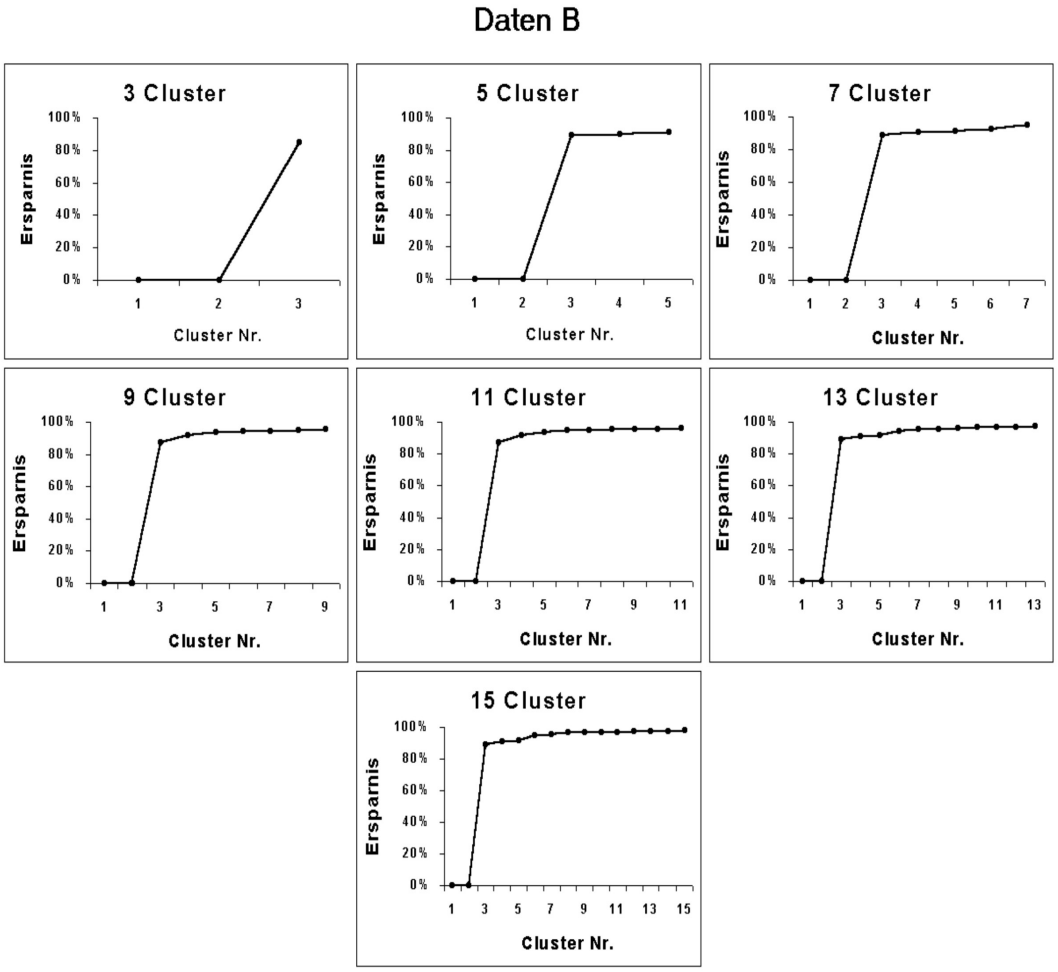


Abbildung 4.7: Ersparnis pro Cluster bei verschiedenen maximalen Clusterzahlen und Ähnlichkeitsschranke 0,5 (Datentabelle B)

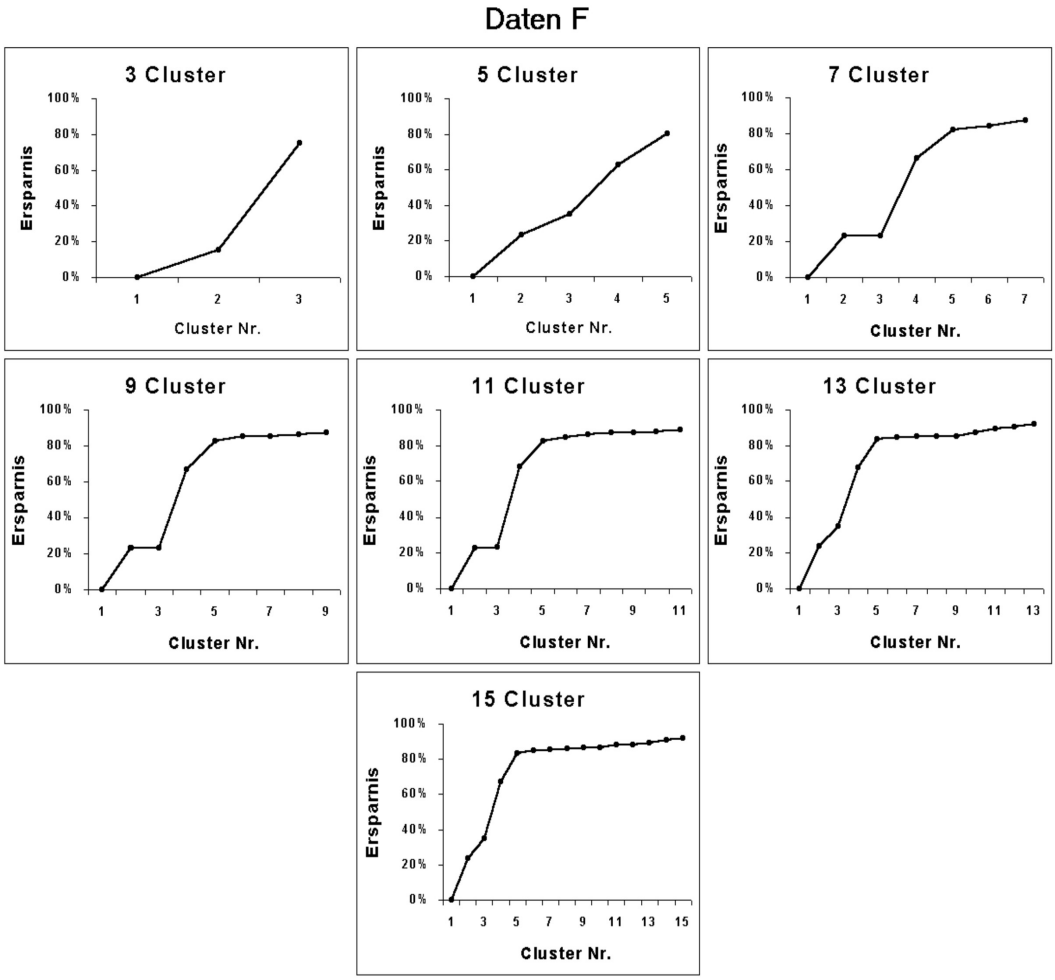


Abbildung 4.8: Ersparnis pro Cluster bei verschiedenen maximalen Clusterzahlen und Ähnlichkeitsschranke 0,5 (Datentabelle F)

Für die Gesamtersparnis ergaben sich folgende Werte:

Clusteranzahl	Ersparnis	
	Datentabelle B	Datentabelle F
3	28,27 %	30,23 %
5	53,99 %	40,32 %
7	65,45 %	52,31 %
9	72,37 %	59,93 %
11	76,81 %	65,38 %
13	80,11 %	69,93 %
15	82,53 %	72,49 %

Deutlich erkennt man, wie die Ersparnis mit wachsender Clusteranzahl steigt. Dies liegt wohl an der im vorletzten Abschnitt erläuterten „Anziehungskraft“ großer Cluster und an der Tatsache, dass die maximale Clusteranzahl fast immer ausgeschöpft wird (bei den Testläufen wurde sie immer erreicht). Denn es gibt fast immer Datensätze, die in keines der bisher vorhandenen Cluster hineinpassen, d.h. die mit jedem dieser Cluster einen negativen Score haben. Mit solchen Datensätzen wird ein neues Cluster gegründet, falls die maximale Clusteranzahl noch nicht erreicht ist. Da aufgrund der „Anziehungskraft“ der großen Cluster aber die meisten Datensätze den großen Clustern zugeordnet werden, ändert sich bei Erhöhen der maximalen Clusteranzahl die Zahl der großen Cluster nur wenig, während viele kleine Cluster entstehen. Da viele Datensätze bereits nach Vergleich mit den großen Clustern zugeordnet werden können, spart man mit wachsender Zahl kleiner Cluster folglich viele Vergleiche ein, was die hohen Ersparnisse bei hohen Clusteranzahlen erklärt. Die Testergebnisse bestätigen dies: bei Datentabelle B enthält bei variierender maximaler Clusteranzahl jede der Clusterungen 2 große Cluster, weswegen bei diesen beiden Clustern nichts eingespart wird (siehe Abb. 4.7). Die übrigen Cluster hingegen sind sehr klein, sodass hier hohe Einsparungen gemacht werden können. Bei wachsender maximaler Clusteranzahl erhöht sich lediglich die Anzahl kleiner Cluster, die der großen bleibt konstant bei 2. Erhöht man jedoch gleichzeitig die Ähnlichkeitsschranke, was die „Anziehungskraft“ großer Cluster abschwächt (siehe voriger Abschnitt), so ändert sich die Größenverteilung und damit auch die Ersparnis. Abbildung 4.9 zeigt die Ersparniskurven für eine Ähnlichkeitsschranke von 0,7.

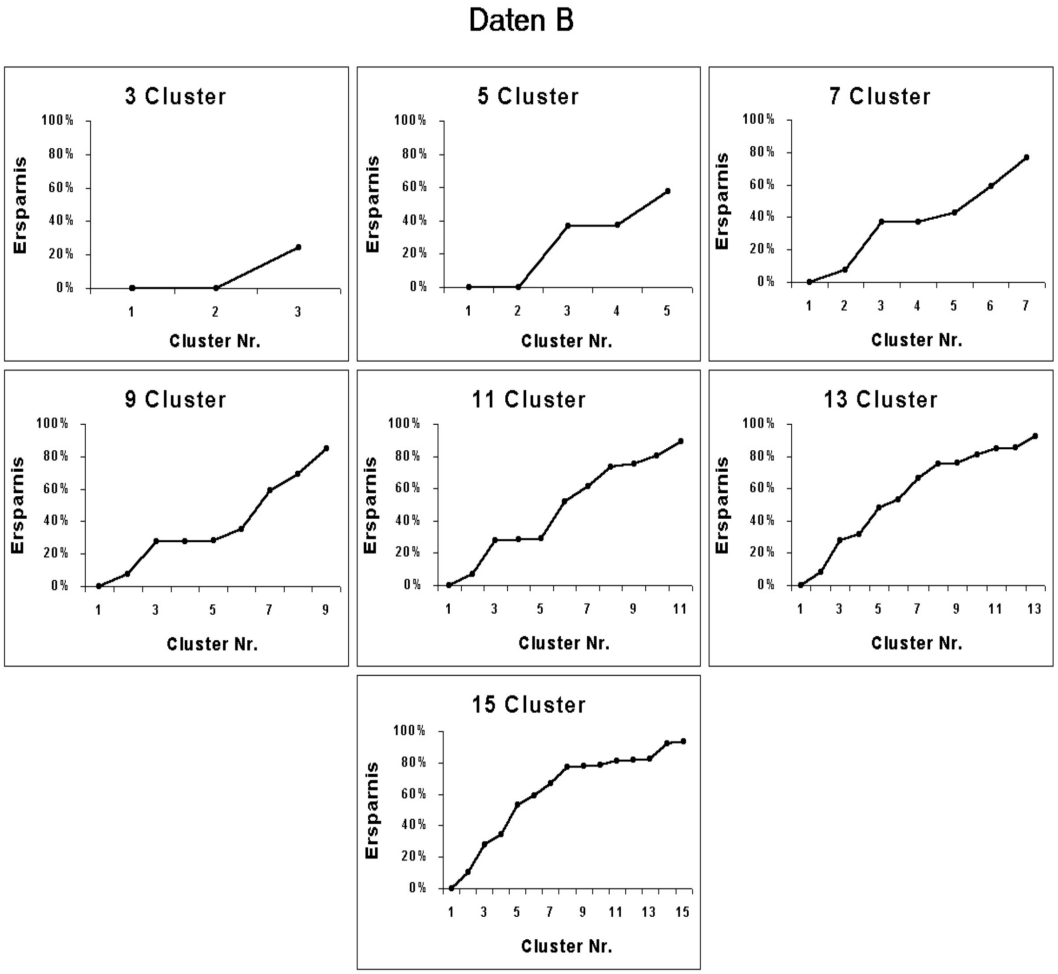


Abbildung 4.9: Ersparnis pro Cluster bei verschiedenen maximalen Clusterzahlen und Ähnlichkeitsschranke 0,7 (Datentabelle B)

Die Gesamtersparnis ändert sich dann folgendermaßen:

Clusteranzahl	Ersparnis	
	Datentabelle B	Datentabelle F
3	8,06 %	20,31 %
5	26,43 %	24,52 %
7	37,21 %	31,39 %
9	37,75 %	47,31 %
11	47,66 %	49,05 %
13	56,18 %	49,59 %
15	61,15 %	53,51 %

Bei vielen Data Mining-Anwendungen liegt das Hauptinteresse auf den großen Clustern. Clustert man beispielsweise die Kunden einer Bank, um charakteristische Kundengruppen zu finden, auf die dann spezifischer eingegangen werden soll, so wird das Hauptaugenmerk auf den großen Kundenclustern liegen, da sich bei kleinen Clustern spezifische Maßnahmen nicht lohnen würden.[4] Um die Bildung zu vieler kleiner Cluster zu vermeiden, ist es sinnvoll bei einer Erhöhung der maximalen Clusteranzahl auch die Ähnlichkeitsschranke höher zu wählen. Dies verringert zwar die Ersparnis, trotzdem kann man davon ausgehen, dass man die Hälfte der Vergleiche einsparen kann. (Unter Erhöhung der maximalen Clusteranzahl wird hier die Wahl eines Wertes größer als der default-Wert von 9 verstanden).

[4]Natürlich gibt es auch Anwendungen, bei denen man an kleinen Clustern interessiert ist, beispielsweise, wenn man die Kunden einer Versicherung clustert, um potenzielle Versicherungsbetrüger zu erkennen (die ja nur einen sehr kleinen Teil der Versicherungsnehmer ausmachen). Aber auch in diesem Fall ist man nicht an sehr vielen kleinen Clustern interesssiert.

Kapitel 5

Zusammenfassung

Der Demographic Clustering Algorithmus ist ein im *IBM DB2 Intelligent Miner for Data* implementierter Cluster-Analyse-Algorithmus. Im Rahmen dieser Diplomarbeit wurde untersucht, wie sich die Laufzeit des Algorithmus verbessern lässt. Es wurde festgestellt, dass ein Datensatz nicht zwangsläufig mit jedem Cluster verglichen werden muss, um das ähnlichste Cluster zu finden, dass also die Schleife über alle Cluster unter gewissen Umständen früher abgebrochen werden kann. Verschiedene Abbruchkriterien wurden vorgeschlagen und diskutiert. Drei der Vorschläge wurden auch implementiert (für jeden Attributtyp jeweils einer). Hierbei war entscheidend, dass an den wesentlichen Kontrollstrukturen des Algorithmus nichts verändert werden musste und dass alle nötigen Berechnungen vor der Schleife über alle Datensätze durchgeführt werden konnten, da sonst trotz gesparter Datensatz-Cluster-Vergleiche die Gesamtlaufzeit höher wäre als zuvor.
Eine theoretische Abschätzung über die zu erwartende Einsparung an Datensatz-Cluster-Vergleichen (mit eher pessimistisch gewählten Parametern) ergab eine Ersparnis von über 12% bei gleichverteilten Clustergrößen mit konstantem Abstand und eine Ersparnis von über 35% bei exponentiell wachsenden Clustergrößen. Bei Tests mit verschiedenen Testdaten wurden Einsparungen von bis zu 80% erzielt. Die Höhe der Ersparnis ist abhängig von der Verteilung der Attributwerte und von der Parameterwahl, hierbei insbesondere von der Höhe der Ähnlichkeitsschranke und der maximalen Clusteranzahl. Doch auch wenn diese Werte ungünstig gewählt wurden (also eine hohe Ähnlichkeitsschranke und eine geringe maximale Clusteranzahl), so wurden trotzdem noch ca. 25% der Vergleiche eingespart. Insgesamt kann mit einer durchschnittlichen Ersparnis von 40% gerechnet werden.

Literaturverzeichnis

[And73] Michael R. Anderberg. *Cluster Analysis for Applications.* Academic Press, Inc. New York, 1973.

[AS68] Milton Abramowitz and Irene A. Stegun. *Handbook of Mathematical Functions.* Dover Publications, 1968.

[Boc74] Hans Hermann Bock. *Automatische Klassifikation.* Vandenhoeck und Ruprecht, 1974.

[Bos98] Karl Bosch. *Statistik-Taschenbuch.* R. Oldenbourg Verlag München, 1998.

[Ca00] Pete Chapman(NCR) and al. Crisp-dm 1.0 - step by step data mining guide. CRISP-DM, 1999, 2000.

[CHS+98] Peter Cabena, Pablo Oscar Hadjinian, Rolf Stadler, Dr. Jaap Verhees, and Alessandro Zanasi. *Discovering Data Mining - From Concept To Implementation.* Prentice Hall PTR, 1998.

[Das00] Parsis Dastani. *Data Mining - eine Einführung.* 2000. www.database-marketing.de/miningmining.htm.

[Dil01] Ruth Dilly. Data mining - an introduction, February 2001.

[dV01] Barry de Ville. *Microsoft Data Mining.* Butterworth-Heinemann, 2001.

[Eck80] Thomas Eckes. *Clusteranalysen.* W.Kohlhammer GmbH, 1980.

[ES00] Martin Ester and Jörg Sander. *Knowledge Discovery in Databases.* Springer Verlag, 2000.

[Eve74] Brian Everitt. *Cluster Analysis.* Heinemann Educational Books Ltd, 1974.

[Fis89] Marek Fisz. *Wahrscheinlichkeitsrechnung und mathematische Statistik.* VEB Verlag Deutscher Wissenschaften, 1989.

[GR98] Johannes Grabmaier and Andreas Rudolph. *Techniques of Cluster Algorithms in Data Mining.* IBM Informationssysteme GmbH, December 1998.

[KR90] Leonard Kaufman and Peter R. Rousseeuw. *Finding Groups in Data. An Introduction to Cluster Analysis.* John Wiley & Sons, 1990.

[KWZ98] Daniela Krahl, Ulrich Windheuser, and Friedrich-Karl Zick. *Data Mining - Einsatz in der Praxis*. Addison Wesley Longman Verlag GmbH, 1998.

[Mai01] Thilo Maier. *Mathematische Analyse des Demographic Clustering-Verfahrens*. Master's thesis, Universität Stuttgart, 2001.

[Mer01] R. Merkl. *Genetische Algorithmen*. May 2001.

[Mic87] Pierre Michaud. Condorcet - a man of the avant-garde. *Applied Stochastic Models and Data Analysis*, 3:173–189, 1987.

[Sch90] Rainer Schlittgen. *Einführung in die Statistik*. R. Oldenbourg Verlag München, 1990.

[SL77] Detlef Steinhausen and Klaus Langer. *Clusteranalyse - Einführung in die Methoden und Verfahren der automatischen Klassifikation*. Walter de Gruyter & Co., 1977.

[Sp"75] Helmuth Späth. *Cluster-Analyse-Algorithmen zur Objektklassifizierung und Datenreduktion*. R. Oldenbourg Verlag München, 1975.

[Sp"77] Helmuth Späth. *Fallstudien Cluster-Analyse*. R. Oldenbourg Verlag München, 1977.

[WF01] Ian H. Witten and Eibe Frank. *Data Mining - Praktische Werkzeuge und Techniken für das maschinelle Lernen*. Carl Hanser Verlag, 2001.

Urhebervermerk

Ich versichere, dass ich die vorliegende Arbeit selbständig angefertigt habe und nur die angegebenen Hilfsmittel und Quellen verwendet wurden.

Konstanz, im Januar 2003

Silke Wagner