Universität Rostock
Fachbereich Informatik
Institut für Computergraphik

Konzeption und Realisierung einer flexiblen Pipeline zur numerischen Vorverarbeitung in der Informationsvisualisierung

STUDIENARBEIT
von
Thomas Nocke

Betreuer: Prof. Dr. Heidrun Schumann
Matthias Kreuseler

Abgabedatum: 7. Juli 1999

Inhaltsverzeichnis

Kapitel 1

Einführung und Motivation

Eine immer größer werdende Informationsflut und ein immer größer werdendes Interesse, ihr Herr zu werden, ist Ursache dafür, daß heute immer neue Techniken entwickelt werden, um Kerninformationen zu extrahieren und diese mit geeigneten Visualisierungstechniken für den Menschen nutzbar zu machen. Im Umfeld der wissenschaftlichen Visualisierung erfolgt häufig eine Visualisierung der gesamten Datenmenge. Dies ist jedoch gerade bei sehr großen Datenmengen unbefriedigend. Idee dieser Arbeit ist daher, eine vorverarbeitende Informationsreduktion für die Visualisierung durchzuführen, wobei die wesentlichen Charakteristika der Daten erhalten bleiben sollen.

Die Notwendigkeit einer Vorverarbeitung ergibt sich daraus, daß die meisten Visualisierungstechniken bei der Darstellung sehr großer Datenmengen überfordert sind. Sie besitzen zwar die Fähigkeit, die Daten mittels visueller Anordnung und geeigneter Attributierung für den Benutzer anschaulich darzustellen, oder anders ausgedrückt, komplexe inhaltliche Beziehungen visuell aufzudecken. Jedoch versagen viele Visualisierungsansätze bei zu großen Informationsmengen, weil die Navigationsfähikeit und Übersichtlichkeit der Visualisierung stark abnimmt, wenn zu viele Daten gleichzeitig dargestellt werden. Dieses Problem kann während der Vorverarbeitung gelöst werden. Ein interessanter Ansatz hierbei ist, mit Hilfe von Klassifikationstechniken Datenobjekte mit ähnlichen Eigenschaften zusammenzufassen. Das heißt, daß die Datenmenge in mehrere Klassen mit „ähnlichen“ Elementen aufgeteilt wird. Diese Klassen können dann mit den Visualisierungsmechanismen des Visualisierungssystems getrennt dargestellt werden und sind, weil es sich jetzt um wesentlich kleinere Datenmengen handelt, besser zu überblicken. Bei dieser Vorgehensweise muß jedoch sichergestellt werden, daß man sich jederzeit die Elemente einer Datenklasse anzeigen lassen kann, damit keine Detailinformationen verloren gehen.

Ziel dieser Studienarbeit ist es, grundlegende Verfahren der Informationsaufbereitung mit dem Schwerpunkt der Clusteranalyse und Klassifikation

von Objektmengen[1] zu untersuchen, aus den untersuchten Verfahren für die Anwendung in der wissenschaftlichen Informationsvisualisierung relevante auszuwählen und eine Auswahl dieser Verfahren in einer möglichst flexiblen Art und Weise umzusetzen. Praktisches Ziel war die Implementation eines modular aufgebauten und leicht um neue Verfahren und mathematische Funktionen erweiterbaren Werkzeugs, welches als Vorverarbeitungsstufe eines komplexen Visualisierungssystems genutzt werden soll.

Das Grundproblem hierbei ist, daß eine Vielzahl von Verfahren, Algorithmen und Maßen einer ebenso großen Vielzahl von unterschiedlichen Datentypen und Klassifikationszielen gegenübersteht. Daraus ergibt sich die Notwendigkeit einer flexiblen Steuerung, die bei bestimmten Anforderungen *automatisch* passende Verfahren auswählt und durchführt. Ziel ist, den Benutzer unabhängig von der Algorithmenebene Anfragen in Form von Anforderungen stellen zu lassen, ohne daß er sich mit den Algorithmendetails auszukennen braucht. Um diese verschiedenen Anforderungen erfüllen zu können, wurde ein Ablaufschema entworfen, welches die Daten- und Steuerströme festlegt. Dieses Schema soll als „Vorverarbeitungspipeline“ bezeichnet werden. Gegenstand der Arbeit ist die Konzeption und Umsetzung dieser Pipeline.

Die Struktur der Arbeit orientiert sich an dieser Pipeline. Im ersten Teil (Kapitel 2 - 3) erfolgt die Problemeingrenzung und die Definition von wichtigen Begriffen. Im Anschluß werden die theoretischen Voraussetzungen für die drei Hauptschritte der Pipeline, den Präprozeß, die Wahl der zugrundeliegenden Maße sowie die Wahl und Durchführung der Klassifikation, gelegt. Vor allem geht es darum, Verfahren, Funktionen und Parameter der einzelnen Pipelineschritte vorzustellen, um diese flexibel und optimal je nach Dateneigenschaften und Untersuchungsziel einsetzen zu können. Die Ausführungen im ersten Teil reflektieren in erster Linie bekannte Ansätze aus der Literatur. Diese Ansätze sind vor allem aus gängigen Klassifikationslehrbüchern entnommen (vgl. [Boc74] und [BEPW96]). Weil nicht vorausgesetzt werden kann, daß sie im Umfeld „Visualisierung“ ausreichend bekannt sind, ist dieser Teil der Arbeit ausführlicher abgefaßt.

Im zweiten Teil (Kapitel 4 - 5) wird die Umsetzung der im vorherigen Kapitel erarbeiteten Konzepte beschrieben. Dort werden die implementierten Verfahren, deren flexible Einbindung in ein erweiterbares Konzept und die damit erzielten Ergebnisse vorgestellt.

[1] Objekte werden durch eine Menge von Eigenschaften, die für alle Objekte definiert sind, charakterisiert.

Kapitel 2

Begriffe und Problemstellung

Bevor die verschiedenen Techniken vorgestellt werden, ist es wichtig, einige grundlegende Begriffe zu klären und die Problemstellung deutlicher zu umreißen.

Vorverarbeitungspipeline Den Hauptfaden dieser Arbeit bildet die Vorverarbeitungspipeline. Diese beinhaltet, wie in Kapitel 1 bereits angedeutet wurde, die drei Hauptschritte der Vorverarbeitung (Abb. 2.A). Vor

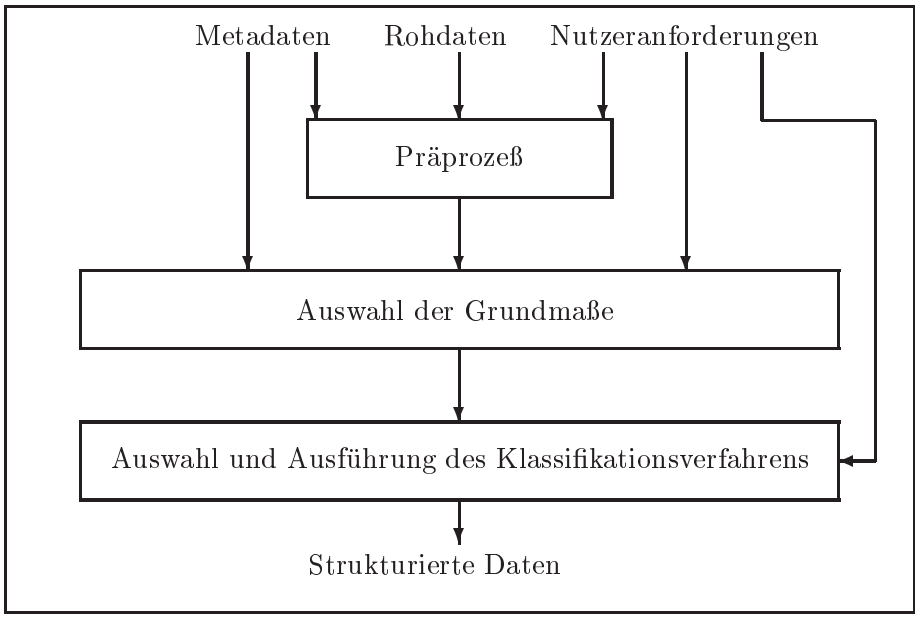

Abbildung 2.A: Vorverarbeitungspipeline für die Visualisierung

Beginn der Pipelineverarbeitung liegen folgende Informationen vor:

- Rohdaten (in Form einer Menge von Objekten mit bestimmten Merk-

malen)

- Metainformationen über die Rohdaten (z.B. Datentyp, Gewichtung oder Relevanz von Merkmalen)
- Nutzeranforderungen (steuern z.B. Art der Klassifikation und Anforderungen an das Ergebnis)

Diese bilden die Grundlage für die 3 Schritte der Pipeline. In Schritt 1 erfolgt ein *Präprozeß*, in dem Vorverarbeitungen auf den Rohdaten durchgeführt werden. Beispielsweise könnten diese Daten einer Normierung unterworfen werden. Schritt 2 beinhaltet die *Wahl von* der Klassifikation *zugrundeliegenden Maßen bzw. Verfahren* und ist Voraussetzung für Schritt 3, in dem die *Klassifikation der Objekte* erfolgt. Die so gewonnenen Daten sind dann Eingangsdaten für ein Visualisierungssystem.

Das Klassifikationsproblem Schwerpunkt bei der Pipelineverarbeitung hat Schritt 3. Um diesen Schritt genauer beschreiben zu können, soll zunächst das Klassifikationsproblem[1] erläutert werden. Es beinhaltet folgendes:

Eine Menge $S = \{O_1, \ldots, O_N\}$ mit N Objekten bzw. Untersuchungsfällen O_k, welche durch Angabe von p Schlüsseleigenschaften (Merkmalen) gekennzeichnet sind, soll derart in Klassen (bzw. Gruppen) $A_i \subset S$ (mit $A_1 \cup A_2, \ldots \cup A_k = S$) aufgeteilt werden, daß ähnliche Objekte in die gleiche Klasse und unähnliche Objekte in unterschiedliche Klassen einsortiert werden. Dabei sollen sich möglichst homogene Klassen bilden. Damit erhält man eine Einteilung der Objekte, die hilft, versteckte Zusammenhänge aufzudecken und Strukturen zu erkennen.

Allerdings ist bei der Informationsinterpretation Vorsicht geboten: Die mit der Klassifikation verbundene Informationsreduktion, die man in gewissem Sinne auch als Abstraktion verstehen kann, verändert die dem Sachverhalt zugrundeliegenden Informationen. Zum einen muß man sich bewußt sein, daß bei der Reduktion Information verloren geht. Informationsreicher, jedoch nicht für den Menschen informativer, sind die ursprünglichen Daten. Gewinn stellt sich nur ein, wenn sowohl Rohdaten als auch das Klassifikationsschema zusammen betrachtet werden. Zum zweiten besteht die Gefahr, daß durch die Wahl der Verfahren oder durch die gewählten Parameter die Informationen derart vom Benutzer verzerrt werden können, daß „natürliche“ Strukturen verloren gehen oder nicht vorhandene Strukturen erzeugt werden[2].

[1] bzw. Gruppierungsproblem

[2] Die Wahl der Parameter und Verfahren für die Erzeugung von Gruppen ist nicht trivial. Oft ist es nicht eindeutig, welche Klassen gebildet werden. Es besteht das Problem, daß bei Variation von Parametern das Ergebnis häufig nicht mehr stabil ist. (Grundtendenzen können verwischt werden)

Begriffe Klassifikation und Clusterung Im Szenario der Vorverarbeitung für ein Visualisierungssystem können Vorab-Informationen über die Strukturierung der Daten wie Anzahl oder Lage der Zielklassen des zu ordnenden Datenbestandes bekannt sein. Allgemein spricht man von *Klassifikation*, egal ob solche Informationen vorliegen oder nicht. Sind sie nicht vorhanden, so bezeichnet man die Verfahren mit *automatische Klassifikation* bzw. *Clusterung*. Diese beiden Begriffe sollen im folgenden synonym verwendet werden. Im Rahmen dieser Arbeit stehen die Techniken aus dem Bereich der *Clusterung* im Vordergrund.

Flexible automatische oder halbautomatische Klassifikation Der Nutzer soll die Möglichkeit haben, durch Einstellung von Klassifikationszielen Art und Parameter der eingesetzten Verfahren zu steuern. Beispiel hierfür wären z.B. die Forderung einer Vorextraktion von Ausreißern aus der Klassifikation oder die Formfestlegung von sich bildenden Clustern. Ziel der Implementation ist dementsprechend, daß der Nutzer des Systems in die Lage versetzt wird, sich die Information durch Auswahl unterschiedlicher Verfahren abhängig von Versuchsziel und möglichen eigenen Kenntnissen, im Prinzip jedoch *automatisch*, aufbereiten zu lassen.

Art der Daten Wie oben bereits erwähnt, handelt es sich bei den Daten um eine Menge von Objekten (bzw. Untersuchungsfällen) O_k, die durch p Schlüsseleigenschaften charakterisiert werden. Für die computerunterstützte Klassifikation müssen diese Objekte und Schlüsseleigenschaften in eine zahlenquantifizierbare Form überführbar sein, was durchaus nicht immer möglich ist. Wenn die Daten in Form einer Tabelle wie in Tabelle 2.a vorliegen, spricht man von einer *Objekt-Merkmals-Matrix*. Von Daten dieser Art

	Haarfarbe (ordinal)	Größe (nominal)	Alter (intervall-skaliert)	Größe (ratio-skaliert)	Geschlecht (ordinal binär)
Karoline	rot	klein	14	1,50 m	weiblich
Robert	blond	groß	22	1,95 m	männlich
Ivonne	schwarz	mittel	24	1,65 m	weiblich

Tabelle 2.a: Objekt-Merkmals-Matrix mit unterschiedlichen Skalentypen

soll im weiteren Verlauf der Ausführungen ausgegangen werden. Formal ausgedrückt bedeutet das, daß für jedes Objekt $O_k \in S$ ein Vektor $x_k \in \mathcal{R}^p$ bekannt ist, welcher die Merkmalsausprägungen der einzelnen Merkmale für die O_k enthält. Eine äquivalente Darstellung der Objektinformationen stellt die $N \times p$ - Objekt-Merkmals-Matrix (x_{ki}) dar, welche sich auch in Form einer Tabelle darstellen läßt. Die Zeilen der Tabelle entsprechen den Objekten und die Spalten den Eigenschaften (bzw. Attributen oder Merkmalen) der Objekte.

Skalentypen Grundsätzlich unterscheidet man vier *Skalentypen*, in de-

nen die Eigenschaftenwerte vorliegen können. Das sind die Nominal-, die Ordinal-, die Intervall- und die Ratioskala. *Nominalskalen* sind Einteilungen qualitativer Eigenschaftsausprägung ohne Ordung. Beipiel hierfür ist Spalte 1 der Abb. 2.a. Die *Ordinalskala* erlaubt zusätzlich die Aufstellung einer Rangordung mit Hilfe von Rangwerten (Spalte 2 Abb. 2.a). Hierbei ist jedoch noch kein exakter Abstand zwischen den Eigenschaftensausprägungen definiert. Dieser kommt erst bei der *Intervallskala* (Spalte 3 Abb. 2.a) hinzu. Bei ihr erfolgt eine gleichmäßige Einteilung der Meßskala in gleichgroße Skalenabschnitte. Bei den Intervallskalen besitzen die Differenzen zwischen den Daten im Gegensatz zu Nominal- und Ordinalskalen Informationsgehalt. Intervallskalierte Daten erlauben die arithmetischen Operationen Addition und Subtraktion und die statistischen Maße Mittelwert und Standardabweichung, aber nicht die Division. Bei der *Ratioskala* (oder Verhältnisskala) kommt noch ein natürlicher Nullpunkt hinzu, an dem sich das entsprechende Merkmal als „nicht vorhanden“ interpretieren läßt. Bei Daten dieser Art besitzt auch das Verhältnis Informationsgehalt. Sie können mit Operationen aller Art manipuliert werden (Spalte 4 Abb. 2.a). Außerdem oft von Bedeutung sind als Spezialfall von ordinal- oder nominalskalierten Merkmalen binäre Merkmale, die nur zwei unterschiedliche Ausprägungen - z.B. männlich und weiblich - annehmen können (Spalte 5 Abb. 2.a).

Maße für die Klassifikation Den im vorhergehenden Absatz ausgeführten Erläuterungen über Skalentypen von Merkmalen kommt eine entscheidende Bedeutung zu, wenn man sie unter dem Gesichtspunkt von Ähnlichkeit/Unähnlichkeit bzw. Distanz/Nähe von Objekten betrachtet. Ähnlichkeiten bzw. Distanzen sind zumeist Voraussetzungen für die Klassifikation, weil Objekte durch sie erst vergleichbar werden. Wichtig ist die richtige Wahl der für die Skalentypen der Merkmale passenden Ähnlichkeits- bzw. Distanzmaße[3], um nicht bereits hier Fehler in der Klassifikation zu machen. Ähnlichkeitsmaße berechnen die Ähnlichkeit zwischen zwei Objekten; Distanzmaße berechnen die Distanz zwischen zwei Objekten. Distanz- und Ähnlichkeitsmaße werden im folgenden auch als Proxymitätsmaße bezeichnet. Die exakte Definition dieser Maße erfolgt in Kapitel 3.2.1.

Weiterhin ist es notwendig, die Homogenität oder Heterogenität von Objekten einer Klasse[4] und auch den Abstand bzw. die Ähnlichkeit von unterschiedlichen Klassen zu formalisieren, um damit berechenbare Maße als Kriterien für die Klassifikation zu haben (s. Kapitel 3.2.2 und 3.2.3).

[3]Ähnlichkeitsmaße bzw. Distanzmaße bringen die Ähnlichkeit bzw. Distanz zwischen zwei Objekten in eine zahlenquantifizierbare Form, um sie rechentechnisch verarbeiten zu können.

[4]Homogenität innerhalb einer Klasse bedeutet, wie ähnlich sich die Objekte einer Klasse im Mittel sind bzw. wie gut sie zusammenpassen. Heterogenität bedeutet entsprechend, wie unähnlich sie sich im mittel sind.

Kapitel 3

Theoretische Konzeption der Pipeline

In diesem Kapitel wird die in Kapitel 2 vorgestellte Vorverarbeitungspipeline unter dem Aspekt der flexiblen Steuerung aus konzeptioneller Sicht beleuchtet. Diese Betrachtungen, die auf einer Literaturrecherche basieren, sind Voraussetzung für die praktische Umsetzung (Kapitel 4). Hauptaugenmerk liegt hierbei darauf, eine Übersicht über vorhandene Techniken (z.B. Maße und Verfahren) zu erstellen, die wichtigsten Eigenschaften zur Charakterisierung von Techniken vorzustellen, wichtige Techniken exemplarisch aufzuführen und anhand der Eigenschaften dem Leser eine Übersicht zu verschaffen, welche Technik wann genutzt werden sollte. Schwerpunkte sind Abschnitt 3.2, welcher die verschiedenen der Klassifikation zugrundeliegenden Maße mit ihren Eigenschaften diskutiert und Abschnitt 3.3, der die eigentlichen Klassifikationstechniken vorstellt.

Bevor man eine Klassifikation durchführt, sollten grundlegende Überlegungen zu den Daten und den erwünschten Klassifikationszielen gemacht werden. Welcher Art diese Überlegungen sein könnten, wird in den folgenden Kapiteln in Form von Fragen oder zu beachtenden Eigenschaften beschrieben. Ein Beispiel für Fragen allgemeiner Natur, die nicht speziellen Pipelineschritten zugeordnet werden können, wie die folgende, sollte vor Beginn der Pipelineausführung überdacht werden:

- **Ist die Zahl der Objekte für eine Klassifikation ausreichend?** Handelt es sich um eine Stichprobe aus einer größeren Grundgesamtheit und sollen aufgrund der Analyse Rückschlüsse auf die Grundgesamtheit gezogen werden, so ist es wichtig, daß genügend Elemente aus den einzelnen Teilgesamtheiten erhoben wurden. Ist dies nicht vollständig der Fall, so ist auch die Interpretierbarkeit der Ergebnisse eingeschränkt.

3.1 Präprozeß

Dieser Abschnitt entspricht dem ersten Schritt der Vorverarbeitungspipeline. Unter dem Begriff Präprozeß sollen dabei alle die Verfahren verstanden werden, welche der Auswahl und Aufbereitung der Ausgangsdaten dienen, noch bevor die eigentliche Clusteranalyse durchgeführt wird. In Form von Fragen, die sich dem Nutzer stellen, werden typische Vorverarbeitungsschritte vorgestellt:

- **Sollen Ausreißer eleminiert werden?** Da man im allg. nicht weiß, welche und wieviele Klassen sich bilden, sollte man im Falle einer Stichprobe „Ausreißer[1]“ vorher aus der Klassifikation entfernen. Diese könnten sonst die Klassifikation verzerren oder die übrigen Objekte zu stark beeinflussen. Als Methode hierfür wird in [BEPW96] vorgeschlagen, die hierarchische Single-Linkage-Klassifikation(s. Kapitel 3.3.2.3) in einem ausreißerentfernenden Lauf vorzuschalten und danch erst die eigentliche Clusterung durchzuführen. Alternativ wäre die Festlegung einer Abstandsschranke. Als Ausreißer gelten dann alle die Objekte, die durchgängig eine höhere Distanz zu allen anderen Objekten haben als diese Schranke. In [Boc74] wird vorgeschlagen, im Fall einer erforderlichen Ausreißerbeachtung ein Klassifikationsverfahren zu wählen, welches eine „nicht exhaustive“ Clusterung im Verfahren integriert durchführt.

- **Müssen gleiche Objekte ausgeschlossen werden?** Für einige Verfahren ist es wichtig, daß keine zwei gleichen Objekte in der Objektmenge S vorliegen, weil diese den Abstand Null haben würden und dies nicht erlaubt ist. Allgemein ist allerdings zu beachten, daß mehrere Objekte mit gleichen Merkmalen auch anzeigen können, daß an dieser Stelle eine starke Objektkonzentration vorliegt. Mit dem Mehrfachauftreten wichten sie so eine Klasse stärker, was durchaus von Bedeutung sein kann.

- **Sollen alle Merkmale in die Klassifikation einfließen?** Ebenso wie für die Anzahl der heranzuziehenden Objekte gibt es auch für die Zahl der einzubeziehenden Merkmale keine eindeutigen Vorschriften. Der Anwender sollte darauf achten, daß nur solche Merkmale bei der Gruppierung berücksichtigt werden, die aus theoretischen und praktischen Gründen als *relevant* für den zu untersuchenden Sachverhalt anzusehen sind. Merkmale, die für den Gruppierungsprozeß als bedeutungslos gelten, sollten vorher entfernt werden. Darunter fällt z.B. die Entfernung konstanter Merkmale in einem Vorverarbeitungsschritt. Weiterhin kann man beispielsweise bei zwei sehr hochkorrelierten Merkmalen (Korrelation > 0.9) direkt eines der beiden

[1] Ausreißer sind sehr unähnlich zu allen anderen Objekten.

Merkmale in einem Präprozeßschritt eliminieren. Weiterhin gibt es eine Vielzahl von Projektionsmethoden, welche eine vorverarbeitende Merkmalsreduktion bei ratioskalierten Merkmalen in einen Unterraum durchführen. Beispiele hierfür sind die Hauptkomponentenmethode und die Faktoranalyse ([Boc74] S.237 - 248 u. [BEPW96] S.189 ff).

- **Sollen die Merkmale unterschiedlich gewichtet werden?** In der Regel läßt sich im voraus nicht bestimmen, ob die betrachteten Merkmale mit unterschiedlichen Gewichten in die Klassifikation einfließen sollten. Praktisch wird deswegen meistens eine Gleichgewichtung der Merkmale unterstellt. Hierbei zu beachten ist, daß durch hochkorrelierte Merkmale bestimmte Aspekte bei der Klassifikation überbetont werden, was zu einer Verzerrung der Ergebnisse führen kann. Wie oben bereits erwähnt wurde, besteht die Möglichkeit, z.B. eine explorative Faktorenanalyse zur Reduktion korrelierter Variablen auf unabhängige Faktoren durchzuführen. Eine andere Möglichkeit ist, Maße zu verwenden, die etwaige Korrelationen zwischen den Merkmalen beachten(s. in Kapitel 3.2.1 z.B. MAHALANOBIS-Distanz (3.45)).

- **Sind die Merkmale vergleichbar?** Zu einer impliziten Wichtung der Merkmale kann es kommen, wenn die Daten auf unterschiedlichen Skalenniveaus erhoben wurden. Um diese unterschiedlichen Merkmalsausprägungen vergleichbar zu machen, kann eine vorverarbeitende Standardisierung durchgeführt werden. Möglichkeit hier sind die Standardisierung der Werte jedes Merkmals auf das Intervall $(0, 1)$

$$\tilde{x}_{ki} := \frac{x_{ki} - z_i}{u_i - z_i} \quad i = 1, \ldots, p \quad k = 1, \ldots, N \tag{3.1}$$

mit

$$z_i := \min\{x_{ki}\}, \qquad u_i := \max\{x_{ki}\}.$$

Weiterhin kann man auch auf den Mittelwert 0 und die Varianz 1 normieren:

$$\tilde{x}_{ki} := \frac{x_{ki} - \bar{x}_i}{\hat{\sigma}_i} \quad i = 1, \ldots, p \quad k = 1, \ldots, N \tag{3.2}$$

mit

$$\begin{aligned} \bar{x}_i &:= \frac{1}{N} \sum_{k=1}^{N} x_{ki} und \\ \hat{\sigma}_i &:= \sqrt{\frac{1}{N} \sum_{k=1}^{N} (x_{ki} - \bar{x}_i)^2} \end{aligned} \tag{3.3}$$

3.2 Festlegung der zugrundeliegenden Maße

Schritt 2 der Vorverarbeitungspipeline beinhaltet die Auswahl von Ähnlichkeits-, Distanz-, Heterogenitäts- und Homogenitätsmaßen sowie die Auswahl von Verfahren zur Bestimmung typischer Objekte. Deswegen soll dieses Kapitel grundlegende Eigenschaften dieser Maße bzw. Verfahren vorstellen, um eine Entscheidungsunterstützung bei der Auswahl von geeigneten Maßen bzw. Verfahren zu erhalten. Diese erfolgt in Äbhängigkeit der vom Nutzer festgelegten Anforderungen an die Clusterung, in Abhängigkeit von den Skalentypen der Daten und an die Erfordernisse des eingesetzten Clusterungsverfahrens[2].

Während Proximitätsmaße erst einmal Ähnlichkeiten oder Distanzen zwischen zwei Objekten bestimmen, ermitteln Homogenitäts- und Heterogenitätsmaße, wie homogen bzw. heterogen eine Objektmenge ist. Zusätzlich ist es ebenfalls sinnvoll, Ähnlichkeiten bzw. Distanzen zwischen Objektgruppen zu definieren. Für einige Verfahren ist es weiterhin erforderlich, typische Objekte einer Objektmenge zu bestimmen. Schwerpunkt der Betrachtungen liegt auf der Untersuchung von Proximitätsmaßen, da diese meist Grundlage für die anderen Maße und Verfahren sind.

Im Rahmen dieser Arbeit kann die Liste der betrachteten Maße bzw. Verfahren nur ausschnitthaft sein. Für weitere Maße bzw. Verfahren und vertiefende Betrachtungen sei hier vor allem auf [Boc74] und [BEPW96] verwiesen.

3.2.1 Proximitätsmaße für Objekte

In diesem Abschnitt sollen zuerst grundlegende Eigenschaften von Proximitätsmaßen vorgestellt und dann für die Auswahl relevante Eigenschaften angegeben werden. Als nächstes erfolgt die Vorstellung von wichtigen Maßen und deren Eigenschaften für die wichtigsten Skalentypen (binär, ordinal und nominal mehrstufig und ratioskaliert). Praktisch wichtig sind die dieses Kapitel abschließenden Bemerkungen zu Strategien beim Vorgehen im Fall von hybriden Merkmalstypen. Die hier gemachten Ausführungen beziehen sich vor allem auf [Boc74] S.24-80, weil dort in der untersuchten Literatur am ausführlichsten und systematischsten auf das Problem der Proximitätsmaße eingegangen wurde.

3.2.1.1 Mathematische Definition der Begriffe Ähnlichkeit und Distanz

Im Gegensatz zur intuitiven alltäglichen Vorstellung von „ähnlichen“ und „unähnlichen“ Dingen, bedarf es bei der computerbasierten Informationsver-

[2]Dieses kann z.B. nur mit Heterogenitätsmaßen arbeiten.

arbeitung der formalisierten Ähnlichkeit s_{jk}, welche die Ähnlichkeit zweier Objekte oder (Untersuchungs-)Fälle mit einem exakten Zahlenwert angibt. Dieser Zahlenwert variert üblicherweise kontinuierlich zwischen 0 (kleinste Ähnlichkeit) und 1 (größte Ähnlichkeit). Formalisiert kann man die Eigenschaften von s_{jk} angeben, indem man fordert, daß $\forall j,k$ mit $1 \leq j,k \leq N$ gilt:

$$\begin{aligned} s_{kj} &= s_{jk} \quad \text{(Symmetrie)} && (3.4)\\ s_{jk} &\geq 0 && (3.5)\\ s_{jk} &\leq s_{jj} && (3.6)\\ s_{jj} &= 1 && (3.7) \end{aligned}$$

Die durch dieses Maß bestimmte $N \times N$ - Matrix (s_{jk}) wird als Ähnlichkeitsmatrix der Objektmenge $S = \{O_1, \ldots, O_N\}$ bezeichnet.

Analog läßt sich die Unähnlichkeit bzw. Distanz zwischen zwei Objekten mittels der reelen Zahl d_{jk} angeben. Sie hat $\forall j,k : 1 \leq j,k \leq N$ die Eigenschaften

$$\begin{aligned} d_{kj} &= d_{jk} \quad \text{(Symmetrie)}, && (3.8)\\ d_{jk} &\geq 0 && (3.9)\\ \text{und} \quad d_{jj} &= 0. && (3.10) \end{aligned}$$

Die Matrix d_{jk} heißt dann Distanzmatrix[3]. Weiterhin ist es sinnvoll, die Bedingung

$$d_{jk} = 0 \Rightarrow d_{ij} = d_{ik} \quad \forall i,j,k \tag{3.11}$$

aufzunehmen[4]. Diese bedeutet, daß wenn zwei Objekte den Abstand 0 besitzen, sie zu allen anderen Objekten den gleichen Abstand besitzen. Im Gegensatz zu anderen Bereichen der Mathematik muß ein Distanzmaß hier nicht immer metrisch sein, d.h., daß die Dreiecksungleichung

$$d_{jk} \leq d_{ij} + d_{jk} \tag{3.12}$$

nicht für alle i, j und k erfüllt sein muß. Allerdings werden metrische Distanzmaße in der Praxis häufig angewendet.

Erwähnt sei hier, daß bei einigen Klassifikationsverfahren im Unterschied zu den eben gemachten Betrachtungen nicht die exakten Werte d_{jk} bzw s_{jk} erforderlich bzw. diese gar nicht bekannt sind, sondern nur die Reihenfolge der $d_{jk} : d_{j_1 k_1} \leq d_{j_2 k_2} \leq \ldots \leq d_{j_f k_f}$ (analog bei s_{jk}) benötigt wird. Solche Ähnlichkeits- oder Distanzreihenfolgen heißen induzierte Präordnungen.

[3]Im allgemeinen ist es für die praktische Umsetzung effektiver, Ähnlichkeits- und Distanzmatrizen zu verwenden (Einsparung von Neuberechnungen und Nutzung der Symmetrie).

[4](3.11) ist für die meisten Verfahren erfüllt.

3.2.1.2 Eigenschaften und Auswahlkriterien von Proximitätsmaßen

In der Praxis eingesetzt werden zahlreiche konkurrierende Ähnlichkeit- und Distanzbestimmungsmethoden, die auf empirischen Überlegungen beruhen oder der mathematischen Statistik entlehnt wurden. Dabei kann nur selten eine Methode als „ bessere“ im Vergleich zu einer anderen bezeichnet werden. Prinzipiell muß sich die Auswahl nach den Erfordernissen des praktischen Falles richten (Nutzerwissen und -anforderungen) und die reale Bedeutung der Daten beachten (Metadaten einbeziehen). Im folgenden sollen Gesichtspunkte angeführt werden, die als Groborientierung bei der Maßauswahl dienen können:

1. **Invarianzeigenschaften von Maßen**
   Invarianz eines Proximitätsmaßes bedeutet, daß sich die Distanzen bzw. Ähnlichkeiten zwischen zwei Objekten bei bestimmten Arten von Datentransformationen nicht ändern.

   Die wichtigsten Invarianzen für ratioskalierte Merkmalsvektoren sind die Skaleninvarianz und die Translationsinvarianz. Die Skaleninvarianz beinhaltet, daß die Größenordnung (Maßeinheit) der einzelnen Merkmale für die resultierende Ähnlichkeit bzw. Distanz unerheblich ist[5]. Die Translationsinvarianz eines Proximitätsmaßes hingegen bedeutet, daß bei Verschiebung der Werte eines Merkmals um einen konstanten Betrag sich die Ähnlichkeiten bzw. Distanzen ebenfalls nicht ändern. Im praktischen Fall ist es entscheidend, daß die einzelnen Merkmale der Objektvektoren (Merkmalsausprägungen) vergleichbar sind, weil ansonsten Dimensionen mit größeren Absolutwerten bei der Distanz- bzw. Ähnlichkeitsberechnung höher gewichtet werden würden. Handelt es sich um ein Maß, daß nicht skalen- und translationsinvariant ist, bedarf es einer Normierung der Daten in den einzelnen Dimensionen (s. Kapitel 3.1), um die Gleichgewichtung der Daten zu sichern.

   Bedeutung für die Auswahl von binären Merkmalen hat die Vertauschungsinvarianz. Diese beinhaltet, daß dem Vorhandensein eines Merkmals bei beiden Objekten die gleiche Gewichtung für das Proximitätsmaß zukommt wie dem Nichtvorhandensein dieses Merkmals bei beiden. Nähere Erläuterungen dazu werden in Abschnitt 3.2.1.3.1 („Beispiele für Proximitätsmaße - Binäre Merkmale“) gegeben.

   Weiterhin kann man sich natürlich auch weitere Invarianzen vorstellen, wie z.B. die Rotationsinvarianz. Allgemein ist es wichtig, daß bei jeder Transformation der Daten die durch die praktische Fragestellung gegebenen Ähnlichkeitseigenschaften nicht verändert werden und damit

[5] d.h. man kann beliebiges Merkmal mit beliebigem Faktor $\neq 0$ multiplizieren und die Ähnlichkeit bzw. Distanz bleibt gleich

die Gruppenstruktur der Objekte unverändert bleibt. Ein invariantes Maß erfaßt alle zur Klassifikation relevante, aber keine überflüssigen Informationen.

2. **Größe oder Tendenz**
   Im vorherigen Abschnitt wurden die Eigenschaften Skalen- und Translationsinvarianz vorgestellt. Diese sichern ab, daß unterschiedliche Merkmale trotz unterschiedlicher Größenordnungen bei der Berechnung des Ähnlichkeits- bzw. Distanzwertes gleich gewichtet sind, falls dies wünschenswert erscheint. Eine andere Grundfrage bei der Auswahl ist, ob für das Ergebnis der Klassifikation entweder die Größenunterschiede zwischen zwei Objekten oder die Tendenz (ähnliches Verhältnis der Komponenten Vektoren x_j und x_k) ausschlagebend sind. So könnten beispielsweise zwei Vektoren

$$x_j = \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix} \quad x_k = \begin{pmatrix} 6 \\ 7 \\ 6 \end{pmatrix} \tag{3.13}$$

   bei einem Proximitätsmaß, das Größenordnungen beachtet, sehr weit auseinanderliegen. Bei einem tendenziösen Maß sind die beiden im Gegensatz dazu sehr ähnlich.

3. **Abhängigkeit von der gesamten Objektmenge**
   Viele Maße $s_{jk} = s(x_j, x_k)$ bzw. $d_{jk} = d(x_j, x_k)$ hängen ausschließlich von den beiden Vektoren x_j und x_k ab und berücksichtigen nicht die globale Struktur der Datenmenge $\{x_1, \ldots, x_N\}$. Diese Vorgehensweise ist jedoch nicht immer angemessen. Will man z.B. die Abhängigkeit (Korrelation) der p Merkamle berücksichtigen, so bedarf es der Schätzung dieser Abhängigkeit. Diese kann (wie schon bei den Invarianzen) in einem Vorverarbeitungsschritt auf den Rohdaten durchgeführt und die Daten anschließend entsprechend manipuliert werden (s. Kapitel 3.1). Es besteht jedoch die Möglichkeit, die Schätzung direkt bei der Bestimmung von s_{jk} bzw. d_{jk} zu verwenden. Maße solcher Art hängen dann von allen N Vektoren $x_1, \ldots, x_N$ ab: $s_{jk} = s(x_j, x_k, \{x_1, \ldots, x_N\})$.

4. **Stochastische Eigenschaften der Daten**
   Als letzter Punkt sei hier angeführt, daß bei der Wahl des Proximitätsmaßes ebenfalls zu beachten ist, ob die Daten von zufälligen Meßfehlern behaftet sind, innerhalb der unbekannten Objektklassen eine natürliche Merkmalsstreuung auftritt oder die Objekte nur ein Ausschnitt aus einer größeren Grundgesamtheit sind[6]. In diesen Fällen

[6]Nicht alle Fälle/Objekte, sondern nur ein Ausschnitt, wurden erfaßt

sind Maße sinnvoll, welche die stochastischen Eigenschaften der Daten in geeigneter Weise berücksichtigen. Man sieht bei ihnen die Objektvektoren $x_k \in \mathcal{R}$ dann als Realisierung von N p-dimensionalen Zufallsvektoren an. Damit sind auch die aus ihnen berechneten Ähnlichkeiten $s_{jk} = s(x_j, x_k)$ bzw. Distanzen $d_{jk} = d(x_j, x_k)$ Zufallsgrößen, deren Wahrscheinlichkeitsverteilung von Art und Ausmaß der Gruppierung der Objekte abhängt. Das Auffinden solcher Verteilungen ist i. allg. nicht trivial.

Die unterschiedlichen Eigenschaften zeigen, welche Vielfalt an Proximitätsmaßen in der Praxis vorliegt. Oft ist es von Bedeutung, den Unterschied zwischen mehreren für die Objektmenge $\{O_1, \ldots, O_N\}$ in Betracht kommenden Maßen zu bestimmen. Als einheitliche Vergleichsbasis benutzt werden die $\binom{N}{2}$ Differenzen $d_{jk} - \delta_{jk}$. Angewandt werden der μ-Abstand

$$\Delta_\mu(d, \delta) := [\frac{1}{2} \sum_{j,k=1}^{N} |d_{jk} - \delta_{jk}|^\mu]^{\frac{1}{\mu}}, \tag{3.14}$$

und das Maß

$$\Delta_{rd}(d, \delta) := \sum_{j<k} \frac{(d_{jk} - \delta_{jk})^2}{d_{jk}^2}, \tag{3.15}$$

bei dem nur relative Differenzen betrachtet werden, und der empirischen Korrelationskoeffizient

$$\Delta_{ek}(d, \delta) := \frac{\sum_{j<k}(d_{jk} - \overline{d})(\delta_{jk} - \overline{\delta})}{\sqrt{\sum_{j<k}(d_{jk} - \overline{d})^2 \cdot \sum_{j<k}(\delta_{jk} - \overline{\delta})^2}} \tag{3.16}$$

bei dem $\overline{d}$ und $\overline{\delta}$ die Mittelwerte der entsprechenden Distanzen sind.

3.2.1.3 Beispiele für Proximitätsmaße

In diesem Abschnitt sollen exemplarisch einige wichtige Ähnlichkeits- und Distanzmaße und deren grundlegende Eigenschaften vorgestellt werden. Vorgestellt werden Proximitätsmaße auf binären, mehrstufigen und ratioskalierten Merkmalen, da diese in der Praxis am häufigsten vorkommen.

3.2.1.3.1 Binäre Merkmale

In diesem Abschnitt sollen Ähnlichkeitsmaße[7] auf binären Merkmalen vor-

[7] Distanzmaße werden hier i. allg. nicht verwendet.

gestellt werden. Ausgehend von der Kontingenztafel werden dann die Eigenschaften der Symmetrie und der Vertauschungsinvarianz für Ähnlichkeitsmaße definiert und anschließend Beispiele für invariante und nicht invariante aufgeführt. Danach wird kurz auf Maße eingegangen, welche Merkmalshäufigkeiten beachten. Abschließend soll die Grundidee von probabilistischen Ähnlichkeitsmaßen auf binären Merkmalen dargelegt werden.

Binäre Merkmale sind Merkmale mit zwei Alternativen. Die beiden Ausprägungen „0“ und „1“ (in der internen Codierung) lassen sich meistens als „Anwesenheit“ und „Abwesenheit“ des Merkmals interpretieren. Bei den folgenden Betrachtungen wird vorausgesetzt, daß alle p Merkmale $M_1, \ldots, M_p$ binär sind und damit nur nur die Werte „0“ oder „1“ in der Datenmatrix stehen.

Die Ähnlichkeit zweier Objekte O_j und O_k läßt sich häufig mit den Anzahlen übereinstimmender und nicht übereinstimmender Komponenten der Vektoren x_j und x_k bestimmen. Diese Anzahlen lassen sich in einer 2×2 - Kontingenztafel ablesen, die für jedes Paar O_j und O_k aufgestellt wird (vgl. Tabelle 3.a).

$O_j \setminus O_k$	0	1	
0	A_{jk}	B_{jk}	$p - E_j$
1	C_{jk}	D_{jk}	E_j
	$p - E_k$	E_k	p

Tabelle 3.a: 2×2 - Kontingenztafel für O_j und O_k. A_{jk} bezeichnet die Anzahl der Merkmale, bei denen beide Vektoren „0“ sind, B_{jk} die Anzahl, wie oft x_j „0“ und x_k „1“ ist u.s.w. E_j (bzw. E_k) bezeichnet die Anzahl des Wertes „1“ in O_j (bzw. O_j). p ist die Gesamtzahl binärer Merkmale.

Entsprechende Ähnlichkeitsmaße s_{jk} hängen dann nur von A_{jk}, B_{jk}, C_{jk} und D_{jk} ab. Betont werden muß, das Maße dieser Art nicht generell anwendbar sind, so daß nach praktischen Gesichtspunkten oft Modifikationen vorgenommen werden müssen. Ein Beispiel dafür ist der Vergleich zweier binärer Rasterbilder, wo es i.allg. keinen Sinn macht, reine Vergleiche der einzelnen Pixel durchzuführen. Hier wäre z.B. ein mustererkennender Algorithmus als Ähnlichkeitsgeber wesentlich geeigneter.

Als nächstes sollen die Eigenschaften der Symmetrie und der Vertauschungsinvarianz definiert werden: Ein Merkmal M_i heißt binär symmetrisch, wenn die Angabe „M_i bei O_j und O_k vorhanden“ die gleiche Information bezüglich der Ähnlichkeit beider Objekte wie die Angabe „M_i bei O_j und O_k nicht vorhanden“ trägt. Beispiel hierfür ist die Alternative „weiblich/männlich“. Wenn die Aussage $x_{ji} = x_{ki} = 1$ eine andere Aussagekraft für die Ähnlichkeit besitzt als die Aussage $x_{ji} = x_{ki} = 0$, so spricht man analog von einem unsymmetrischen Merkmal[8]. Beispiel hierfür ist die Alternati-

[8]In der Überschrift zu diesem Absatz wurde darum verzichtet, binäre Merkmale ein-

ve „rot/nicht rot“, wo Übereinstimmung wesentlich stärker auf Ähnlichkeit hindeutet als Nichtübereinstimmung mit vielfältigen Farbalternativen. Aus der soeben definierten Merkmalsymmetie läßt sich nun die Vertauschungsinvarianz (im folgenden einfach Invarianz) für Ähnlichlichkeitsmaße festlegen: Wenn das Maß s_{jk} in allen p binären Merkmalen die Alternativen „0“ und „1“ symmetrisch behandelt, so heißt es (vertauschungs-)invariant. Alle anderen Maße heißen entsprechend nicht invariant. Ein Maß s_{jk} ist insbesondere invariant, wenn es nur von den Summen $A_{jk} + D_{jk}$ und $B_{jk} + C_{jk}$ abhängt.

1. **Vertauschungs-invariante Ähnlichkeitsmaße**

   (a) **M-Koeffizient (Simple-Matching-Koeffizient)**
   „Als M-Koeffizienten bezeichnet man den relativen Anteil der übereinstimmenden Komponenten von x_j und x_k:“ (Zitat [Boc74] S. 51)

   $$s_{jk} := \frac{A_{jk} + D_{jk}}{p} = \frac{A_{jk} + D_{jk}}{(A_{jk} + D_{jk}) + (B_{jk} + C_{jk})} \tag{3.17}$$

   Es gilt $0 \leq s_{jk} \leq 1$ mit den Grenzen

   $$s_{jk} = 0 \iff x_j = \tilde{x}_k (\text{komplementäre Vektoren}) \tag{3.18}$$

   $$\text{und } s_{jk} = 1 \iff x_j = x_k. \tag{3.19}$$

   Einige Wahrscheinlichkeitstheoretische Untersuchungen zu den Eigenschaften befinden sich in [Boc74] S. 52.

   (b) **Modifikationen des M-Koeffizienten**
   Der M-Koeffizient(3.17) wichtet die übereinstimmenden und die nicht übereinstimmenden Komponenten von x_j und x_k gleich. Oft ist es jedoch sinnvoll, diese unterschiedlich zu wichten. Man ordnet dann den übereinstimmenden Komponenten das Gewicht u ($0 < u < 1$) und den nicht übereinstimmenden das Gewicht $1 - u > 0$ zu und erhält so daß Ähnlichkeitsmaß

   $$s_{jk} := \frac{u \cdot (A_{jk} + D_{jk})}{u \cdot (A_{jk} + D_{jk}) + (1 - u) \cdot (B_{jk} + C_{jk})}. \tag{3.20}$$

   Eingesetzte Spezialfälle hiervon sind

   $$\begin{aligned} u &= \frac{1}{2} && \text{M-Koeffizient} \\ u &= \frac{1}{3} && s_{jk} = \frac{A_{jk} + D_{jk}}{p + (B_{jk} + C_{jk})} \\ u &= \frac{2}{3} && s_{jk} = 1 - \frac{B_{jk} + C_{jk}}{2p - (B_{jk} + C_{jk})} \end{aligned} \tag{3.21}$$

deutig als nominal oder ordinal zu deklarieren. Unter einem gewissen Blickwinkel kann man jedoch sagen, daß symmetrische binäre Merkmale ungeordnet sind und damit als nominal und im Gegensatz dazu unsymmetrische binäre Merkmale entsprechend in zwei Wertigkeiten geordnet sind, und damit als ordinal bezeichnet werden können.

Weiterhin wurde von [Ham61] das Ähnlichkeitsmaß

$$s_{jk} := \frac{(A_{jk} + D_{jk}) - (B_{jk} + C_{jk})}{p} \tag{3.22}$$

verwendet, welches die relative Differenz über- und nicht-übereinstimmender Komponenten angibt.

2. **Nicht vertauschungs-invariante Ähnlichkeitsmaße**

   Bei unsymmetrischen Merkmalen hängt die Ähnlichkeit zweier Objekte O_j und O_k von der Unterschiedlichkeit der Anzahlen A_{jk} und D_{jk} ab. Für die folgenden Maße sei vorausgesetzt, daß die p Merkmale $M_1, \ldots, M_p$ unsymmetrisch und die „1“ immer die wichtigere der beiden Alternativen bezeichnet. Indem man nun die übereinstimmenden 1-Komponenten stärker wichtet, erhält man nicht-invariante Maße s_{jk}.

   (a) **S-Koeffizient und Modifikationen**

   Läßt man in Zähler und Nenner des M-Koeffizienten (3.17) die übereinstimmenden 0-Komponenten außer acht, so erhält man das Ähnlichkeitsmaß

$$s_{jk} := \frac{D_{jk}}{p - A_{jk}} = \frac{D_{jk}}{B_{jk} + C_{jk} + D_{jk}}. \tag{3.23}$$

   Dieses Maß heißt S-Koeffizient (oder auch Tanimoto- bzw. Jaccard-Koeffizient). Es gilt $0 \leq s_{jk} \leq 1$ mit den Grenzen

$$\begin{aligned} s_{jk} = 0 \quad &\Longleftrightarrow \quad \text{keine 1-Komponenten stimmen überein} \\ \text{und}\, s_{jk} = 1 \quad &\Longleftrightarrow \quad x_j = x_k. \end{aligned} \tag{3.24}$$

   Weiterhin benutzt wird der RR-Koeffizient

$$s_{jk} := \frac{D_{jk}}{p}, \tag{3.25}$$

   welcher die übereinstimmenden 0-Komponenten im Nenner berücksichtigt.

   Wenn man analog zum gewichteten M-Koeffizienten (3.20) den übereinstimmenden 1-Komponenten des S-Koeffizienten das Gewicht u und den anderen Komponenten das Gewicht 1-u gibt, so erhält man das Ähnlichkeitsmaß

$$s_{jk} := \frac{u \cdot D_{jk}}{u \cdot D_{jk} + (1 - u) \cdot (B_{jk} + C_{jk})}\ . \tag{3.26}$$

(b) **Verwandte Maße**

Weitere nicht invariante Maße sind

$$s_{jk} := \frac{D_{jk}}{C_{jk} + D_{jk}} \tag{3.27}$$

und

$$s_{jk} := \frac{D_{jk}}{B_{jk} + D_{jk}} \,. \tag{3.28}$$

Diese beiden Maße lassen sich als bedingte Wahrscheinlichkeit dafür deuten, daß ein zufällig gewähltes Merkmal auch O_k (bzw. O_j) vorhanden ist, wenn es bereits bei O_j (bzw. O_k) auftritt. Verallgemeinerungen dieses Ansatzes sind das arithmetische Mittel

$$s_{jk} := \frac{1}{2}(\frac{D_{jk}}{E_j} + \frac{D_{jk}}{E_k}) \tag{3.29}$$

von (3.27) und (3.28) sowie deren geometrisches Mittel

$$s_{jk} := \frac{D_{jk}}{\sqrt{E_j \cdot E_k}} \,. \tag{3.30}$$

3. **Maße mit Berücksichtigung von Merkmalshäufigkeiten und Abhängigkeiten**

Die bisher aufgezählten binären Ähnlichkeitsmaße berücksichtigen weder die Häufigkeiten der einzelnen Merkmalsalternativen noch eventuelle Abhängigkeiten zwischen den Merkmalen. Diese Ansätze wurden in den folgenden Maßen beachtet.

Will man die Merkmalshäufigkeiten der Merkmale im gesamten Datensatz beachten, so muß man zuerst die beiden Zahlen

$$N_{i1} := \sum_{k=1}^{N} x_{ki} \quad \text{und} \quad N_{i0} := \sum_{k=1}^{N} (1 - x_{ki}) = N - N_{i1} \tag{3.31}$$

berechnen, welche die Anzahlen des Auftretens von „1“ bzw. „0“ im Merkmal M_i zählen. Gewichtet man nun die Übereinstimmung zweier Einsen mit $\frac{N_{i0}}{N}$, die Übereinstimmung zweier Nullen mit $\frac{N_{i1}}{N}$ und bei Nichtübereinstimmung mit 1, so kann man die Parameter

$$\begin{aligned} D_{jk} &= \sum_{i=1}^{p} x_{ji}x_{ki} \quad \text{durch} \quad \tilde{D}_{jk} := \sum_{i=1}^{p} x_{ji}x_{ki}(\frac{N_{i0}}{N}) \\ \text{und} \quad A_{jk} &= \sum_{i=1}^{p} (1 - x_{ji})(1 - x_{ki}) \\ \text{durch} \quad \tilde{A}_{jk} &:= \sum_{i=1}^{p} (1 - x_{ji})(1 - x_{ki})(\frac{N_{i1}}{N}) \end{aligned} \tag{3.32}$$

ersetzen. Nun kann man die bereits oben vorgestellten Maße mit den modizierten Parametern benutzen und beachtet dadurch die Merkmalshäufigkeiten. Zu beachten ist, daß ein Gruppierungsverfahren, welches mit den modifizierten Parametern arbeitet, eher nach seltenen Merkmalen gruppieren wird. Objektklassen, die eher durch eine bestimmte Kombination von Merkmalsalternativen charakterisiert sind, sind damit nur schlecht erkennbar.

Um eventuelle Abhängigkeiten der Merkmale zu beachten, wird der (empirische) Korrelationskoeffizient

$$\hat{p}_{v\mu} = \frac{N \cdot \sum_{k=1}^{N} x_{kv} x_{k\mu} - N_{v1} N_{\mu 1}}{\sqrt{N_{v1} N_{\mu 1} (N - N_{v1})(N - N_{\mu 1})}} \tag{3.33}$$

der Merkmale v und μ definiert. Als Ähnlichkeitsmaß ergibt sich dann

$$s_{jk} := \sum_{v=1}^{p} \sum_{\mu=1}^{p} \hat{p}_{v\mu} \cdot t_{v\mu} \tag{3.34}$$

$$\text{mit} \quad t_{v\mu} := \begin{cases} 1, & \text{für } x_{jv} = x_{j\mu} = x_{kv} = x_{k\mu} \\ -1, & \text{für } x_{jv} = x_{j\mu} = 1 - x_{kv} = 1 - x_{k\mu} \\ 0, & \text{sonst.} \end{cases}$$

Dieses Maß ist nicht vertauschungs-invariant.

4. **Stochastische Ansätze**

Geht man davon aus, daß in der Objektmenge $\{O_1, \ldots, O_N\}$ die Merkmale eine zufällige Steuung aufweisen, so kann man die Vektoren $x_1, \ldots, x_N$ als Zufallsvektoren auffassen. Mit der Untersuchung, ob diese Vektoren voneinander unabhängig oder abhängig sind, kann man nun Ähnlichkeitsmaße definieren.

Beispiel hierfür ist der Korrelationskoeffizient

$$r_{jk} := \frac{A_{jk} D_{jk} - B_{jk} C_{jk}}{\sqrt{(p - E_k) E_k \cdot (p - E_j) E_j}} \quad \text{(speziell für binär)}, \tag{3.35}$$

der sowohl selbst als auch sein Quadrat ${r_{jk}}^2$ als Ähnlichkeitsmaß verwendet werden kann. Ein weiteres Maß dieser Art ist der Yule'sche Assoziationskoeffizient

$$s_{jk} := \frac{A_{jk} D_{jk} - B_{jk} C_{jk}}{A_{jk} D_{jk} + B_{jk} C_{jk}}, \tag{3.36}$$

der nach [Boc74] angewendet werden sollte, „wenn die Tatsache, daß die Menge der bei einem Objekt vorhandenen in der Menge der beim anderen Objekt vorkommenden Merkmale enthalten ist, bereits auf Ähnlichkeit schließen läßt.“

Problematisch bei Maßen dieser Art ist, daß ein hoher Wert s_{jk} nicht notwendigerweise eine große Ähnlichkeit von O_j und O_k bedeuten muß, sondern lediglich auf die empirische (lineare) Unabhängigkeit beschränkt ist. Deswegen wurde von [Goo64] ein Ähnlichkeitsmaß entwickelt, welches die Bestimmung der Ähnlichkeit eines Objektpaares als Zufallsexperiment beschreibt (vgl. [Boc74] S.63-64).

Zusammenfassung Tabelle 3.b faßt die wichtigsten binären Maße mit ihren Eigenschaften zusammen. Die in der Tabelle dargestellten Maße wurden beispielhaft ausgewählt, um einen Überblick für die Vorverarbeitung über die in der Literatur vorgestellten Maße zu erhalten. Prinzipiell kann man sich mit Nutzung/Nichtnutzung der Merkmalshäufigkeitsmethode mit einem der anderen Verfahren fast alle Kombinationen zusammenstellen. Kein Verfahren ist eindeutig zu bevorzugen.

Eigenschaft → Maß ↓	Invarianz	Korrelation zwischen Merkmalen	Korrelation zwischen Vektoren	Beachtung der Häufigkeiten
M-Koeffizient (3.17)	ja	nein	nein	nein
S-Koeffizient (3.23)	nein	nein	nein	nein
Merkmalshäufigkeitsmethode (3.32)	-	-	-	ja
M-Korrelationskoeffizient (3.34)	ja	ja	nein	nein
V-Korrelationskoeffizient (3.35)	ja	nein	ja	nein
Yule'scher Assoziationskoeffizient (3.36)	ja	nein	ja	nein

Tabelle 3.b: Zusammenfassung wichtiger binärer Ähnlichkeitsmaße und deren Eigenschaften

3.2.1.3.2 Mehrstufige Merkmale

Im Gegensatz zu den Ähnlichkeitsmaßen auf binären Merkmalen geht es hier vor allem um Merkmale mit mehr als zwei Zuständen. Dabei sind die nominalen mehrstufigen (z.B. Fichte, Tanne und Buche) meist von höherer praktischer Bedeutung als die ordinalen mehrstufigen Merkmale (z.B. klein - mittel - groß) und werden deswegen hier ausführlicher untersucht. Voraussetzung bei den folgenden Maßen ist, daß es sich bei den Merkmalen um

„exklusive“ Alternativen handelt[9]. Im folgenden werden einige Maße mit und ohne Beachtung der Alternativenzahl vorgestellt:

1. **Verallgemeinerter M-Koeffizienz**

   Analog zu den Betrachtungen bei binären Merkmalen kann man auch hier die Ähnlichkeit zwischen zwei Objekten O_j und O_k über die Anzahl μ_{jk} der übereinstimmenden Komponenten von x_j und x_k bestimmen. Das resultierende Ähnlichkeitsmaß

   $$s_{jk} := \frac{\mu_{jk}}{p} \tag{3.37}$$

   ist eine Verallgemeinerung des M-Koeffizienten für binäre Merkmale. s_{jk} ist vertauschungs-invariant[10] und damit nur für nominale Merkmale geeignet. Eine Modifikation ergibt sich bei unterschiedlicher Wichtung der Anzahl μ_{jk} übereinstimmender Komponenten und der Anzahl ω_{jk} nichtübereinstimmender Komponenten mit der Zahl u:

   $$s_{jk} := \frac{u \cdot \mu_{jk}}{u \cdot \mu_{jk} + (1-u)\omega_{jk}} \tag{3.38}$$

2. **Beachtung der Alternativenanzahl und der Häufigkeit**

   Die Maße (3.37) und (3.38) sind von den Anzahlen $m_1, \ldots, m_p$ der einzelnen Merkmalsalternativen unabhängig. Oft ist es jedoch sinnvoll, die Ähnlichkeit von O_j und O_k höher einzuschätzen, wenn bei Gleichheit eines Merkmals die Anzahl der vorhandenen Alternativen größer ist. Deshalb wurde folgendes Maß vorgeschlagen:

   $$s_{jk} := \frac{1}{m} \sum_{i=1}^{p} m_i \cdot \delta(x_{ji}, x_{ki}) \tag{3.39}$$

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

   Um Verzerrrungen dieses Maßes auszuschließen, sollten nur solche Alternativen zur Bestimmung der Anzahlen m_i herangezogen werden, die auch tatsächlich im Datensatz auftreten.

   Oft ist es jedoch sinnvoller, das tatsächliche Auftreten von Alternativen als Gewichtung aufzunehemen, und dieses nicht wie in (3.39) als

[9] Bei Merkmalen mit „exklusiven“ Alternativen schließen sich die Alternativen aus, wohingegen bei „nicht exklusiven“ mehrere Alternativen gleichzeitig auftreten können. Maße auf „nicht exlusiven“ Merkmalen findet man z.B. in [LW67].

[10] unabhängig von der Numerierung der Alternativen

	Merkmal 1: $\geq$ klein ?	Merkmal 2: $\geq$ mittel ?	Merkmal 3: $\geq$ groß ?
Karoline	ja	nein	nein
Robert	ja	ja	ja
Ivonne	ja	ja	nein

Tabelle 3.c: Beispiel für die Zerlegung des ordinalen Merkmals „Größe“ aus Tabelle 2.a in drei binäre Merkmale

gleichverteilt anzunehmen. Je seltener eine Alternative in der Objektmenge in einem Merkmal M_i auftritt, je stärker geht sie bei gleichzeitigem Auftreten in x_j und x_k in die Ähnlichkeit ein. Wenn also N_{iv} die Häufigkeit des Auftretens der Alternative v bei Merkmal M_i in der Menge $\{x_i, \ldots, n_N\}$ ist, kann man damit das Ähnlichkeitsmaß

$$s_{jk} := \sum_{i=1}^{p} \sum_{v=0}^{m_i-1} \frac{1}{N_{iv}} \cdot \delta(x_{ji}, v) \cdot \delta(x_{ki}, v) \tag{3.40}$$

definieren, wobei $\delta(u, v)$ wie in Formel (3.39) definiert ist.

Alternativ zu den dargestellten Maßen wird in [BEPW96] vorgeschlagen, mehrstufige Merkmale in binäre umzuwandeln und dann binäre Maße zu verwenden. Dazu wird für jede Merkmalsalternative ein neues Merkmal erzeugt, z.B. wird das Merkmal Farbe mit den Alternativen {rot, grün, gelb, blau} in 4 Merkmale mit den Alternativen „rot/nicht rot“, „grün/nicht grün“ usw. aufgespalten. Problematisch ist diese Vorgehensweise bei einer großen Anzahl von Merkmalsalternativen, weil Merkmale mit vielen Alternativen so höher gewichtet werden als Merkmale mit weniger Alternativen. Außerdem ist ein weiteres Problem dieses Ansatzes, daß künstlich Korrelationen erzeugt werden[11]. Sinnvoller ist diese Vorgehensweise bei ordinalen mehrstufigen oder auch bei intervallskalierten Merkmalen. Dort kann man Schranken definieren, über oder unter welche die entsprechenden Merkmalsalternativen eingeordnet werden. Kodiert man dann die einzelnen Schranken mit binären Merkmalen, kann die Zuordnung einer Alternative zu einem binären Merkmal durch die Frage, ob es unter oder über der Schranke liegt, erfolgen. Als Beispiel wurde das Merkmal „Größe“ aus Tabelle 2.a gewählt und die Aufspaltung in Tabelle 3.c dargestellt.

Eine weiterer Ansatz zur Ähnlichkeitsbestimmung von mehrstufigen Merkmalen ist die Verwendung von probabilistischen Ähnlichkeitsmaßen. Dieser Ansatz hat den Vorteil, daß er sowohl für nominale als auch für ordinale Merkamle verwendet werden kann (siehe [Boc74] S.70).

Zusammenfassung Tabelle 3.d faßt die wichtigsten mehrstufigen Maße und ihre Eigenschaften für die Nutzung in der Vorverarbeitung noch einmal

[11] Binäre Merkmale aus einem mehrstufigen Merkmal sind voneinander abhängig.

zusammen. Auch hier kann man nicht eindeutig festlegen, welches Merkmal besser ist als ein anderes. Je nach Anfordung kann hier jedes Maß sinnvoll einsetzbar sein. Benötigt man allerdings ein Maß auf ordinalen mehrstufigen Merkmalen, sollte man keines dieser Maße verwenden.

Eigenschaft → Maß ↓	Gleichgewichtung von $=$ und $\neq$	Beachtung der Alternativenzahl	Beachtung der Altern.-häufigkeit	Spez. Eignung für ord. Merk.
Verallgemeinerter M-Koeffizient (3.37)	ja	nein	nein	nein
Gewichteter verallg. M-Koeffizient (3.38)	nein	nein	nein	nein
Maß (3.39)	nein	ja	nein	nein
Maß (3.40)	nein	ja	ja	nein

Tabelle 3.d: Zusammenfassung mehrstufiger Ähnlichkeitsmaße und deren Eigenschaften

3.2.1.3.3 Ratioskalierte Merkmale

In diesem Abschnitt sollen Beispiele für Proximitätsmaße auf ratioskalierten Daten angegeben werden. Vorgestellt werden sollen unter anderem die Euklidische Distanz, die L_R-Distanzen, die Mahalanobis-Distanz und der Korrelationskoeffizient. Hauptaugenmerk dabei liegt auf der Herausarbeitung der wichtigsten Eigenschaften wie z.B. Invarianzen, um deren flexible Einbindung in die Pipeline zu gewährleisten. Voraussetzung hier ist, daß alle p Merkmale ratioskaliert oder ratioskalar interpretierbar sind. Ansonsten muß eine Trennung der Merkmale in unterschiedliche Merkmalstypen erfolgen, bevor die Berechnung der Distanzen bzw. Ähnlichkeiten beginnt. Diese werden dann getrennt berechnet (s. Abschnitt 3.2.1.4).

1. **Euklidischer Abstand**

   Als erstes Maß für ratioskalierte Proximitätsmaße sei hier der Euklidische Abstand

   $$d_{jk} := \parallel x_k - x_j \parallel = \sqrt{\sum_{i=1}^{p}(x_{ki} - x_{ji})^2} \tag{3.41}$$

   angegeben[12].

   Der Euklidische Abstand ist translationinvariant und invariant gegenüber orthogonalen linearen Transformationen[13] der Vektoren $x_1, \ldots,$

[12] Verwendet wird auch der quadratische Euklidische Abstand ${d_{jk}}^2$.
[13] z.B. Rotation

x_N. Er ist jedoch nicht skaleninvariant, was eine vorverarbeitende Normierung zur Sicherung der Gleichgewichtung der einzelnen Merkmale erforderlich machen kann.

2. **L_R-Distanzen (Minkowski-Metriken)**

   Eine Verallgemeinerung des Euklidischen Abstandes sind die L_R-Distanzen

$$d_{jk}{}^{(r)} := \sqrt[r]{\sum_{i=1}^{p} \mid x_{ki} - x_{ji} \mid^r}. \tag{3.42}$$

   Auch verwendet wurde deren Normierung $\Delta_{jk}{}^{(r)}$:

$$\Delta_{jk}{}^{(r)} := \frac{d_{jk}{}^{(r)}}{\sqrt[r]{p}} \tag{3.43}$$

   Praktische Relevanz von (3.42) und (3.43) haben vor allem die Spezialfälle r=1 (Summe der relativen Differenzen) und r=2 (Euklidischer Abstand 3.41)). L_R-Distanzen sind metrisch und außer bei r=2 weder translations- noch skaleninvariant noch invariant gegenüber orthogonalen Transformationen, was eine Normierung erforderlich macht. Diese Normierung erfolgt hier auf theoretischen Überlegungen beruhend auf dem Mittelwert 0 und auf dem r-ten, zentralen, absoluten Moment 1:

$$\tilde{x}_{ki} := k\frac{x_{ki} - \overline{x}_i}{\hat{\sigma}^{(r)}}, \quad i = 1, \ldots, p \quad \wedge \quad k = 1, \ldots, N \tag{3.44}$$

$$\text{mit} \quad \overline{x}_i := \frac{1}{N}\sum_{k=1}^{N} x_{ki}, \quad i = 1, \ldots, p$$

$$\text{und} \quad \hat{\sigma}^{(r)} := \sqrt[r]{\frac{1}{N} \cdot \sum_{i=1}^{N} \mid x_{ki} - \overline{x}_i \mid^r}, \quad i = 1, \ldots, p$$

   Je größer r gewählt wird, um so stärker gehen große Merkmalsunterschiede im Vergleich zu kleinen Merkmalsunterschieden in den Distanzwert ein.

3. **MAHALANOBIS-Distanz**

   Eine andere Verallgemeinerung der Euklidischen Distanz durch Verschärfung der Invarianzforderungen ist die MAHALANOBIS-Distanz

$$d_{jk}{}^2 := (x_j - x_k)^T \, \hat{\Sigma}^{-1} \, (x_j - x_k). \tag{3.45}$$

Dabei ist $(x_j - x_k)$ der Differenzvektor zwischen x_j und x_k, $(x_j - x_k)^T$ dessen transponierter (waagerechter) Vektor und $\hat{\Sigma}^{-1}$ die inverse Matrix der empirischen $p \times p$ - Kovarianzmatrix

$$\hat{\Sigma} := \frac{1}{N} \sum_{k=1}^{N} (x_k - \overline{x})(x_k - \overline{x})^T \tag{3.46}$$

zu den Vektoren $x_1, \ldots, x_N$. Hier ist $\overline{x}$ der Mittelvektor der Vektoren $x_1, \ldots, x_N$, und er wird berechnet mit

$$\overline{x} := \frac{1}{N} \sum_{k=1}^{N} x_k. \tag{3.47}$$

Unter der Voraussetzung, daß $x_1, \ldots, x_N$ voneinander unabhängig sind, eliminiert dieses Maß Korrelationen zwischen den Merkmalen. Es ist auf eine näherungsweise Normalverteilung der Ausgangsvektoren zugeschnitten und hängt nicht nur von den beiden Objekten O_j und O_k ab, sondern von der gesamten Objektmenge. Es ist skalen- und translationsinvariant und die Daten bedürfen damit keiner Normierung. Man kann die MAHALANOBIS-Distanz als euklidische Distanz zweier transformierter Vektoren y_i und y_k auffassen.

4. **Korrelationskoeffizient**

   Weitere Ähnlichkeitsmaße für ratioskalierte Merkmale sind der Korrelationskoeffizient

$$r_{jk} := \frac{\hat{\tau}_{jk}}{\sqrt{\hat{\tau}_{jj} \cdot \hat{\tau}_{kk}}} \tag{3.48}$$

   oder dessen Quadrat

$$s_{jk} := {r_{kj}}^2. \tag{3.49}$$

   Hierbei sind

$$\hat{\tau}_{jk} := \frac{1}{p} \sum_{i=1}^{p} (x_{ji} - \overline{x}_j)(x_{ki} - \overline{x}_k)$$

   die Elemente der $N \times N$ - Kovarianzmatrix $\hat{\Upsilon} = (\hat{\tau}_{jk})$ und

$$\overline{x}_k := \frac{1}{p} \sum_{i=1}^{p} x_{ki}$$

   der Mittelwert des Merkmals M_k. Der Korrelationskoeffizient ist weder skalen- noch translationsinvariant. $s_{jk} = {r_{jk}}^2 = 0$ bedeutet, daß die

beiden Vektoren linear unkorreliert (unabhängig) sind. Bei $r_{jk} = -1$ und bei $r_{jk} = 1$ sind sie maximal linear korreliert. Um davon allerdings auf allgemeine Unabhängigkeit schließen zu können, müßten die Merkmale paarweise normalverteilt sein. Sinnvoll ist seine Anwendung, wenn es vor allem um gleiche Verläufe geht und die spezielle Ausprägung nicht im Vordergrund steht.

5. **Weitere Maße**

   Weiter angewendete empirische Distanzmaße sind die Maße

$$d_{jk} := \frac{\sum_{i=1}^{p} |k_{ki} - k_{ji}|}{\sum_{i=1}^{p} (k_{ki} + k_{ji})} \tag{3.50}$$

   und

$$d_{jk} := \sum_{i=1}^{p} \frac{|k_{ki} - k_{ji}|}{|k_{ki}| + |k_{ji}|}. \tag{3.51}$$

   Verwendung fand auch das normierte Skalarprodukt

$$s_{jk} := \frac{x_j \, x_k}{|x_j| \cdot |x_k|} \tag{3.52}$$

   von x_j und x_j[14].

Zusammenfassung Tabelle 3.e faßt die wichtigsten Maße auf ratioskalierten Merkmalen aus Sicht der Vorverarbeitung noch einmal zusammen. Auffällig dabei ist, daß die MAHALANOBIS-Distanz fast alle Eigenschaften besitzt. Damit ist sie als Proximitätsmaß für viele Fälle geeignet. Probleme könnten allerdings durch die höhere Rechendauer (Invertierung einer Matrix) auftreten. Außerdem ist auf das Zutreffen von wahrscheinlichkeitstheoretischen Voraussetzungen zu achten.

3.2.1.4 Vorgehensweisen bei hybriden Merkmalen

Häufig sind unter den p Mermalen $M_1, \ldots, M_p$ Merkmale verschiedenen Typs. So können beispielsweise gleichzeitig ratioskalierte und binäre Merkmale auftreten. In solchen Fällen gibt es die folgenden Vorgehensweisen:

[14] Interessant ist, daß das Skalarprodukt als Ähnlichkeitsmaß in der zusammenfassenden Literatur teilweise keine Erwähnung findet.

[15] Außer bei r=2.

Eigenschaft → Maß ↓	skalen-invariant	translations-invariant	Korrelation zw. Merkmalen	Korrelation zw. Vektoren
Euklidischer Abstand (3.41)	nein	ja	nein	nein
L_R-Distanzen (3.42)	nein	nein[15]	nein	nein
MAHALANOBIS-Distanz (3.45)	ja	ja	ja	nein
Korrelations-koeffizient (3.48)	nein	nein	nein	ja

Tabelle 3.e: Zusammenfassung wichtiger ratioskalierter Ähnlichkeitsmaße und deren Eigenschaften

1. Es wird eines der Distanzmaße für ratioskalierte Merkmale benutzt und die Merkmale anderer Skalentypen auf die Ratioskala abgebildet. Sinnvoll ist diese Methode jedoch nur bei ordinalen mehrstufigen, selten bei binären Merkmalen.

2. Beim Auftreten von ratioskalierten und nominalen mehrstufigen Merkmalen wählt man zwei feste Zahlen $\triangle_2 \geq \triangle_1 > 0$ und setzt:

$$u_{jk} := \left\{ \begin{array}{c} \text{Anzahl der über-} \\ \text{einstimmenden} \\ \text{nominalen Komponen-} \\ \text{ten von } x_j \text{ und } x_k \end{array} \right\} + \left\{ \begin{array}{c} \text{Anzahl der ratio-} \\ \text{skalierten Kompo-} \\ \text{nenten } x_{ji}, x_{ki} \text{ mit} \\ |x_{ji} - x_{ki}| \leq \triangle_1 \end{array} \right\}$$

$$v_{jk} := \left\{ \begin{array}{c} \text{Anzahl der nicht} \\ \text{übereinstimmenden} \\ \text{nominalen Komponen-} \\ \text{ten von } x_j \text{ und } x_k \end{array} \right\} + \left\{ \begin{array}{c} \text{Anzahl der ratio-} \\ \text{skalierten Kompo-} \\ \text{nenten } x_{ji}, x_{ki} \text{ mit} \\ |x_{ji} - x_{ki}| \geq \triangle_2 \end{array} \right\} \tag{3.53}$$

   Analog zum verallgemeinerten M-Koeffizienten (3.37) und zum gewichteten M-Koeffizienten (3.38) mit $u = \frac{1}{3}$ wird die Ähnlichkeit dann durch $s_{jk} := u_{jk}/p$ oder $s_{jk} := u_{jk}/v_{jk}$ festgelegt. Dabei legt die Schranke $\triangle_1$ fest, wie sehr sich die Objekte O_j und O_k in den ratioskalierten Merkmalen unterscheiden dürfen, um noch als ähnlich zu gelten. Umgekehrt legt $\triangle_2$ fest, ab wann sie als unähnlich gelten. Diese Vorgehensweise setzt eine gleichmäßige Normierung der ratioskalierten Merkmale voraus.

3. Bei gleichzeitigem Auftreten von ratioskalierten und ordinalen mehrstufigen Merkmalen kann man die ratioskalierten durch Diskretisierung in ordinale mehrstufige Merkmale überführen und die entsprechenden

Maße verwenden. Problem dabei ist der mit der Einführung von Intervallen verbundene Informationsverlust.

4. Berechnet man für die Merkmale gleichen Skalentyps getrennte Ähnlichkeiten[16], so kann man die Gesamtähnlichkeit s_{jk} der Objekte O_j und O_k als gewichteten Mittelwert[17] der Einzelähnlichkeiten bestimmen.

Weiterer Ansatz bei gemischten Merkmalen ist z.B. die Verwendung von probabilistischen Ähnlichkeitsmaßen.

Am flexibelsten für die Pipeline scheint mir Vorgehensweise 4 zu sein, weil sie am allgemeinsten unterschiedliche Typen miteinander kombinieren kann. Die Wahl des umzusetzenden Verfahrens zur Verarbeitung hybrider Merkmalse fiel deswegen auf sie (s. Kapitel 4).

3.2.1.5 Anmerkungen zu Proximitätsmaßen zwischen Objekten

In diesem Abschnitt soll kurz auf das Problem von fehlenden Daten und auf die Möglichkeit der Wichtung von Merkmalen eingegangen werden:

1. **Fehlende Daten** Wenn in der Objekt-Merkmals-Matrix (x_{ki}) einige Daten nicht bekannt sind, muß das Proximitätsmaß modifiziert werden. Man geht dann von einem Maß s_{jk} bzw. d_{jk} aus, bei dem jeder einzelne Merkmalsvergleich einen additiven Beitrag zum Gesamtwert liefert. In der entsprechenden Summe berücksichtigt man dann nur die Komponenten, die in beiden Vektoren bekannt sind, und dividiert dann durch die Anzahl dieser Komponenten. Als Beispiel sei der modifizierter M-Koeffizient

$$s_{jk}{}^{\star} := \frac{1}{\tilde{p}} \cdot \left\{ \begin{array}{c} \text{Anzahl der} \\ \text{übereinstimmenden} \\ \text{Komponenten von } x_j \text{ und } x_k \end{array} \right\} \tag{3.54}$$

   angegeben.

2. **Wichtung der Merkmale durch Proximitätsmaße** „Bei der Berechnung von Distanzen und Ähnlichkeiten können die p Merkmale $M_1, \ldots, M_p$ verschieden gewichtet werden, indem man für jedes Merkmal M_i ein Gewicht $\omega_i > 0$ einführt und den (additiven) Beitrag jedes Merkmals M_i zu d_{jk} bzw. s_{jk} mit dem Faktor ω_i multipliziert.“ ([Boc74] S.76) So würde beispielsweise der Euklidische Abstand (3.41) das Distanzmaß

$$d_{jk}{}^{\star} := \sqrt{\sum_{i=1}^{p} \omega_i \cdot (x_{ki} - x_{ji})^2} \tag{3.55}$$

[16] Aufspaltung in mehrere Vektoren

[17] Wichtung der Einzelähnlichkeiten mit der Anzahl der Merkmale pro Merkmalstyp

ergeben. Üblich ist, die Gewichte auf $\omega_1 + \ldots + \omega_p = 1$ zu normieren.

Allgemein kann man nicht sagen, ob eine Gewichtung (z.B. Gleichgewichtung) und falls ja, welche sinnvoll ist. Allerdings ist es so, daß „die Gefahr besteht, daß durch Wahl passender Gewichte die Gruppierungsverfahren im Hinblick auf ein erwünschtes Klassifikationsergebnis manipuliert werden." ([Boc74] S.76) Anders verhält es sich allerdings, wenn die Gewichte ω_i in Äbhängigkeit von der gesamten Objektmenge automatisch berechnet werden.

3.2.2 Proximitätsmaße zwischen Objektmengen

Für viele Klassifikationsverfahren ist die Ähnlichkeit bzw. Distanz zwischen Objektmengen[18] Grundlage der Klassifikation. Deshalb sollen hier beispielhaft einige vorgestellt werden. Häufige Vorgehensweisen sind die Berechnung mit Ähnlichkeits- oder Distanzmatrizen (Nutzung der Maße aus Kapitel 3.2.1), die Berechnung über Nutzung der Mittelwerte, die Berechnung durch Vergleich der Verteilungsfunktionen und spezielle Verfahren für nominale und ordinale Daten (vgl. [Boc74] S. 81 ff).

Als erstes benötigt man eine Definition für Proximitätsmaße auf Objektmengen: Die Ähnlichkeit zweier disjunkter Objektmengen $A_i, A_j \subseteq S$ wird durch die reelle Zahl $S_{A_iA_j}$ ($S_{A_iA_j} = S_{A_jA_i} \geq 0$) und die Distanz (Unähnlichkeit) durch die Zahl $D_{A_iA_j}$ ($D_{A_iA_j} = D_{A_jA_i} \geq 0$) gemessen. Diese haben analoge Eigenschaften wie s_{ij} in (3.4) und wie d_{ij} in (3.8). Weiterhin von Bedeutung für die im folgenden dargestellten Maße ist die Zahl $n_i := |A_i|$, welche die Anzahl der Objekte in der Menge A_i beinhaltet.

Als Beispiel sollen hier Maße unter Nutzung von Ähnlichkeits- bzw. Distanzmatrizen vorgestellt werden:

Ist für die Objekte eine Ähnlichkeitsmatrix (s_{jk}) vorgegeben, werden zwei Objektmengen als ähnlich bezeichnet wenn die Objekte $O_k \in A_i$ und die Objekte $O_l \in A_j$ alle oder die überwiegende Anzahl eine große Ähnlichkeit s_{jk} aufweisen. Beispiel hierfür sind:

$$S_{A_iA_j} := \min_{\forall k \in A_i, l \in A_j} \{s_{kl}\} \tag{3.56}$$

$$S_{A_iA_j} := \frac{1}{n_i \cdot n_j} \cdot \sum_{k \in A_i} \sum_{l \in A_j} s_{kl} \tag{3.57}$$

$$S_{A_iA_j} := \max_{\forall k \in A_i, l \in A_j} \{s_{kl}\} \tag{3.58}$$

Hierbei ist die Definition (3.56), welche die Ähnlichkeit des unähnlichsten Objektpaares wiederspiegelt, eine ziemlich starke Forderung. Diese wird in (3.57) (mittlere Ähnlichkeit) abgeschwächt. (3.58) ist dagegen eine sehr

[18] bzw. Klassen oder Gruppen

schwache Forderung und z.B. gegenüber zufälligen Fehlern in der Matrix empfindlich.

Analog kann man die Distanz $D_{A_iA_j}$ unter Verwendung der Distanzen d_{jk} berechnen. Das Maximum ist dann die starke und das Minimum die schwache Forderung.

3.2.3 Heterogenitäts- und Homogenitätsmaße

Bei der Spezifikation der Problemstellung der Klassifikation (Kapitel 2) wurde bereits erwähnt, welchen grundsätzlichen Stellenwert die Homogenität und Heterogenität von Objektmengen für den Klassifikationsbegriff haben. Um diesbezüglich quantitative Aussagen machen zu können, führt man für jede Objektmenge $A \subseteq S$ eine Zahl $k(A) \geq 0$ ein, die als Homogenität bezeichnet wird. Diese gibt an, wie ähnlich sich die einzelnen Objekte im Mittel sind bzw. wie gut sie zusammenpassen. Umgekehrt kann man $g(A) \geq 0$ als Heterogenität (Inhomogenität) definieren. Je kleiner g(A), je homogener ist die Menge A. Für die im folgenden vorgestellten Maße sei $n := |A|$ die Anzahl der Objekte in A. Grundsätzliche Herangehensweisen zur Berechnung von Heterogenitäten bzw. Homogenitäten sind die Nutzung der Distanz- bzw. Ähnlichkeitsmatrix, die Nutzung des Mittelpunktes und informationstheoretische Vorgehensweisen (vgl. [Boc74] S.91 ff).

Als Beispiel sollen hier die Berechnung der Heterogenitäten bzw. Homogenitäten unter Nutzung der Distanz- bzw. Ähnlichkeitsmatrix vorgestellt werden: Ein brauchbares Maß für die Heterogenität ist die mittlere Distanz der $\binom{n}{2}$ Objektpaare aus A, also

$$g(A) := \frac{1}{n(n-1)} \sum_{k \in A} \sum_{j \in A} d_{jk}. \tag{3.59}$$

Weiterhin benutzt wurde daß Maß

$$g(A) := d(A) := \max_{\forall j,k \in A} \{d_{jk}\}. \tag{3.60}$$

Analog kann mann auch die Homogenität $k(A)$ berechnen, indem d_{jk} durch s_{jk} und max durch min ersetzt wird.

3.2.4 Bestimmung von typischen Objekten

Die Fragestellung dieses Abschnitts ist, ob sich in einer n Objekte enthaltenden Menge $A \subseteq S$ typische (repräsentative) und untypische Objekte charakterisieren lassen. Diese Bestimmung ist Grundlage einiger Gruppierungsverfahren und spielt vor allem eine wichtige Rolle bei der Interpretation und praktischen Verwertung[19] gefundener homogener Klassen. Betrachtet wer-

[19] z.B. Darstellung einer gefundenen Klasse durch ein repräsentatives Element in einem Visualisierungssystem

den hier die Bestimmung von zentralen Punkten und die Bestimmung von Kernpunkten.

1. **Zentrale Punkte** Oft läßt sich eine Objektmenge einfach durch einen zentralen Punkt, z.B. das arithmetische Mittel der Einzelvektoren bestimmen. Ist jedoch gewünscht, daß es sich um ein Objekt aus der Menge handelt, so wird das Objekt mit dem geringsten Abstand zum Mittelpunkt ausgewählt. Möchte man die Berechnungsgeschwindigkeit erhöhen oder liegen nicht ratioskalierte Mermale vor, kann man eine der folgenden Vorschriften zur Bestimmung eines zentralen Punktes t bei Nutzung der Ähnlichkeits- bzw. Distanzmatrix verwenden:

$$t_1(A_k) := \sum_{j \in A} d_{jk} \rightarrow \min_{k \in A} \tag{3.61}$$

oder

$$t_2(A_k) := \sum_{j \in A} s_{jk} \rightarrow \max_{k \in A}. \tag{3.62}$$

2. **Kernpunkte** Die unter 1 bestimmten zentralen Punkte bzw. Objekte sind nur repräsentativ, wenn die Objektmenge eine runde oder ovale Form aufweist. Ist die Form jedoch gekrümmt oder verzweigt, müssen die zentralen Punkte nicht repräsentativ sein und können sogar weit außerhalb der Menge liegen.

   In solch einem Fall wird man solch ein Objekt als typisch ansehen, das in irgendeinem Sinn die größte Punktkonzentration („Kern“) besitzt. Ein Beispiel für die Bestimmung solcher Punkte ist im folgenenden angegeben:

   Man wähle ein geeignete Distanzschranke $d^* > 0$ und suche das Objekt, für das gilt:

   $$t_3(A_k) := \{\text{Anzahl der Objekte } j \in A \text{ mit } d_{jk} \leq d^*\} \rightarrow \max_{k \in A}. \tag{3.63}$$

   bzw. analog mit $0 < s^* < 1$

   $$t_3(A_k) := \{\text{Anzahl der Objekte } j \in A \text{ mit } s_{jk} \geq d^*\} \rightarrow \max_{k \in A}. \tag{3.64}$$

   Weitere Möglichkeiten sind die Definitionen mit Hilfe des Entropieheterogenitätsmaßes und mittels Hauptkomponenten (vgl. [Boc74] S.102-103).

3.3 Klassifikationstechniken

Die Auswahl einer Klassifikationsstechnik und Durchführung der Klassifikation entsprechen dem 3. Schritt der Vorverarbeitungspipeline. Sie bilden den Kern des Klassifikationsprozesses. In diesem Abschnitt wird ein Überblick über Eigenschaften und Auswahlkriterien von Klassifikationsstechniken gegeben und anhand von Beispielen einige wichtige Verfahren vorgestellt. Am Anfang soll jedoch zuerst eine Einordnung der Klassifikation in das mathematische Umfeld erfolgen.

3.3.1 Einordnung der Klassifikation in das mathematische Umfeld

Zunächst sollen einige grundsätzliche Bemerkungen zur Klassifikation gemacht werden. Historisch wird die Klassifikation zur Statistik gezählt. Es werden aber auch Erkenntnisse aus anderen Bereichen genutzt, wie z.B. aus der Graphentheorie oder aus der künstlichen Intelligenz. Allgemein faßt man unter Klassifikation all die Verfahren, in denen unbekannte Klassen entdeckt und die Objekte in diese Klassen einsortiert werden. Damit ist sie eine Weiterentwicklung der Diskriminanzanalyse. Dort wird ein vorgegebenes Objekt in bekannte bzw. vorgegebene Klassen einsortiert. Aufgabe der Diskriminanzanalyse kann in diesem Umfeld jedoch sein, die mittels Klassifikation erhaltene Klassenstruktur auf ihre Eigenschaften zu überprüfen und die Ergebnisse zu interpretieren. Von besonderem Interesse auch für die Visualisierung sind die Clusterung bzw. automatsche Klassifikation, in der als Teilgebiet der Klassifikation keine Vorabinformation über Lage und Anzahl der Klassen vorliegt.

3.3.2 Einteilungen, Auswahleigenschaften und Überblick

In diesem Kapitel wird ein Überblick über Clusterungs- und Klassifikationstechniken erstellt. Ziel ist es, die Techniken nach allgemeinen Kriterien zu klassifizieren und für ihre Auswahl relevante Eigenschaften herauszuarbeiten. Weiterhin sollen die wichtigsten Herangehensweisen kurz vorgestellt werden.

Präzisierung des Klassifikationsproblems Das Klassifikationsproblem, welches bereits in Kapitel 2 vorgestellt wurde, soll hier noch einmal präzisiert werden: Zu N Objekten $O_1, \ldots, O_N$ sei eine Datenmatrix (x_{ki}) bekannt. Daraus wird für viele Verfahren eine Ähnlichkeitsmatrix (s_{jk}) oder eine Distanzmatrix (d_{jk}) abgeleitet, die die Ähnlichkeitsstruktur der Objektmenge $S = O_1, \ldots, O_N$ charakterisiert. Andere Verfahren arbeiten direkt auf den Daten oder alternativ lediglich auf der Ordung der Distanzen bzw. Ähnlichkeiten. Weitere Eingabemaße wie Homo- und Heterogenitätsmaße finden ebenfalls Verwendung. Gesucht ist nun aufbauend auf den Ma-

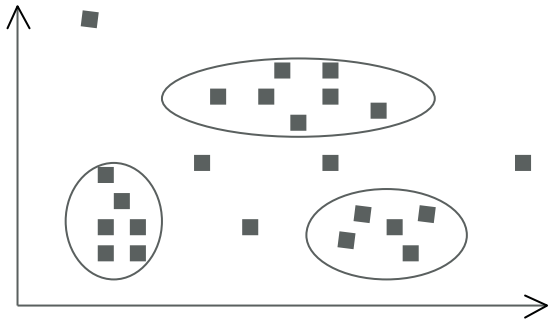

Abbildung 3.A: Disjunkte, nicht exhaustive Klassifikation im $\mathcal{R}^2$

ßen eine Klassifikation $\mathcal{A} = (A_1, A_2, \dots)$, welche die Ähnlichkeitsstruktur der Objekte möglichst gut wiedergibt und eine hinreichende Datenreduktion durchführt. Diese Forderung ist verbal so interpretierbar, daß die Objekte in jeder Klasse A_i möglichst große Ähnlichkeit besitzen sollen (Homogenität) und daß verschiedene Klassen leicht unterscheidbar sind (Separierbarkeit).

Einteilungen nach der Zielstruktur Grundsätzlich gibt es drei wichtige, strukturell unterschiedliche Klassifikationsarten:

- *Disjunkte Klassifikation:* Die Klassen $A_1, A_2, \dots$ von $\mathcal{A}$ dürfen sich nicht überschneiden.
- *Nichtdisjunkte Klassifikation:* Die Klassen $A_1, A_2, \dots$ von $\mathcal{A}$ dürfen sich (beliebig oder begrenzt) überschneiden.
- *Hierarchische Klassifikation:* Die Klassen $A_1, A_2, \dots$ von $\mathcal{A}$ sind sich in Form eines Baumes über- bzw. untergeordnet.

Ausgewählt werden sollte die Art, welche der vermuteten Struktur am meisten entspricht. Handelt es sich beispielsweise um die Klassifikation von Tieren unterschiedlicher Rasse, so ist sicherlich die hierarchische Klassifikation zu bevorzugen, weil sie den zugrundeliegenden Sachverhalt der Evolution am besten wiedergeben kann. Bei relativ homogener Objektverteilung ist die Wahl einer nichtdisjunkten Gruppierung oft überlegenswert. Manchmal kann die Zielklassenstruktur auch von einer möglichen Weiterverarbeitung der Daten bestimmt werden.

Weiterhin unterscheidet man *exhaustive* und *nicht exhaustive* Klassifikationen: Eine *exhaustive* Klassifikation ordnet alle Objekte in die Klassen $A_1, A_2, \dots$ ein. Bei der *nicht exhaustiven* Klassifikation gibt es dagegen auch unklassifizierte Objekte(s. Abb 3.A). Die Anwendung der *nicht exhaustiven* Klassifikation ist sinnvoll, wenn Verzerrungen der Klassen durch Ausreißer und zwischen den Klassen liegende Objekte vermieden werden sollen.

Schwierigkeitsgrade bei der Klassifikation Grundsätzlich unterscheidet man drei wichtige Typen von Fragestellungen mit wachsendem

Schwierigkeitsgrad:

1. Im einfachsten Fall wird die „wahre“ Klassenzahl m als bekannt vorausgesetzt und man fordert, daß $\mathcal{A}$ exhaustiv ist. Alternativ können in verschiedenen Verfahren auch Homogenitäts- bzw. Heterogenitätsschranken innerhalb der Klassen oder Distanzschranken zwischen den Klassen vorgegeben sein.

2. Bei vielen Anwendungen ist die Klassenzahl m nicht von vornherein bekannt. Das Verfahren bestimmt dann sowohl die exhaustive Klassifikation als auch die Klassenzahl.

3. Im allgemeinsten Fall sollen m Klassen $A_1, \ldots, A_m$ gebildet werden, wobei m unbekannt und die Gruppierung nicht exhaustiv ist.

Bei dieser Einteilung ist Punkt 1 für den interaktiven Visualisierungsnutzer geeignet, der durch Parametervariation selbst Modifikationen des Klassifikationsergebnisses vornehmen will. Dies ist vor allem sinnvoll, wenn der Nutzer schon Kenntnis über die Strukturierung der Objekte hat. Ist dies nicht der Fall, sollten eher Verfahren nach Punkt 2 oder Punkt 3 verwendet werden, welche die Klassenzahl selbst bestimmen.

Einteilung nach datenabhängigem Modell Die Einteilungen der letzten Absätze basierten auf der Zielstruktur der Klassifikation. Will man im Gegensatz dazu eine Einteilung aufgrund der Dateneigenschaften vornehmen, so unterteilt man in *stochastische* und *deterministische* Modelle. In einem *stochastischen* Modell werden die Zahlen x_{ki} als Realisierung von Zufallsgrößen angesehen. Diese Modellart sollte gewählt werden, wenn die Daten Meßfehlern oder einer natürlichen Streuung unterworfen sind bzw. eine Zufallsstichprobe aus einer größeren Grundgesamtheit darstellen. Stochastische Modelle führen zu einer statistischen Behandlung des Klassifikationsproblems und erlauben - zumindest prinzipiell - zu testen, ob die Objektmenge S überhaupt eine Klassenstruktur aufweist. *Deterministische* Modelle wählt man, wenn S eine feste Objektmenge ist, in der die Eigenschaften zwar von Objekt zu Objekt variieren, jedoch keinen Zufallsschwankungen unterworfen sind. Wahrscheinlichkeitsaussagen sind bei solchen Daten offenbar sinnlos.

Stabilität Ein weiteres wichtiges Kriterium zur Bewertung und Interpretation der Ergebnisse ist die Stabilität einer Klassifikation. Wichtig hierbei ist die Frage, wie stark sich die Klassenstruktur bei Änderung von Parametern, beim Herausnehmen von einzelnen Objekten oder der Veränderung des zugrundeliegenden Maßes ändert.

Klassenform - Klassentrennung Oft von Interesse sowohl für Interpretation der Ergebnisse als auch für die geeignete Verfahrenswahl ist die Frage, welche Klassenformen durch das benutzte Verfahren erkannt werden

können bzw. welchen Formannahmen sie zugrundeliegen. Hier gibt es verschiedene Formen:

- Viele Verfahren gehen von einer runden Klassenform aus.
- Andere Techniken führen zu Kettenbildung, was bedeutet, daß sich auch langgestreckte Klassen herausbilden können.
- Weiterhin kann die Klassentrennung durch sogennte Hyperebenen erfolgen.
- Ein anderer Ansatz ist die Betrachtung der Punktdichte, um Klassen zu identifizieren. Dort sind je nach verwendeter Verteilungsdichtefunktion verschiedene Formen möglich.

Im folgenden sollen nun einige Herangehensweisen vorgestellt werden. Es sei darauf hingewiesen, daß über die hier vorgestellten Verfahren hinaus in neuerer Zeit weitere Ansätze enwickelt wurden, Daten zu strukturieren und zu klassifizieren. Beispiel hierfür sind u.a. die Lernverfahren mit neuronalen Netzen. Eine Vorstellung dieser Techniken würde allerdings den Rahmen dieser Arbeit sprengen.

3.3.2.1 Disjunkte Verfahren

In diesem Abschniitt sollen die verschiedene Arten von disjunkten Klassifikationen vorgestellt werden. Es handelt sich dabei um Optimale Gruppierungen, Numerische Verfahren, um die Analyse der Punkt- und Verteilungsdichte und um graphentheoretische Methoden.

1. **Optimale Gruppierungen**

   Optimale Gruppierungen erzeugen disjunktive Klassifikationen. Diese erfolgen meist in Abhängigkeit von den den Objekten O_j zugehörigen Vektoren x_j oder auch auf der Distanz- bzw. Ähnlichkeitsmatrix basierend[20]. Grundlage optimaler Gruppierungen ist die Einführung eines Gütekriteriums $k(\mathcal{A})$ bzw. $g(\mathcal{A})$ für jede mögliche Partition $\mathcal{A}^* = (A_1^*, \ldots, A_m^*)$ von S. Ziel ist dann, diejenige Partition zu finden, für welche die Funktion $k(\mathcal{A})$ maximal bzw. die Funktion $g(\mathcal{A})$ minimal ist. Unterschiedliche Kriterien wichten die folgenden drei Anliegen unterschiedlich:

   - Wie groß ist die Ähnlichkeit der Objekte *innerhalb* der einzelnen Klassen A_i von $\mathcal{A}$? (Homogenität von $\mathcal{A}$)

[20] Bei Maßen mit Verwendung der D/Ä-Matrizen besteht die Möglichkeit, nichtratioskalierte Merkmale einzubeziehen

- Wie klein ist die Ähnlichkeit von Objekten *verschiedener* Klassen und wie gut sind die Klassen voneinander getrennt? (Separation von $\mathcal{A}$)
- Fallen „ähnliche" Objekte tatsächlich in gleiche und „unähnliche" Objekte aber in verschiedene Klassen?

Hat man nun das Kriterium bestimmt, kann man aus theoretischer Sicht jede bezüglich des Kriteriums optimale Partition als Lösung der Klassifikationsproblems betrachten. Praktisch ist dies jedoch problematisch, weil man i. allg. für *alle* Kombinationen von möglichen Partitionen den Kriteriumswert bestimmen muß („Abzählverfahren"). Weil ihre Anzahl jedoch exponentiell mit der Zahl der Objekte wächst (s. [Boc74] S.109 ff.), ist diese Vorgehensweise meist ineffektiv. Deswegen wurden Verfahren entwickelt, die im Rahmen akzeptablen Rechenaufwandes approximative Lösungen nahe dem globalen Optimum liefern.

Im folgenden sollen Beispiele für Optimalitätskriterienkriterien angegeben werden:

- Möglichkeiten zur Bestimmung von Gütekriterien sind z.B. entscheidungstheoretische Modelle auf der Basis von Normalverteilungen. Dort wird jede gesuchte Objektklasse durch eine p-dimensionale Normalverteilung charakterisiert. Bestimmmt werden die Parameter für die der einzelnen Verteilungen und Anzahl m der Klassen durch Maximum-Likelihood-Schätzer oder Bayesverfahren. Dabei repräsentieren die Eigenwerte der Normalverteilungen die Klassenschwerpunkte und ihre Kovarianzmatritzen die räumliche Größe der Klassen. Es werden dabei runde Klassenformen zugrundegelegt.
- Ein weiteres Kriterium für ratioskalierte Daten ist das Varianzkriterium

$$\begin{aligned} g(\mathcal{A}) &:= g(\mathcal{A}, x_1, \ldots, x_N) \\ &:= \sum_{i=1}^{m} \sum_{k \in A_i} \| x_k - \bar{x}_{A_i} \|^2 \rightarrow \min . \end{aligned} \tag{3.65}$$

  Hierbei ist $\bar{x}_{A_i}$ der Mittelpunkt der Vektoren x_{A_i} der Klasse A_i. Das Varianzkriterium fordert, daß die Homogenität (ausgedrückt durch die Varianz) der Klassen im Mittel minimal ist. Es setzt voraus, daß die Merkmale unabhängig, die Klassen kugelförmig und die Aufteilung der Objekte gleichmäßig[21] ist. Aus dem Varianzkriterium für die disjunktive Klassifikation leitet sich auch

[21] bzw. bestimmten Forderungen unterworfen ([Boc74]) S. 163)

das Ward-Verfahren der hierarchischen agglomerativen Klassifikation ab, indem das Kriterium auf die Partionierung nur zweier Klassen eingrenzt wird. Das Ward-Verfahren führt eine Approximation des optimalen Problems durch, um mit Hilfe einer Gruppenhierarchie eine disjunkte Klassifikation zu konstruieren.

- Als Verallgemeinerung des Varianzkriteriums wurden Optimalitätskriterien entwickelt, welche die Korrelation der Merkmale untereinander beachten. Bei dieser Betrachtungsweise wird beachtet, daß die Art der Abhängigkeit meist unbekannt ist und möglicherweise von Klasse zu Klasse variiert.
- Während bei den eben genannten Kriterien die Klassen A_i durch die Mittelpunkte der zugehörigen Vektoren repräsentiert sind, können auch Kriterien auf Hyperebenen basierend definiert werden. Ziel ist, eine Klasse statt durch einem einzigen Punkt durch eine möglichst niedrigdimensionale Hyperebenen zu repräsentieren.
- Für nominal mehrstufige Merkmale ist ebenfalls ein Kriterium entwickelt worden. Grundidee hierbei ist, daß jedes Merkmal M_t eine Partition $\mathcal{A}^t = (A_1^t, A_2^t, \dots)$ der Objekte definiert. Will man nun die Partition $\mathcal{A}$ auf ihre Güte hin untersuchen, bestimmt man die Übereinstimmung von $\mathcal{A}$ und den $\mathcal{A}^t$ im Mittel.

2. **Numerische Verfahren** Wie im vorhergehenden Abschnitt angedeutet, ist die Bestimmung der optimalen Partition wegen des großen Aufwands nicht generell möglich. Außerdem ist in der Praxis die vorgestellte Interpretation des Gruppierungsproblems zu eng. Dort können zahlreiche, oft auch konkurrierende Forderungen auftreten. Weiterhin ist die Beschränkung auf exhaustive Klassifikation oft unzweckmäßig. Deswegen wurden verschiedene numerische Verfahren entwickelt, welche die optimale Partition approximieren und in diesen Punkten flexibler sind. Die Verfahren empfehlen sich wegen ihrer hohen Flexibilität (Variation von Parametern, verschiedene Proximitätsmaße) und ihrer schnellen Berechenbarkeit.

   Im folgenden sollen diese Techniken kurz vorgestellt werden:

   (a) **Iterative Verbesserung der Anfangsklassifikation** Verfahren dieser Art versuchen, bei vorgegebenem Optimalitätskriterium $g(\mathcal{A})$ (bzw. $k(\mathcal{A})$) approximative Lösungen zu finden. Eine vorgegebene Anfangsklassifikation $\mathcal{A}^0$ wird durch systematische Umgruppierung der Objekte iterativ solange verbessert, bis ein stabiler Zustand eintritt. So gelangt man zu einem lokalen Minimum der Funktion $g(\mathcal{A})$, i. allg. jedoch nicht zum globalen Minimum. Im Austauschverfahren werden durch Verlagerung von

Objekten zwischen den einzelnen Klassen immer neue Partitionen $\mathcal{A}^v$ gebildet, wodurch $g(\mathcal{A}^v)$ verkleinert wird. Problematisch hierbei ist die günstige Wahl der Anfangsklassen und die richtige Wahl der Anfangsklassenzahl. Vorgehensweise hier kann z.B. sein, mit einer geringen Klassenzahl zu beginnen und bei beendeter Verringerung von $g(\mathcal{A})$ zu prüfen, ob eine Spaltung von Klassen sinnvoll ist und diese Spaltung gegebenenfalls durchzuführen.

(b) **Rekursiver Aufbau von Gruppen um Kerne** Dieses Verfahren kann sowohl auf den Beobachtungsvektoren $x_1, \ldots, x_k$ als auch auf der Distanz- bzw. Ähnlichkeitsmatrix basieren. Es bestimmt die noch unbekannte Anzahl der Klassen und kann alternativ disjunktiv exhaustiv und nicht exhaustiv gruppieren. Idee ist daß sich die Elemente jeder Klasse dicht um ein „typisches" oder „zentrales" Objekt scharen und die Klassen um diese Objekte aufgebaut werden. Verfahren dieser Art benutzen das folgende Konstruktionsprinzip:

i. Man suche in der Menge $S = \{O_1, \ldots, O_N\}$ das „typischste" Objekt O_{k_1}, betrachte es als Kern einer ersten Gruppe A_1 und setze zunächst $A1 = \{O_{k_1}\}$.

ii. Man suche aus der Menge $U_1 := S - A_1$ jenes Objekt O_k, daß die größte Ähnlichkeit zu A_1 besitzt und füge es zu A_1 hinzu.

iii. Wiederholung von Punkt 2. bis bis Heterogenität der Menge $A_1 + \{O_{k_1}\}$ eine Grenze übersteigt. A_1 ist nun eine fertige Klasse.

iv. Man entfernt nun die Klasse A_1 aus der Objektmenge S und sucht in $S - A_1$ einen zweiten Kernpunkt O_{k_2} und eine dazu gehörige Klasse A_2 usw. Durch Iteration dieser Schritte entsteht eine Folge disjunkter Klassen $A_1, A_2, A_3, \ldots$

v. Das Verfahren wird abgebrochen, sobald alle Objekte aus S klassifiziert sind (exhaustive Gruppierung) oder wenn nur noch unwesentliche Gruppen auftreten (nicht exhaustive Gruppierung).

Eingesetzt wurden verschiedene Maße (s. Abschnitt 3.2). Die gefundenen Gruppierungen sollten noch mit Hilfe anderer Methoden verbessert werden. Insbesondere eignen sie sich für Bestimmung der Anfangsklassifikation von optimalen Verfahren.

(c) **Heuristische und kombinierte Verfahren** Wie oben bereits angedeutet, können die Bedingungen an die Klassifikation komplexerer Natur sein. Z.B. könnten minimale oder maximale Klassengrößen vorgegeben und Mindestabstände zwischen den Klassen definiert werden müssen. Solche mglw. wiedersprechenden

Anforderungen können nur approximativ gelöst werden. Meist werden dazu Verfahren kombiniert oder „elementare“ Verfahren mittels heuristischer Gesichtspunkte modifiziert.

3. **Analyse der Punkt- und Verteilungsdichte** Grundlegender Ansatz bei der Analyse von Punkt- und Verteilungsdichten ist die anschauliche Vorstellung , daß eine Punktgruppe A einem zusammenhängenden Bereich des Raumes entspricht, in dem die einzelnen Punkte überdurchschnittlich dicht liegen und sie durch Bereiche niedrigerer Punktdichte abgetrennt sind. Die spezielle Gestalt der Menge A spielt keine Rolle. Die Stelle maximaler Punktkonzentration wird als Zentrum oder Kern der entsprechenden Gruppe angesehen. Gruppiert werden die Objekte $O_1, \ldots, O_N$ aufgrund der N Beobachtungsvektoren $x_1, \ldots, x_N \in \mathcal{R}^p$, womit Verfahren dieser Art nur auf ratioskalierte Merkmale anwendbar sind.

   (a) **Verteilungsmischungen** Annahme hierbei ist, daß die Objekte $O_1, \ldots, O_N$ eine zufällige Stichprobe aus einer größeren Grundgesamtheit darstellen. Die Gesamtpopulation $\prod$ weise weiterhin wirklich eine Klassenstruktur auf und bestehe aus den Einzelpopulationen $\prod_1, \ldots, \prod_m$ derart, daß jede Population $\prod_i$ durch eine eigene Verteilungsdichte $f_i(x)$ charakterisiert ist ($x \in \mathcal{R}^p$). Die Wahrscheinlichkeit dafür, daß ein zufällig aus S herausgegriffenes Objekt zur Population $\prod_i$ gehört, sei p_i. Die marginale Dichte des Zufallsvektors X ergibt sich dann mit

$$f(x) = p_1 f_1(x) + p_2 f_2(x) + \ldots + p_m f_m(x). \tag{3.66}$$

   Diese Mischung der Dichten $f_1(x), \ldots, f_m(x)$ wird als Verteilungsmischung bezeichnet. f(x) beschreibt die Verteilung eines Merkmalsvektors X aus $\prod$, der zu einem zufällig gezogenen Objekt O gehört.

   Für das Gruppierungsproblem sind nun vor allem folgende Schritte von Bedeutung:

   - Bestimmung der Verteilungsmischung $f(x)$ z.B. durch Approximation der $f_i(x)$ mit Normalverteilungen und Bestimmung der zugehörigen Parameter.
   - Bestimmung der Population des Objektes O_k. Die Lösung dieser Frage kann i. allg. im Rahmen der Diskriminanzanalyse erfolgen.

   (b) **Nichtparametrische Schätzung der Verteilungsdichte** Im vorherigen Paragraphen reduzierte sich die Bestimmung von f(x) auf die Schätzung unbekannter Parameter. Parametrische Schätzungen sind jedoch nicht immer möglich und häufig auch unzweckmäßig. Bei Techniken dieser Art erfolgt die Schätzung auf nicht

parametrische Art. Eine Techniken hier sind die Verwendung von Kernfunktionen, Reihenentwicklungen und die Verwendung von Ranggrößen (s. [Boc74] S.265 ff).

(c) **Gruppierung unter Verwendung der Punktdichte** In Verfahren dieser Art hat jeder Punkt mit einer ihn überlagernden Funktion $f_i(x)$ einen Einfluß auf die Gesamtdichte f(x). Die Punkte x_i werden dann z.B. nach einem Gradientenverfahren in Richtung des nächsten Maximums der aktuellen Verteilungdichte verschoben. Diese Vorgehensweise erzeugt immer eindeutigere Maxima der Verteilungsdichte und gruppiert gleichzeitig die Punkte in diesen Maxima.

Will man dann z.B. eine nicht exhaustive Gruppierung erzeugen, kann man mittels einer Dichteschranke s alle die Maxima, für die f(x) unter s liegt als unklassifizierte Objekte aussortieren.

(d) **Sequentielle, selbst-adaptierende Verfahren** In der Praxis kann man nicht immer voraussetzen, daß alle Daten gleichzeitig vorliegen. Verfahren dieser Art sind in der Lage, mit einem kontinuierlichen Datenstrom der Objekte O_j umzugehen und neue Objekte in die Klassifikation einzubeziehen.

4. **Graphentheoretische Methoden** Methoden dieser Art basieren auf einer $N \times N$-Distanzmatrix (d_{jk}) und erzeugen auf dieser Basis Gruppierungen, die generell die Homogenität oder Separation der Objektmengen sichern. Grundlage ist die Definition von Objektmängen $A \subseteq S$ als „Gruppen“. Ist ein solcher Gruppenbegriff definiert, reduziert sich das Klassifikationsproblem auf die eindeutig formulierte Aufgabe, die in S enthaltenen Gruppen zu bestimmen.

   Dafür wählt man eine beliebige, aber feste Distanzschranke $d > 0$ und bezeichnet zwei Objekte O_j und O_k als „ähnlich“ genau dann, wenn $d_{jk} \leq d$ gilt. Die Menge aller Objekte O_j mit $d_{jk} \leq d$ heißt *d-Umgebung* von O_k. Für jede Gruppe A fordert man dann:

   (a) A soll nicht leer sein.

   (b) Mit jedem Objekt O_k soll auch die gesamte d-Umgebung von O_k zu A gehören.

   (c) Keine in A enthaltene Teilmenge $B(\neq A)$ soll a und b erfüllen. (Minimalitätsbedingung)

   Die Konstruktion der Gruppen von S erfolgt nun, indem man mit einem Objekt beginnt, alle Objekte seiner d-Umgebung zur Gruppe hinzufügt und mit diesen Objekten fortsetzt. Ist die Gruppe vollständig, fährt man mit mglw. verbleibenden Objekten fort, für diese ebenfalls

Gruppen zu bilden. Derart erhält man einen Graphen, dessen isolierte Teilgraphen die einzelnen Gruppen bilden. Die dadurch entstandene Partition $\mathcal{A}$ bezeichnet man als Gruppierung der Stufe d. Diese Methode wird als Gruppierung der Zusammenhangskomponenten oder auch als Single-Linkage-Methode bezeichnet. Vorteil dieser Methode ist die Steuerbarkeit durch den Parameter d. Sie neigt zur Kettenbildung und identifiziert Ausreißer.

Durch Konstruktion des Minimalbaumes aus dem vollständig verbundenen Graphen kann eine Hierarchie erzeugt werden (vgl. Single-Linkage als hierarchisches Verfahren). Diese Konstruktion erfolgt, indem man zu einem Anfangsobjekt O die minimale Distanz zum nächsten Objekt bestimmt und dieses in den Baum einfügt. Bestimmt man dann fortschreitend immer die kürzeste Distanz eines Baumknotenobjekts mit einem noch nicht eingefügten Objekt und fügt dieses ein, so entsteht der Minimalbaum.

Weitere Ansätze zur axiomatischen Gruppierung mit graphentheoretischen Verfahren sind z.B. die Gruppierung durch reziproke Paare und durch k-Gruppen.

3.3.2.2 Nichtdisjunkte Klassifikation

Für viele praktische Anwendungen ist die Beschränkung auf disjunktive Klassifikationen unzweckmäßig. Wenn die gesuchte Gruppierung graduelle Abstufungen mit fliessendem Übergang widerspiegeln soll, ist die Verwendung von nichtdisjunktiven Verfahren sinnvoller. Voraussetzung der im folgenden beschrieben Techniken ist eine Distanz- bzw. Ähnlichkeitsmatrix. Die Vorgehensweisen sind hierbei weitgehend axiomatisch in Anlehnung an die graphentheoretischen Verfahren der disjunkten Klassifikation. Es handelt sich dabei um die Definition von Gruppen oder Cliquen. Aufgrund dieser Definition erfolgt dann die Konstruktion, die meist mathematisch exakt definiert ist. Die wichtigsten Verfahren seien hier kurz angeführt:

1. **Maximale Cliquen** Bei maximalen Cliquen wird von der Forderung ausgegangen, daß die Distanz jedes Objektes einer Gruppe zu ausnahmslos allen anderen Objekten dieser Gruppe eine vorgeschriebene maximale Distanz d nicht überschreiten soll: Eine Menge A von Objekten aus S heißt *Clique der Stufe d* ganau dann, wenn die Ungleichung

$$d_{ij} \leq d \quad \forall\, O_i, O_j \in A \tag{3.67}$$

erfüllt ist. Weiterhin heißt eine Clique *maximal*, wenn zu A kein Objekt O_k aus der Restmenge $S - A$ hinzugefügt werden kann, ohne daß Bedingung (3.67) verletzt würde.

Mit diesen Definitionen läßt sich nun relativ leicht eine Konstruktionsvorschrift bestimmen, die alle maximalen Cliquen erzeugt. Weil Gruppen der Größe eins oder zwei ebenfalls erfaßt werden, empfiehlt es sich häufig, diese zu eliminieren.

Allgemein kann man sagen, daß maximale Cliquen Homogenität und Vollständigkeit der entsprechenden Objektmengen sichern, jedoch die Separation zu anderen Gruppen nicht einbeziehen. Dies kann oft zu sehr ähnlichen Gruppen führen, was keine sinnvolle Informationsreduktion darstellt. Außerdem ist der Durchmesser d einer Klasse oft eine zu harte Einschränkung. Ist der „natürliche“ Klassendurchmesser größer als d oder variieren die Klassendurchmesser, sind maximale Cliquen kaum sinnvoll einsetzbar. Außerdem sind sie mit ihrem strengen Kriterium für zufällige Fehler in der Matrix anfällig. Um diese Mängel zu beseitigen, wurde eine Vielzahl von Verfahren vorgeschlagen (s. [Boc74] S.330).

Allgmein ist zu empfehlen, das Verfahren mit mehreren d-Werten auszuführen. Damit besteht die Möglichkeit, eine Hierarchie mit disjunkten Klassen aufzubauen.

2. **R-Gruppen** Eine weitere Vorgehensweise ist die Definition und Konstruktion von R-Gruppen. Ziel dabei ist, die Inflexibilität von Cliquen zu überwinden. Das geschieht, indem man bei der Berechnung von Homogenitäten statt der Extremwerte mittlere Ähnlichkeiten benutzt. So kann ein Objekt zu einer Gruppe gehören, wenn es zu einem repräsentativen Teil der Objekte der Gruppe ähnlich ist, was der anschaulichen Vorstellung von „natürlichen“ Objektklassen entspricht.

   Eine Objektmenge $A \subseteq S$ heißt *R-Gruppe* genau dann, wenn für alle Objekte $O_i \in A$

   $$\sum_{j \in A} s_{ij} - \sum_{j \in S-A} s_{ij} \geq 0 \tag{3.68}$$

   gilt. Ziel ist nun, möglichst kleine R-Gruppen zu erzeugen, deren innere Ähnlichkeit größer als die äußeren Ähnlichkeiten sind.

   Es kann relativ leicht ein Algorithmus bestimmt werden, der R-Gruppen zur Objektmenge S erzeugt ([Boc74] S. 341 ff). Um möglichst minimale R-Gruppen zu erzeugen, werden die so erhaltenen R-Gruppen - falls möglich - in kleinere R-Gruppen aufgespaltet.

   Problem dieses Ansatzes ist, daß auch große Objektmengen typischerweise R-Gruppen sind und die kleinen nur bei extremer Homogenität und Separation R-Gruppen bilden. Deswegen ist dieser Ansatz ungeeignet, wenn die Objektmenge kleine überschneidende Objektklassen enthält oder die Erzeugung großer Klassen von vornherein als un-

zweckmäßig erscheint. Deswegen wurden z.B. die S-Gruppen verwendet, deren Kriterium

$$\frac{1}{|A|-1}\sum_{j\in A} s_{ij} \geq \frac{1}{N-|A|}\sum_{j\notin A} s_{ij} \tag{3.69}$$

schwächer ist als das der R-Gruppen. Die S-Gruppen erlauben die Gruppenbildung kleiner Gruppen, falls die Gruppen bereits oberhalb des Gesamtähnlichkeitsdurchschnitts der Objektmenge liegen.

Eine weitere Verbesserung der R-Gruppen stellen die GR-Gruppen dar (s. [Boc74] S.349 ff). Zusätzlich zur Homogenität innerhalb der Gruppe wird dort die Heterogenität zu Nichtgruppenmitgliedern gefordert.

3.3.2.3 Hierarchische Klassifikation

Problem und Vorgehensweise Grundlegendes Problem der meisten bisher vorgestellten Verfahren ist, daß dort als Kriterium entweder die Klassenzahl m bzw. die Homogenität k der Klassen (z.B. in Form des Klassendurchmessers) vorgegeben werden mußte. Problematisch dabei ist jedoch, daß diese Parameter meist schwer mit den „tatsächlichen“ Größen der Gruppierung in Übereinstimmung zu bringen sind. Außerdem ist es für die Visualisierung (s. Kapitel 1) meist sinnvoll, in mehreren Stufen durch die entstandene Struktur navigieren zu können.

Vorgehensweise bei der hierarchischen Klassifikation ist deswegen, keine einzige disjunkte Gruppierung zu bestimmen, sondern eine ganze Folge solcher Gruppierungen mit steigender Anforderung an die Homogenität (und steigender Klassenzahl) zu erzeugen (sogennte Partitionen der Stufe h). Hauptforderung dabei ist, daß die Gruppen vergleichbar sein sollen und daß sich nach dem Prinzip der schrittweisen Verfeinerung größere Klassen in feinere Unterklassen aufteilen.

Die Partitionenhierarchie - Repräsentation durch ein Dendrogamm Das Ergebnis solch einer Klassifikation läßt sich graphisch als sogenanntes *Dendrogramm* darstellen (Abb. 3.B), in dem die einzelnen Klassen nach Art eines „Stammbaums“ angeordnet sind. Jede Klasse in einem Dendrogramm wird durch einen Knoten dieses Baumes repräsentiert. Eine Klasse enthält die Objekte, die durch Fusion der Objekte der zugehörigen Sohnknoten entstehen.

Ein Dendrogramm heißt *indiziert*, wenn an den Baumknoten die Heterogenität bzw. die Homogenität der Klasse eingetragen wird, die dem Knoten entspricht. Ein indiziertes Dendrogramm bestimmt eindeutig eine Partitionenhierarchie $\bar{\mathcal{H}}$ von Partitionen $\mathcal{A}^v$ auf den Heterogenitätsstufen h.

Die Heterogenität h einer Klasse steigt, je näher sie sich am Wurzelknoten des Dendrogramms befindet. Analog steigt die Homogenität, je kleiner

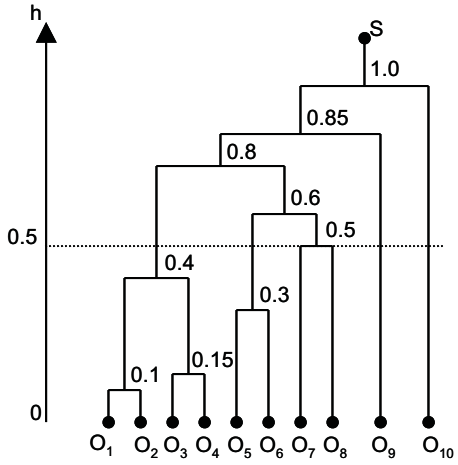

Abbildung 3.B: Indiziertes Dendrogramm mit 10 Objekten und Darstellung der Heterogenitäten der einzelnen Klassen an den Knoten

die Klassen werden. Um diese Eigenschaft abzusichern, sollten ausschließlich monotone Heterogenitäts- bzw. Homogenitätsmaße verwendet werden.

Ziel der hierarchischen Klassifikation ist, ein indiziertes Dendrogramm zu erzeugen, welches die Ähnlichkeitsstruktur der Objekte am besten wiedergibt. Das entspricht der Forderung, daß die Stufe h der Hierarchie, auf welcher sich zwei Objekte O_j und O_k befinden, umso niedriger sein soll, je größer die Ähnlichkeit zwischen ihnen ist.

Interpretationsmöglichkeiten von Hierarchien Bei der Benutzung von Hierarchien ist es möglich, rein visuell die Heterogenität bzw. Homogenität von Klassen in der Hierarchie zu vergleichen, die nicht ineinander enthalten sind. Ein weiterer Vorteil solch einer Hierarchie ist z.B., daß man leicht sehen kann, ob die Klassen eher durch Fusion von Klassen vergleichbarer Größenordnung (starke Gruppenstruktur) oder eher durch Einfügen einzelner Objekte entstanden sind (schwache Gruppenstruktur). Eine Klasse, die über einen weiten Heterogenitätsbereich fast unverändert bleibt, kann als deutlich ausgeprägt, gut separiert und damit als „natürliche“ Klasse bezeichnet werden. Ausreißer sind daran zu erkennen, daß sie erst in einer relativ hohen Heterogenitätsstufe einsortiert wurden.

Anwendungsbereiche Wichtige Anwendungsbereiche sind solche, wo bereits aufgrund sachlicher, außermathematischer Gesichtspunkte eine hierarchische Struktur der Objektmenge zu vermuten ist. Hierarchische Klassifikationen sind aber auch anwendbar, wenn eine disjunkte Klassifikation[22]

[22]In der Visualisierung sind disjunkte Klassifikationen auf meheren Stufen sinnvoll einsetzbar. Die Darstellung von Dendrogrammen, in denen in einer Ebene meist nur eine Fusionierung stattfindet, ist meist zu unübersichtlich und führt im Sinne der Visualisie-

auf bestimmten Heterogenitätsstufen gesucht ist.

Verfahren Im folgenden sollen unterschiedliche Verfahren zur Bestimmnung von Hierarchien vorgestellt werden:

1. **Optimale Hierarchien** Bei optimalen Hierarchien soll ein indiziertes Dendrogramm mittels einer Optimalitätsforderung erzeugt werden.

   Erste Möglichkeit ist die Erzeugung des Dendrogramms aus der Distanzmatrix durch Nutzung von Ultrametriken. Eine Ultrametrik ist ein Distanzmaß mit bestimmten Eigenschaften (vgl. [Boc74] S. 265 ff). Dabei existiert eine eindeutige Abbildung von einer Ultrametrik auf ein indiziertes Dendrogramm. Aufgabe bei der Bildung von optimalen Hierarchien mit Ultrametriken ist, für das gegebene Distanzmaß d der Objektemenge S eine Ultrametrik zu finden, welche d approximiert (vgl. [Boc74] S.371 ff).

   Als zweite, weniger theoretische Möglichkeit zur Definition eines Optimalitätskriteriums ergibt sich die Forderung, den Heterogenitätszuwachs von einer Hierarchiestufe zur nächsten möglichst gering zu halten. Agglomerative wie auch diversive Verfahren setzen eine Approximation dieser Forderung um. Von einer Heterogenitätsstufe zur nächst höheren werden dort gerade die Klassen bzw. Objekte[23] zusammengefaßt, die den geringsten Heterogenitätszuwachs erzeugen. Im allgemeinen werden dabei zwei Klassen bzw. Objekte fusioniert, es sei denn, die Fusionierung mehrerer Klassen bzw. Objekte ergibt den selben Heterogenitätswert wie die Fusionierung nur zweier Klassen bzw. Objekte.

2. **Agglomerative Verfahren**

   In der Praxis von großer Bedeutung sind die agglomerativen Verfahren. Grundidee hier ist, daß die Partitionenhierarchie $\bar{\mathcal{H}}$, beginnend mit der feinsten Partition $\mathcal{A}^0 = (\{O_1\}, \ldots, \{O_N\})$ derart aufgebaut wird, daß aus der feineren Partition $\mathcal{A}^{v-1}$ durch Fusionierung von ihr zugehörigen Klassen die gröbere Partition $\mathcal{A}^v$ erzeugt wird. Es handelt sich hier also um „buttom-up"-Verfahren.

   Konstruktionsprinzip Voraussetzung für die Konstruktion dieser Hierarchie ist ein Distanzmaß D_{AB}, welches die Distanz bzw. Unähnlichkeit zweier Klassen mißt. Analog könnte auch ein Ähnlichkeitsmaß S_{AB} verwendet werden. Im folgenden soll der zugrundeliegende Algorithmus präzisiert werden:

   (a) Für $v = 0$ sei $\mathcal{A}^0 = (\{O_1\}, \ldots, \{O_N\})$ die feinste Partition der Objektmenge $S = \{O_1, \ldots, O_N\}$.

rung eine zu geringe Reduktion der Ausgangsinformation durch.

[23] Einobjektige Klassen

(b) Zu Beginn von Schritt v sei die $(v-1)$-te Partition $\mathcal{A}^{v-1} = (A_1, \ldots, A_{m_{v-1}})$ bereits konstruiert. Unter $A_1, \ldots, A_N$ suche man 2 Klassen A_r und A_s, für die die Distanz $D_{A_r A_s}$ minimal ist:

$$D_{A_r, A_s} = \min_{\forall i \neq j} \{D_{A_i A_j}\}$$

Die neue Partition $\mathcal{A}^v = (A_{v_1}, \ldots, A_{v_{mv}})$ entsteht dann dadurch, daß die beiden Klassen A_r und A_s von P^{v-1} zu einer einzigen Klasse $A = A_r + A_s$ fusioniert werden.

(c) Der Agglomerationsschritt b wird für $v = 1, 2, \ldots$ iteriert, bis nach $N-1$ Schritten alle Objekte in der Klasse $\{O_1, \ldots, O_N\}$ vereinigt sind.

Je nach Distanzmaß D_{AB} bzw. S_{AB} unterscheidet man verschiedene agglomerative Verfahren. Wichtig dabei ist, daß das Maß monoton ist, da es i. allg. gleich direkt als Heterogenitäts bzw. Homogenitätsmaß für die Indizierung des Dendrogramms verwendet wird.

Man unterscheidet bei diesen Verfahren die Eigenschaften dilatierend, kontrahierend und konservativ. Ein Verfahren ist dilatierend, wenn es dazu neigt, die Objekte in kleinere, etwa gleich große Klassen zusammenzufassen. Im Gegensatz dazu erzeugen kontrahierende Verfahren eher wenige große Klassen, die vielen kleinen gegenüberstehen. Kontrahierende Verfahren sind zur Ausreißeridentifikation geeignet. Konservative Verfahren weisen weder die eine noch die andere Tendenz verstärkt auf.

Die aus unterschiedlichen Maßen resultierenden Verfahren und ihre Eigenschaften sollen im folgenden kurz vorgestellt werden:

- **Single-Linkage-Verfahren** Das Single-Linkage-Verfahren für die hierarchische Klassifikation ist eine Anwendung des Verfahrens aus Kapitel 3.3.2.1 - Abschnitt „Graphentheoretische Methoden“. Das Abstandsmaß

$$D_{A_r A_s} := \min_{\forall j \in A_r, k \in A_s} d_{jk} \tag{3.70}$$

beinhaltet die minimale Distanz zweier Objekte aus den beiden Klassen A_r und A_s. Das Verfahren fusioniert die beiden Klassen, in denen zwei Objekte jeweils aus einer Klasse sich am nächsten liegen. Das Single-Linkage-Verfahren neigt zur Kettenbildung und ist geeignet zur Erkennung von Ausreißern (kontrahierendes Verfahren). Vorteil dieses Verfahrens ist, daß man mathematisch beweisbare Aussagen über das Klassifikationsergebnis ableiten kann. Zur Vermeidung der unerwünschten Verkettungseigenschaft wurden Modifikationen des Konstruktionsprinzips und eine nicht disjunkte Hierarchie eingeführt.

- **Complete-Linkage-Verfahren** Das Abstandsmaß

$$D_{A_r A_s} := \max_{\forall j \in A_r, k \in A_s} d_{jk} \tag{3.71}$$

beinhaltet die maximale Distanz zweier Objekte aus den beiden Klassen A_r und A_s. Das Verfahren fusioniert die beiden Klassen, für die der Durchmesser der zusammengefügten neuen Klasse minimal ist. Das Complete-Linkage-Verfahren tendiert eher zur Bildung kleiner Gruppen und versucht die Objekte verstärkt in gleich große Gruppen zusammenzufügen (dilatierendes Verfahren). Ausreißer können hier das Fusionierungsergebnis verzerren und sollten vorher eleminiert werden. Vorteil dieses Verfahrens ist, daß man mathematisch beweisbare Aussagen über das Klassifikationsergebnis ableiten kann.

- **Zentroid-Verfahren** Das Abstandsmaß

$$D_{A_r A_s} := \|\bar{x}_{A_r} - \bar{x}_{A_s}\|^2 \tag{3.72}$$

beinhaltet die Distanz der Mittelpunkte von A_r und A_s. Es fusioniert die Klassen mit dem ähnlichsten Mittelpunkt. Das Zentroid-Verfahren ist konservativ.

- **Average-Linkage-Verfahren** Das Abstandsmaß

$$D_{A_r A_s} := \frac{1}{n_r n_s} \cdot \sum_{j \in A_r} \sum_{k \in A_s} d_{jk} \tag{3.73}$$

beinhaltet die mittlere Distanz aller Objekte von A_r und A_s. Es fusioniert die Klassen, deren Objektdistanzen im Mittel am kleinsten sind. Das Average-Linkage-Verfahren ist konservativ.

- **Ward-Verfahren** Das Ward-Verfahren gehört zu einer Klasse von Verfahren, die nicht auf der Distanz bzw. Ähnlichkeit der Klassen einer Partition $\mathcal{A}^v$, sondern wie in der Definition von indizierten Dendrogrammen direkt auf Heterogenitäts- bzw. Homogenitätsmaßen basieren. Ziel bei diesen Verfahren ist, den Heterogenitätszuwachs von der Partition $\mathcal{A}^{v-1}$ zur Partition $\mathcal{A}^v$ zu minimieren. Diese Forderung tritt im Konstruktionsprinzip an die Stelle der Distanzforderung. Das Verfahren von Ward im speziellen versucht, das Varianzkriterium (vgl. 3.65) zu minimieren. Dies bedeutet, daß sich die Summe der Varianzen der einzelnen Klassen sich nur minimal erhöhen soll. Das Ward-Verfahren ist konservativ und bildet etwa gleich große Klassen. Es ist damit unter den vorgestellten Maßen lt. [BEPW96] am besten geeignet, hierarchische Klassifikationen durchzuführen. Weitere Verfahren basieren z.B. auf informationstheoretischen Heterogenitätskriterien.

Für die hier aufgeführten Verfahren besteht die Möglichkeit der Beschleunigung der Berechnung der Distanzmatrix, indem diese iterativ in jedem Schritt an die neue Partition angepaßt wird (vgl. [Boc74] S. 404 und [BEPW96] S. 286 - 287). Damit entfällt beim Klassenvergleich die aufwendige Distanzberechnung zwischen *allen* Objekten beider Klassen.

Weiterhin besteht die Möglichkeit, während der hierarchischen agglomerativen Konstruktion durch Stoppen bei einer bestimmten Partition eine disjunkten Klassifikation zu erzeugen. Stopkriterien sind u.a. das Erreichen einer vorgegebenen Klassenzahl, falls der Heterogenitätszuwachs einen vergleichsweise großen Sprung ausführt (Elbow-Kriterium) oder wenn der Informationsverlust beim Partitionsübergang eine gewisse Schranke übersteigt. Diese Kriterien sind u.a. auch für die Anwendung in der Visualisierung von Interesse, weil dort auf verschiedenen Ebenen disjunkte Klassifikationen erzeugt werden (vgl. Abschnitt 4.2.1).

3. **Diversive Verfahren** Ebenso wie bei den agglomerativen Verfahren wird bei den diversiven Verfahren aufgrund einer Distanz- oder Ähnlichkeitsmatrix ein indiziertes Dendrogramm erzeugt. Der Unterschied besteht darin, daß die Verfahren beginnend mit der Klasse $\mathcal{A}^0 = S$ eine Unterteilung bereits gefundener Klassen durchführen („top-down"-Verfahren). Dabei wird eine Klasse „optimal" in zwei Unterklassen aufgeteilt. Diese werden dann rekursiv immer wieder in zwei Klassen aufgeteilt, bis die resultierenden Klassen einelementige Objekte bilden.

   Grundforderung bei der Aufspaltung einer Klasse $A \subseteq S$ in zwei Unterklassen B_1 und B_2 ist die Forderung, daß B_1 und B_2 möglichst homogen und möglichst gut getrennt sind oder daß die Heterogenität der Partition $\mathcal{A} = (B_1, B_2)$ möglichst klein ist. Die zwei Hauptvorgehensweisen sind (vgl. [Boc74] S.412 - 419):

   - **Polythetische Verfahren** Von polythetischen Merkmalen spricht man, wenn die p Merkmale in symmetrischer Weise berücksichtigt werden.
   - **Monothetische Verfahren** Im Gegensatz zu den polythetischen Methoden erfolgt bei den monothetischen Methoden die Zerlegung auf Grundlage *eines* charakteristischen Merkmals.

4. **Weitere hierarchische Methoden** Weitere Methoden zur Erzeugung von hierarchischen Klassifikationen basieren auf der Modifikation von disjunkten bzw. nichtdisjunkten Verfahren. Die Hierarchien werden dort durch Variation von Parametern der Verfahren erzeugt. Beispiele hierfür sind das Verfahren Schnell und das Verfahren von

Wishart (s. [Boc74] S.420 ff). Interessant ist auch die Erzeugung von nichtdisjunkten Hierarchien.

3.3.3 Interpretation und Validierung der gewonnenen Ergebnisse

In diesem Kapitel sollen einige Bemerkungen dazu gemacht werden, wie man die mit der Klassifikation gewonnenen Ergebnisse interpretieren kann. Wichtig ist dabei vor allem die Frage, wie stabil und gesichert man das Ergebnis ansehen kann. Auf bereits im vorigen Abschnitt 3.3.2 eingegangene Interpretationen und Validierungsansätze soll hier nicht noch einmal eingegangen werden.

Stabilität Grundsätzlich von Bedeutung für die Interpretation von Klassifikationen ist die Frage der Stabilität der entstandenen Klassifikation. Geprüft werden sollte, wie stark das Ergebnis von *leicht* veränderten Parametern, vom speziell eingesetzten Verfahren und von den zugrundeliegenden Maßen beeinflußt wird. Weiterhin ist zu beachten, welchen Einfluß die Auswahl bestimmter Merkmale und Objekte bzw. die Beachtung von Korrelationen zwischen Merkmalen und zwischen Objekten haben. Empfehlenswert kann z.B. sein, die Interpretation erst zu formulieren, wenn sie auf mehreren „Simulationsläufen“ mit leicht veränderten Parametern beruht. Dann kann eine relativ umfassende „Kernaussage“ getroffen werden.

Analyse von Klasseneigenschaften Weiterhin interessant für die Interpretation von Klassifikationsergebnissen ist die Analyse von Klassen auf den Einfluß von bestimmten Merkmalen hin. Vor allem kann man fragen, wie homogen ein Merkmal innerhalb einer Klasse vertreten ist und wie stark bei den Objekten einer Klasse bestimmte Merkmale ausgeprägt sind.

Um diese Fragen zu untersuchen, kann man beispielsweise für ratioskalierte Merkmale die F- und die t-Werte bestimmen. Der F-Wert eines Merkmals i in einer Klasse A berechnet sich nach der Vorschrift

$$F(i, A) = \frac{\sigma(i, A)}{\sigma(i)}, \tag{3.74}$$

wobei $\sigma(i, A)$ die Varianz des Merkmals i in der Gruppe A und $\sigma(i)$ die Varianz des Merkmals i in der gesamten Objektmenge bezeichnen. Je kleiner der F-Wert, desto geringer ist die Streuung dieses Merkmals im Vergleich zur gesamten Objektmenge. Wenn der F-Wert 1 überschreitet, weist das Merkmal eine größere Streuung in der Klasse als in der gesamten Objektmenge auf. „Ein Cluster ist als vollkommen homogen anzusehen, wenn alle F-Werte kleiner als 1 sind.“ (Zitat [BEPW96] S.310)

Um die Ausprägung eines Merkmals i in einer Klasse A zu bestimmen,

zieht man den t-Wert mit

$$t(i, A) = \frac{\bar{x}(i, A) - \bar{x}(i)}{s(i)} \tag{3.75}$$

zur Interpretation heran. Hierbei bezeichnet $\bar{x}(i, A)$ den Mittelwert des Merkmals i der Objekte der Klasse A, $\bar{x}(i)$ den Mittelwert des Merkmals i in der gesamten Objektmenge und $s(i)$ die Gesamtstandardabweichung des Merkmals i. Negative t-Werte eines Merkmals zeigen an, daß dieses in der Klasse unterrepräsentiert ist. Analog zeigen positive Werte an, daß ein Merkmal in der Klasse überrepräsentiert ist. Je größer der Betrag des t-Wertes ist, je stärker ist die Unter- bzw. die Überrepräsentation dieses Merkmals in dieser Klasse.

Mit dem t-Wert ist ein Hilfsmittel gegeben, welches in der Visualisierung z.B. zur automatischen Beschriftung von Klassen eingesetzt werden kann. Damit kann der Informationsgehalt und die Verständlichkeit einer visuellen Darstellung gesteigert werden.

Kapitel 4

InfoSonne - Ein Tool zur Vorverarbeitung

Nachdem im vorhergehenden Kapitel die theoretischen Grundlagen für die Vorverarbeitungspipeline gelegt wurden, soll in diesem Kapitel die Umsetzung des Tools „InfoSonne“ vorgestellt werden. Schwerpunkt dabei ist, die flexible und erweiterbare Umsetzung der Pipeline vorzustellen. Weiterhin wird die implementierte Auswahl der im vergangenen Teil behandelten Verfahren vorgestellt.

Die Implementierung erfolgte unter dem Betriebssystem „Windows NT“ und in der Programmiersprache „Visual C++ 6.0“. Verwendet wurde die „Standard Template Library“. Eine Portierung nach „Unix“ ist angedacht und grundsätzlich möglich[1].

4.1 Zugrundeliegende Datenstrukturen

In diesem Abschnitt sollen die implementierten Datenstrukturen vorgestellt werden. Dabei soll erläutert werden, wie die Datenstrukturen durch ein geeignetes Design den flexiblen Anforderungen der Pipeline entsprechen. Grundlegender Ansatz bei der Speicherung der Daten ist die Trennung von Meta-, Nutzer- und Verfahrensinformationen auf der einen Seite und den Rohdaten und den aus ihnen transformierten strukturbestimmenden Datenstrukturen auf der anderen Seite. Erst damit kann die geforderte Flexiblität erreicht werden.

Ziel bei dem Design der Datenstrukturen war, Membervariablen wenn möglich durch die Deklaration als *private* zu verstecken und den Zugriff nur über Memberfunktionen und Memberoperatoren zu erlauben. Dieses Konzept soll absichern, daß der Benutzer der geschaffenen Datenstrukturen unabhängig von der darunterliegenden Implementation eine Schnittstelle zur

[1] keine Verwendung der MFC

Verfügung gestellt bekommt, auf der er aufsetzen kann. Damit wird einerseits gesichert, daß bei Änderungen der internen Implementation sich nach außen hin nichts ändert[2] und der Nutzer auch daran gehindert wird, die interne Konsistenz der Datenmenge zu gefährden[3].

Unterabschnitt „Grunddatenstrukturen“ (4.1.1) beinhaltet die Vorstellung der Ausgangs-, Zwischen- und Ergebnisdatenstrukturen. Im darauffolgenden Unterabschnitt „Das Deskriptorkonzept“ (4.1.2) wird das Steuerungskonzept vorgestellt, welches die Transformationen der Daten aus 4.1.1 kontrolliert.

4.1.1 Grunddatenstrukturen

In diesem Abschnitt werden die Datenstrukturen vorgestellt, auf denen die Klassifikation basiert. Die Rohdaten, welche in Form einer Objekt-Merkmals-Matrix gegeben sind, werden in der Datenstruktur VectorArray gespeichert. Im Verlauf der Pipeline wird eine Ähnlichkeits- oder Distanzmatrix aufgestellt, wenn die Klassifikation auf der Berechnung von Distanzen oder Ähnlichkeiten beruht. Für diese Matrizen wurde die Datenstruktur TriangleMatrix eingeführt. Diese ist für viele Klassifikationsverfahren die Basis der Klassenberechnung. Das erzielte Ergebnis der Klassifikation wird in der Datenstruktur HierachyTree gespeichert. Im folgenden werden die einzelnen Datenstrukturen vorgestellt:

4.1.1.1 VectorArray und Vector

Den Rohdaten, welche in Form einer Objekt-Merkmals-Matrix vorliegen, wurde mit der Datenstruktur „VectorArray“ eine passende Form gegeben. Für das Design dieser Klasse wichtig war die Realisierung des flexiblen Zugriffs sowohl auf Einzelwerte der Matrix als auch auf den zu einem Objekt O_j zugehörigen Vektor x_j. Um für die Arbeit mit den erhaltenen Vektoren ein flexibles Interface zu haben, wurde zusätzlich zur Klasse VectorArray eine Klasse „Vector“ definiert, die z.B. als Übergabeparameter für die Proximitätsfunktionen verwendet wird. Der Zugriff auf die einzelnen Objektevektoren erfolgt mit der Funktion VectorArray :: getVec, der als Übergabeparameter der Index des zum Vektor gehörenden Objektes übergeben wird. Mit dem Operator VectorArray :: operator() ist der Zugriff auf einzelne Elemente der Objekt-Merkmals-Matrix möglich, indem der Objekt- und der Merkmalsindex übergeben werden.

Zusätzlich zu den Matrix-Werten werden im VectorArray ebenfalls die Namen der Objekte in der Variable VectorArray :: ObjNames und die Namen der Merkmale in der Variable VectorArray :: AttNames gespeichert.

[2] Es sollen höchstens neue Funktionen hinzukommen.
[3] z.B. Einstellung undefinierter Werte

Weiterhin implementiert wurde die Klasse Vector, die eine flexible Definition der Vektordimension zuläßt. Diese flexible Vektorlänge ist z.B. für die Aufteilung des gesamten Merkmalsvektor in Vektoren mit unterschiedlichen Skalentypen bei der Berechnung von hybriden Maßen von Bedeutung.

Bei diesen beiden Klassen wurde nicht nach unterschiedlichen Skalen getrennt. D.h., daß in ihnen alle Werte vom Typ „double“ sind. Die Unterscheidung nach den Typen wird im „DatenDeskriptor“ festgelegt (vgl Kapitel 4.1.2). Wenn ein Merkmal beispielsweise im „DatenDeskriptor“ als binär festgelegt ist, so enthält dieses Merkmal im VectorArray nur die Werte „1“ oder „0“. Analog sind für mehrstufige Merkmale nur ganzahlige Werte definiert.

Die genaue Definition dieser Klassen ist in der Datei „VectorArray.hpp“ in Anhang B.1 enthalten.

4.1.1.2 TriangleMatrix

Die Datenstruktur „TriangleMatrix“ wurde als Kontainer für Ähnlichkeits- und Distanzmatrizen konzipiert. Vorteil bei der Benutzung solcher Matrizen ist, Mehrfachberechnungen von Ähnlichkeits- und Distanzwerten zu vermeiden.

Unter Nutzung der Eigenschaft der Symmetrie von Proxymitätsmaßen kann weiterhin der Speicherbedarf der symmetrischen $N \times N$-Matrizen durch die Einsparung der unteren (bzw. oberen) Halbmatrix und der Diagonale[4] mehr als halbiert werden.

Außerdem ergab sich als Anforderung aus den Verfahren der agglomerativen hierarchischen Klassifikation (vgl. Kapitel 3.3.2.3), die Größe dieser Matrix durch Streichung von Spalten verkleinern zu können. Dies geschieht dann, wenn zwei Objekte bzw. Klassen in solch einem Verfahren fusioniert werden. Durch Neuberechnung der Distanzen der fusionierten Klassen mit entsprechenden Rekursionsformeln entfällt so die Neuberechnung der *gesamten* entstandenen Teilmatrix.

Die Nummer der Spalte bzw. Zeile entspricht dem Index des im VectorArray festgelegten Objektes. Die Löschfunktionalität wurde mit der Funktion TriangleMatrix :: deleteLineAndColumn und der Variable TriangleMatrix :: validLinesAndColumns umgesetzt. Zugriff auf die Werte d_{jk} bzw. s_{jk} erlaubt der Operator TriangleMatrix :: operator().

Die genaue Definition dieser Klasse ist in der Datei „TriangleMatrix.hpp“ in Anhang B.2 enthalten.

[4] In der Diagonale stehen nur die Distanzen bzw. Ähnlichkeiten der Objekte zu sich selber. Diese sind jedoch per Definition gleich 1 bzw. 0.

4.1.1.3 HierachyTree

Die Datenstruktur „HierachyTree“ dient als Kontainer für die Klassifikationsergebnisse der Pipeline. Wichtig war beim Design dieses Kontainers, eine Datenstruktur zu entwerfen, die alle Typen von Klassifikationen unterstützt. Sie soll sowohl disjunktive, nichtdisjunktive und auch hierarchische Klassifikationsergebnisse speichern können. Die Entscheidung fiel auf die Konzeption eines Baumes, weil dieser alle vorgegebenen Strukturen überdeckt. Jeder HierachyTree gehört eindeutig zu einem VectorArray und trägt an seinen Blättern Verweise auf die entsprechenden Objektvektoren des VektorArrays. Der Wurzelknoten entspricht der gesamten Klassifikation, zwischen Knoten einzelnen Klassen und die Blätter den Objekten. Bei der disjunkten und nichtdisjunkten Klassifikation ist ein solcher Baum auf drei Ebenen reduziert[5]. Im Fall nichtdisjunkter Klassifikation können Objekte in unterschiedlichen Klassen mehrmals eingetragen werden.

Ein HierachyTree besteht aus einer Menge von Knoten (Klasse Node), die hierarchisch geordnet sind. Zugriff erhält man über den Wurzelknoten HierachyTree :: root des Baumes. Von dort aus kann man sich entweder rekursiv oder iterativ mit der zu einem Knoten gehörenden Liste Node :: sons durch den Baum navigieren oder die einzelnen Unterobjekte eines Knotens mit dem Iterator TreeIter durchwandern.

Jeder Knoten besitzt weiterhin eine Heterogenität, was den HierachyTree als indiziertes Dendrogramm nutzbar macht.

Weiterhin dienen die Funktionen Node :: heter und HierachyTree :: buildHeterogenityHierachyFromDendrogram zur Erzeugung eines n-nären Baumes aus einem binären Dendrogramm. Außerdem besteht die Möglichkeit, Klassifikationen in das „Typ0“-Format von KOAN[6] zu exportieren.

Die genaue Definition dieser Klasse ist in der Datei „Klassifikation.hpp“ in Anhang B.3 enthalten.

4.1.2 Das Deskriptorkonzept

In diesem Abschnitt soll beschrieben werden, wie die Anforderungen an das Vorverarbeitungssystem „InfoSonne“ durch Wahl eines geeigneten Konzeptes umgesetzt wurden.

Eingangsdaten in die Pipeline sind die Rohdaten, deren Metadaten und die vom Nutzer vorgegebenen Nutzeranforderungen (s. Abb. 2.A). Wie die Rohdaten in eine Datenstruktur umgesetzt wurden, wurde bereits in Abschnitt 4.1.1.1 vorgestellt. Da es für die Flexibilität der Pipeline von großer Bedeutung ist, die Metadaten von den Rohdaten zu trennen und die Nut-

[5] 1. Ebene ist die Klassifikation, die 2. Ebene sind die Klassen und die 3. Ebene sind die Objekte

[6] KOntext ANalysator, entwickelt bei Siemens AG Abteilung ZT

zeranforderungen und die resultierenden Algorithmeninformationen nicht starr im Quelltext zu verankern, mußte ein diesen Anforderungen entsprechendes Konzept geschaffen werden. Eine weitere Anforderung an ein solches Konzept ist die leichte Erweiterbarkeit um neue Verfahren und Maße, ohne die äußere Schnittstelle zu beeinflussen. Dem entstandenen Konzept wurde der Name „Deskriptorkonzept“ gegeben, weil es sich sich hier um *Beschreibungen* von Daten- und Algorithmeninformationen handelt.

Diese Vorgaben führten zur Entwicklung von drei Deskriptorentypen. Dies sind der „NutzerDeskriptor“, welcher die Nutzeranforderungen an die Klassifikation enthält, der „DatenDeskriptor“, welcher die Eigenschaften der Merkmale und Objekte enthält und der „ProzessDeskriptor“, der die für die Daten und Anforderungen passenden Verfahren und Parameter beschreibt. Mit der Funktion Pipeline :: calcProzessDeskriptor wird der ProzessDeskriptor automatisch aus den beiden anderen berechnet. In dieser Funktion liegt die Hauptintelligenz bei der Algorithmen-, Maß- und Parameterwahl.

Weil gleichzeitig nur eine Klassifikation durchgeführt wird und fast alle Funktionen auf die Deskriptoren zurückgreifen, wurden die Deskriptoren als globale Variablen deklariert. Dies macht sie von überall frei verfügbar und verkleinert dadurch wesentlich die Übergabeparameterliste der beteiligten Funktionen und erhöht die Übersichtlichkeit des Quelltextes.

Im folgenden sollen diese drei Deskriptoren und deren grundlegende Eigenschaften kurz beschrieben werden. Die zugehörigen Quelltextdefinitionen können in der Datei „deskriptor.hpp“ (s. Anhang B.4) eingesehen werden.

4.1.2.1 Der DatenDeskriptor

Die Datenstruktur „Datendeskriptor“ enthält die Metainformationen über die Objekt-Merkmals-Matrix. Deswegen wird ihm eindeutig das VectorArray DatenDeskriptor::va zugeordnet. Er enthält einen „ObjektDeskriptor“ und einen „AttributDeskriptor“. Der ObjektDeskriptor enthält Informationen über die Objekte. Im aktuellen Stand enthält er die Information, welche Objekte in die Klassifikation einbezogen werden sollen. Der AttributDeskriptor enthält die Eigenschaften der Merkmale. Aktuell enthält er für jedes Merkmal die Variablen *skalenTyp* (Skalentyp des Merkmals), *bRelevanz* (Einbeziehung des Merkmals in Klassifikation), *Gewichtung* und *bInfoLücken* (Fehlen von Daten in diesem Merkmal). Diese werden in der Map *AttMap* gespeichert. Im Fall, daß alle Merkmale gleich sind, wird zur Vereinfachung und Geschwindigkeiterhöhung stattdessen lediglich die Variable *allEqualType* gefüllt.

4.1.2.2 Der NutzerDeskriptor

Die Datenstruktur „NutzerDeskriptor“ enthält die Zielvorgaben des Nutzers an die Klassifikation. Derzeit besteht die Möglichkeit, die Variablen *clusterArt* (Art des Klassifikationszieles: hierarchisch oder disjunkt), *bAbsoluteDistanzen* (Schwerpunkte absolute Distanzen conta Profilverläufe), *bAusreisserIdentifizieren*, *bAusreisserEleminieren*, *klassenForm* (Festlegung der Klassenform z.B. rund oder kettenförmig) und *EinzubeziehendeDimensionen* festzulegen.

4.1.2.3 Der ProzessDeskriptor

In diesem Abschnitt soll die Datenstruktur „ProzessDeskriptor“ vorgestellt werden. Er wird unter der Vorgabe der beiden anderen Deskriptoren durch die Funktion Pipeline :: calcProzessDeskriptor berechnet. Diese Funktion legt fest, welche Art von Maß der Klassifikation zugrundeliegen soll und welches Verfahren mit welchen Parametern einzusetzen ist.

Hauptteile dieses Deskriptors sind der „StandardisierungsDeskriptor“, der „AehnlichkeitsDistanzDeskriptor“ und der „GruppierungsAlgorithmus“. Der „StandardisierungsDeskriptor“ legt fest, welche Vorverarbeitungen durchgeführt werden. Der „AehnlichkeitsDistanzDeskriptor“ legt die Art und die Funktionen der zugrundeliegenden Maße fest. Der „GruppierungsAlgorithmus“ spezifiziert die Art und die Parameter des gewählten Klassifikationsverfahrens.

4.2 Umsetzung der Pipelinestruktur

In diesem Abschnitt soll die Implementierung des Pipelineablaufes kurz vorgestellt werden.

Die Funktionen für die Pipelineabarbeitung und das Einlesen und Berechnen der Deskriptoren wurden in der Klasse „Pipeline“ zusammengefaßt (vgl. Anhang B.5).

Im folgenden soll der Ablauf der Vorverarbeitung kurz beschrieben werden (vgl. Anhang B.6 - „main“-Funktion des CommandLineTools). Vor Beginn der Pipelineausführung werden zuerst die Funktionen Pipeline :: readData und Pipeline :: readDeskriptors aufgerufen, welche das Einlesen der Rohdaten, der Metadaten und der Nutzeranforderungen in die entsprechenden Datenstrukturen ausführen. Als nächstes wird mit Hilfe der Funktion Pipeline :: calcProzessDeskriptor die Auswahl der Maße und Verfahren durchgeführt und das Ergebnis der Auswahl im ProzessDeskriptor gespeichert. Aufgrund der vorliegenden Datenstrukturen erfolgt dann die Ausführung der Pipeline in der Funktion Pipeline :: executePipeline. Diese ruft dabei der Reihe nach die Funktionen Pipeline :: executePraeprozess, Pipeline ::

calculateProxyMatrix und Pipeline :: calculateKlassifikation auf, was der theoretisch vorgestellten Vorverarbeitungpipeline entspricht.

Ergebnis der Ausführung dieser Funktionenfolge ist eine Klassifikation, welche durch die Datenstruktur HierachyTree repräsentiert wird. Diese Struktur kann als Eingabe in ein Visualisierungssystem genutzt, in einen n-nären Hierarchiebaum überführt oder in eine Datei ausgegeben werden.

4.2.1 Implementierte Algorithmen und Verfahren

Dieser Abschnitt hat zum Ziel, implementierte Verfahren und Algorithmen vorzustellen.

Hauptüberlegung bei der Auswahl von zu implementierenden Verfahren war, daß diese einerseits eine hohe praktische Relevanz besitzen und andererseits aber immer noch so einfach sein sollten, daß ihre Struktur und ihr Verhalten überschaubar bleiben.

Vorverarbeitung Für die Vorverarbeitung wurden verschiedene Normierungen implementiert.

Proximitätsmaße Bei der Definition der Skalentypen und der Umsetzung von zugehörigen Maßen wurde der Schwerpunkt auf binäre, nominal mehrstufige und ratioskalierte Merkmale gelegt, weil diese Skalentypen in der Praxis die größte Relevanz besitzen. Umgesetzt wurden überwiegend Distanz- und Ähnlichkeitsmaße. So wurden für binäre Merkmale beispielsweise der M- und der S-Koeffizient, für nominal mehrstufige Merkmale der modifizierte M-Koeffizient und für ratioskalierte Merkmale die L_r-Distanzen, das normierte Skalarprodukt und die MAHALANOBIS-Distanz implementiert (vgl. Abschnitt 3.2.1.3).

Für die Ähnlichkeitsfunktionen der binären und mehrstufigen Merkmale wurden mittels der Transformation $d_{jk} = 1 - s_{jk}$ Distanzmaße geschaffen, weil diese bei der Arbeit mit ratioskalierten Distanzmaßen gebraucht wurden.

Weiterhin wurde die Möglichkeit geschaffen, hybride Proxymitäten zu berechnen. In der Funktion calcAD werden die Objektvektoren nach ihren Merkmalen aufgespalten und für die einzelnen Teilvektoren mit den ihrem Skalentyp zugehörigen Maßen die Proximität berechnet. Die so entstandenen Proximitäten werden dann in Abhängigkeit von der Anzahl der Merkmale des jeweiligen Skalentyps gewichtet und zusammenaddiert (vgl. Abschnitt 3.2.1.4).

Klassifikationsverfahren Für die reine disjunktive Klassifikation wurde der Algorithmus „Rekursiver Aufbau um Kerne“ implementiert.

Weiterhin wurden die agglomerativen Verfahren zur Erzeugung von indizierten Dendrogrammen implementiert. Zur Beschleunigung und Vereinheitlichung wurde dabei die Berechnung mit rekursiven Distanzen verwendet (vgl. Abschnitt 3.3.2.3). So konnten z.B. das Ward- oder das Single-

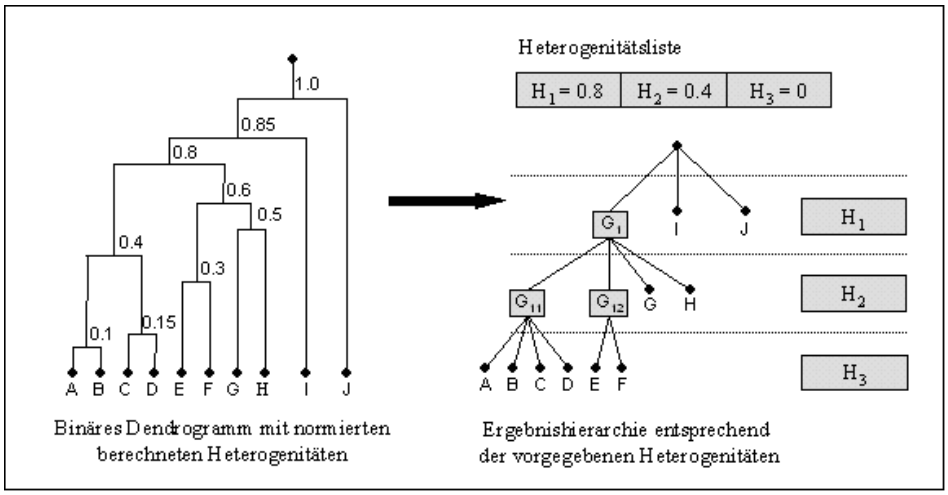

Abbildung 4.A: Algorithmus zur Erzeugung eines n-nären Hierarchiebaumes aus indiziertem Dendrogramm mit 10 Objekten durch Vorgabe der Heterogenitätsstufen 0.8, 0.4 und 0 (Graphik entwickelt von Mathias Kreuseler, Uni Rostock, Fachbereich Informatik)

Linkage-Verfahren leicht aus dem allgemeinen Algorithmus durch Paramteränderungen abgeleitet werden. Bei der Fusion zweier Klassen wird bei dieser Vorgehensweise die einer Klasse zugehörige Spalte und Zeile aus der TriangleMatrix entfernt und die Zeile und Spalte der anderen Klasse mit der entsprechenden Rekursionsformel (vgl. [Boc74] S. 404 und [BEPW96] S. 286 - 287) neu berechnet. Es werden also statt der Neuberechnung der ganzen Matrix lediglich die Proximitäten der neu entstandenen Klasse mit den unveränderten Klassen bestimmt.

Algorithmus zur Erzeugung eines n-nären Hierarchiebaumes aus indiziertem Dendrogramm Wie bereits erwähnt wurde, ist für die Visualisierung ein indiziertes Dendrogramm oft zu unübersichtlich. Aus diesem Grund wird das Dendrogramm für die Darstellung in mehrere Hierarchieebenen eingeteilt, in denen disjunkte Klassifikationen erzeugt werden. Ein entsprechender Algorithmus wurde von uns entwickelt.

Abbildung 4.A zeigt beispielhaft, wie mit diesem Algorithmus aus einem indizierten Dendrogramm mit 10 Objekten $O_1, \ldots, O_N$ ein Hierarchiebaum mit den drei Heterogenitätsstufen 0.8, 0.4 und 0 aufgebaut wird. Wie man dort sieht, liegen in der Hierarchiestufe H_3 ausschließlich Einzelobjekte, weil nur diese eine Herogenität von 0 besitzen. In der nächsthöhergelegenen Stufe 0.4 wurden alle die Klassen und Objekte eingegliedert, die eine maximale Heterogenität kleiner gleich 0.4 besitzen. Alle Unterklassen dieser Klassen wurden aufgrund ihrer geringeren Heterogenität nicht in den neuen Baum übernommen. Hiervon ausgeschlossen sind die Klassen, die unterhalb oder

auf der nächsttieferen Heterogenitätsstufe liegen. Nach dem selben Prinzip wurden Klassen und Objekte auf der Heterogenitätsstufe 0.8 eingefügt.

Allgemein folgt der Algorithmus den folgenden Schritten:

1. Erzeuge die Wurzel des Hierarchiebaumes.
2. Wähle aus der Heterogenitätsstufenliste die erste Heterogenitässtufe aus.
3. Wähle die beiden Sohnknoten der Dendrogrammwurzel aus.
4. Untersuche für die beiden aktuellen Knoten, ob ihre Heterogenitätswerte bereits kleiner gleich der aktuellen vorgegebenen Heterogenitätsstufe sind.
5. - Fall 1: Füge für die Knoten, die unterhalb der Schranke liegen, einen neuen Sohnknoten im Hierarchiebaum ein. Merke dir den Vaterknoten des untersuchten Knotens.
   - Fall 2: Für die Knoten, die oberhalb der Schranke liegen, untersuche wiederum deren Sohnknoten mit Schritt 4.
6. Wiederhole 4., bis für alle entsprechend gefundenen Sohnknoten unterhalb der aktuellen Hierarchiestufe liegen. (Dies geschieht spätestens, wenn die Klassen nur noch ein Objekt enthalten[7].)
7. Wähle aus der Heterogenitätsstufenliste die nächste Heterogenitässtufe aus und fahre mit den in Schritt 5 gemerkten Vaterknoten[8] mit Schritt 4. fort.
8. Falls Heterogenitätsliste ist am Ende: STOP des Algorithmus.

[7] Vorraussetzung für den Algorithmus: Alle Objekte mit der Distanz = 0 bzw. Ähnlichkeit = 1 müssen vorher reduziert werden.

[8] bei denen Fall 1 in der letzten Heterogenitätsstufe aufgetreten ist

Kapitel 5

Fallbeispiele

In diesem Abschnitt wird anhand von zwei Beispielen die Nutzung meines Tools vorgestellt. Beschrieben wird die Festlegung der Deskriptoren, die daraus resultierende Vorverarbeitung, die Maßauswahl und die daran anschließende Klassifikation. Vorgestellt werden hierarchische Klassifikationen. Diese wurden durch die Erzeugung eines Dendrogramms und die Transformation des so erhaltenen Dendrogramms in einen n-nären Hierarchiebaum gebildet. Um die Einsetzbarkeit der Vorverarbeitung für Visualisierungssysteme zu verdeutlichen, werden die Ergebnisse der Klassifikationen mit dem MagicEyeView[1] und mit dem System KOAN[2] dargestellt.

5.1 Der Autodatensatz

Vorstellung des Datensatzes Bei diesem Datensatz handelt es sich um 38 amerikanische PKW (Objekte) mit 6 Merkmalen. Die Merkmale „MPG“ (Meilen pro Galone), „Weight“ (Gewicht), „Drive-ratio“ (Fahreffektivität), „ Horsepower“(Pferdestärke), „Cylinders“(Anzahl der Zylinder) und „Displacement“(Hubraum) wurden ratioskaliert dargestellt. Angegeben sei hier ein Auszug aus der Datendatei (s. Anhang A.1):

„6 38
Car MPG Weight Drive-Ratio Horsepower Displacement Cylinders
Buick-Estat-Wagon 16.9 4.36 2.73 155.00 350.00 8.00
Ford-Country-Squire-Wagon 15.5 4.05 2.26 142.00 351.00 8.00
Chevy-Malibu-Wagon 19.2 3.60 2.56 125.00 267.00 8.00
[...]“

Festlegung des Daten- und Nutzerdeskriptors In den Datendeskriptor erfolgten folgende Eintragungen: Alle 6 Merkmale sind ratioskaliert

[1] Visualisierungskomponente entwickelt von ... an der Universität Rostock, Fachbereich Informatik

[2] KOntextANalysator, entwickelt bei Siemens AG, Abteilung ZT

und für die Klassifikation relevant[3].

Im Nutzerdeskriptor wurde festgelegt, daß eine hierarchische disjunkte Klassifikation entstehen soll. Weiterhin wurden festgelegt, daß absolute Distanzen Grundlage der Klassifikation sein sollen[4]. Für diese Beispiele variiert wurde, ob die entstehenden Klassen eher kettenförmig sein und Ausreißer identifizieren sollen (Fall 1) oder eher runde gleichgroße Klassen erzeugen sollen (Fall 2). Weiterhin festgelegt wurde eine Normierung der Werte der einzelnen Merkmale auf das Intervall (0..1) (siehe (3.1)). Diese wurde gewählt, weil z.B. das Merkmal „Cylinders“ mit einem Intervall von 4 bis 8 z.B. mit dem Intervall des Merkmals „Horsepower“ von 65 bis 150 vergleichbar wird. Weil kaum Ausreißer in den einzelnen Merkamalen auftreten, ist die Nutzung der Null-Eins-Normierung sinnvoll. Beim Auftreten von Ausreißern wäre die Nutzung der Varianznormierung (3.2) eher sinnvoll. Eine Ausreißerextraktion wurde nicht gefordert.

Der berechnete Prozeßdeskriptor Aus den vorgegebenen Deskriptoren wurden dann mit Hilfe der Funktion „calcProzessDeskriptor“ das Distanzmaß und das Klassifikationsverfahren ausgewählt. Für die (ratioskalierten) Merkmale wurde die L_3-Distanz ausgewählt (Höhergewichtung großer Distanzen). Im Fall 1 fiel die Wahl auf das agglomerative Single-Linkage-Verfahren und in Fall 2 auf das agglomerative Ward-Verfahren.

Pipelineausführung:

1. **Präprozeß** Im Präpozeß erfolgte eine Normierung der ratioskalierten Merkmale auf das Intervall (0..1). Ausreißer und Korrelationen wurden nicht beseitigt.

2. **Berechnung der Distantmatrix** In diesem Schritt erfolgte die Berechnung einer 38×38 großen Dreiecks-Distanzmatrix.

3. **Klassifikation** In diesem Schritt erfolgte die Berechnung der beiden indizierten Dendrogramme mit dem Single-Linkage-Verfahren und dem Ward-Verfahren (vgl. Kapitel 3.3.2.3 - Agglomerative Verfahren).

Interpretation Weil die durch die Pipelineausführung erhaltenen Dendrogramme ziehmlich unübersichtlich sind, wurden sie für die Visualisierung und die Interpretation in Hierarchiebäume überführt. Als erstes wurden die beiden Dendrogramme mit den Heterogenitätsstufen $0.5, 0.3$ und 0 in derartige Hierarchiebäume überführt (vgl. Algorithmus aus Kapitel 4.2.1). Die Ergebnishierarchien sind in den Abbildungen 5.A und 5.B dargestellt.

In Abbildung 5.A ist zu erkennen, daß das Single Linkage Verfahren auf der Heterogenitätsstufe 0.5 die drei Ausreißer „Audi-5000“, „Datsun-810“ und „Mercury-Zephir“ identifiziert. Ansonsten ist diese Hierarchie je-

[3] d.h. alle wurden einbezogen

[4] keine Beachtung von Profilverläufen

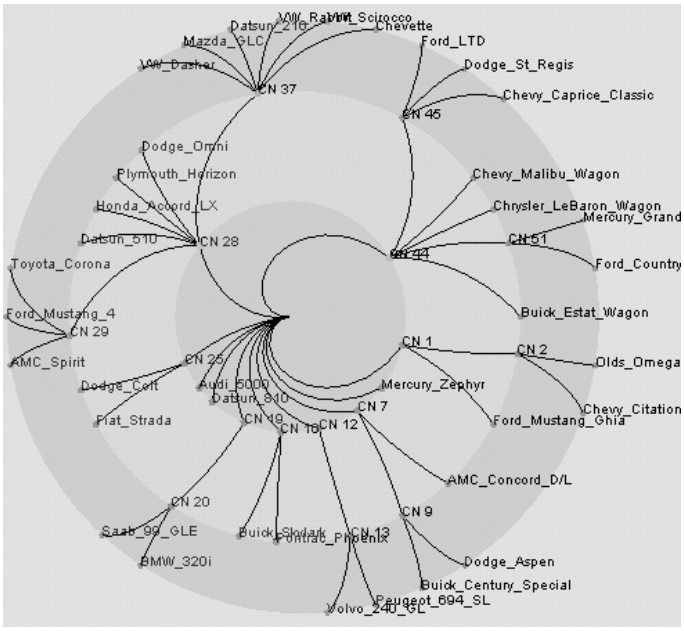

Abbildung 5.A: MagicEyeView-Darstellung des Autodatensatzes durch einem Hierarchiebaum mit drei Heterogenitätsstufen $0.5, 0.3$ und 0; Basis:Single-Linkage-Verfahren

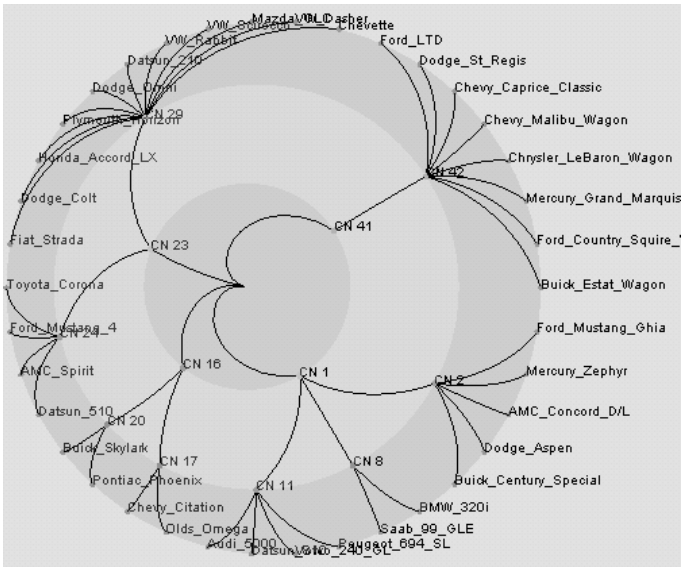

Abbildung 5.B: MagicEyeView-Darstellung des Autodatensatzes durch einem Hierarchiebaum mit drei Heterogenitätsstufen $0.5, 0.3$ und 0; Basis:Ward-Verfahren

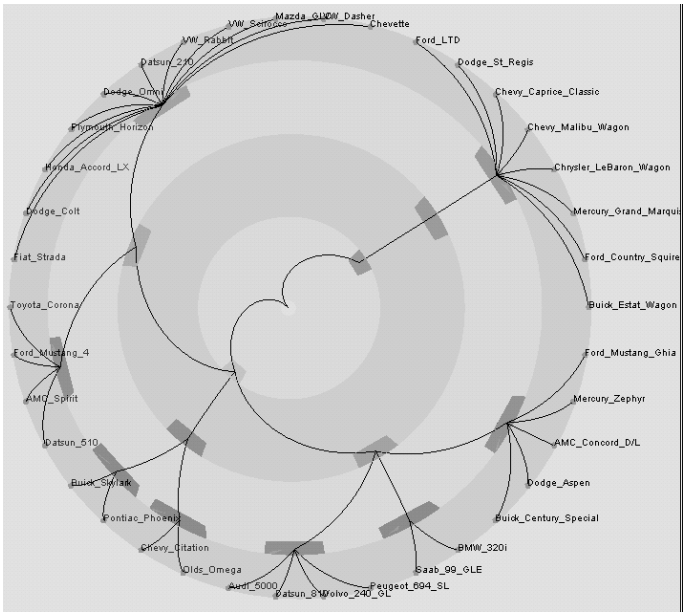

Abbildung 5.C: MagicEyeView-Darstellung des Autodatensatzes durch einem Hierarchiebaum mit vier Heterogenitätsstufen $0.8, 0.5, 0.3$ und 0; Basis:Ward-Verfahren

doch relativ unübersichtlich, weil viele Klassen mit unterschiedlichen Klassenstärken entstehen. Dies geschieht aufgrund der kontrahierenden Eigenschaft des Single-Linkage-Verfahrens: Viele ein- oder zweiobjektige Klassen stehen wenigen großen Klassen gegenüber.

Im Gegensatz dazu ist die mit dem Ward-Verfahren erzeugte Darstellung 5.B wesentlich leichter interpretierbar. Dieses Verfahren ist konservativ, und es entstehen daher vier in etwa homogene Klassen auf der Hierarchiestufe 0.5. Ausreißer sind hier nicht zu erkennen. Zur besseren Interpretation dieser Klassen wurde nun die weitere Hierarchiestufe 0.8 hinzugenommen. Damit ergibt sich die Darstellung 5.C.

Man sieht dort, daß die drei in Abbildung 5.B links und unten befindlichen Klassen sich auf dem Heterogenitätsniveau 0.8 zu einer Oberklasse vereinen, während die Klasse rechts oben erhalten bleibt.

Aufgrund der Analyse der t-Werte für die auf der Heterogenitätsstufe 0.8 entstandenen Klassen (vgl. Tabelle 5.a) erkennt man, daß die nicht veränderte alleinstehende Klasse die „großen“ Wagen representiert und die fusionierten drei Klassen die „kleineren“ Wagen repräsentieren. Begründet werden kann diese Aussage durch Analyse der t-Werte der einzelnen Merkmale der Klassenobjekte (vgl. Kapitel 3.3.3). In der „Großwagen“-Klasse sind die t-Werte der Merkmale „Weight“, „ Horsepower“, „Cylinders“ und „Displacement“, die man im allgemeinen größeren Wagen zuschreibt, wesentlich

Merkmal	„Kleinere Wagen“	„Groß-Wagen“
MPG	1.84	-6.91
Weight	-2.44	9.15
Drive-Ratio	2.13	-7.99
Horsepower	-2.25	8.45
Displacement	-2.74	10.27
Cylinders	-2.67	10.02

Tabelle 5.a: Berechnung der t-Werte der zwei auf der Heterogenitätsstufe 0.8 in der Abbildung 5.C gebildeten Hauptklassen, um Rückschlüsse über die Präsenz von Merkmalen in diesen Klassen zu erhalten.

größer als Null. Dies bedeutet eine Überrepräsentation dieser Merkmale in dieser Klasse. Die Merkmale „MPG“ und „Drive-ratio“, die eher „kleineren Wagen“ zugeschrieben werden, sind mit t-Werten kleiner 0 in dieser Klasse unterrepräsentiert. Genau entgegengesetzt dazu verhalten sich die Wagen der anderen Klasse, bei denen „MPG“ und „Drive-ratio“ überrepräsentiert und die anderen Merkmale überrepräsentiert sind. Diese Klassen enthalten damit die „kleineren und mittleren Wagen“. Allerdings ist die Ausprägung der Eigenschaften nicht so stark wie in der „Großwagen-Klasse“, was eine stärkere Streuung innerhalb dieser Klasse vermuten läßt. Diese Vermutung wird noch dadurch bestärkt, daß wie bereits in Abb 5.B dargestellt, diese Klasse auf dem Heterogenitätsniveau 0.5 in drei Klassen zerfällt.

Als letzte Möglichkeit der Darstellung dieses Datensatzes soll die Möglichkeit der Verfeinerung der Daten durch interaktive Steuerung vorgestellt werden. Idee dabei ist, alle Klassen bzw. Objekte einer Hierarchieebene in einer Darzustellung zu verbinden. Besteht nun die Notwendigkeit, eine bestimmte Klasse genauer zu untersuchen, kann der Nutzer des Visualisierungssystems durch Mausklick auf die graphische Repräsentation der Klasse eine Darstellung aller Unterklassen dieser Klasse auf der nächsttieferen Heterogenitätsstufe erhalten. Dadurch ist eine gute Navigation durch Datenmengen möglich, bedenkt man, daß der Nutzer mehrere über- bzw. untergeordnete Ebenen gleichzeitig darstellen kann.

Als Beispiel für eine solche hierarchische geschachtelte Visualisierung wurde die Darstellung der einzelnen Hierarchieebenen in einem KOAN-Graphen durchgeführt. Abbildung 5.D zeigt die Autohierarchie aus Abbildung 5.B auf der Heterogenitätsstufe 0.5. Zu sehen sind hier wieder die vier sich auf dieser Stufe bildenden Klassen. Deutlich zu erkennen ist auch die Separation der drei hinteren Klassen („kleine und mittlere Wagen“) von der vorderen („Groß-Wagen“). Wird nun auf die am weitesten oben dargestellte Klasse ein Doppelklick ausgeführt, öffnet sich ein neues KOAN-Fenster, welches in diesem Fall alle in dieser Klasse enthaltenen Objekte darstellt[5] (s.

[5]Es wird direkt Hierarchieebene 0 gewechselt.

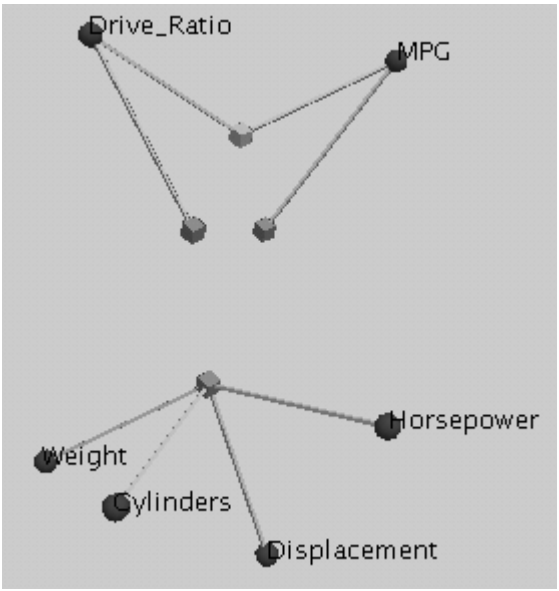

Abbildung 5.D: KOAN-Darstellung des Autodatensatzes erzeugt mit dem Ward-Verfahren auf der Hierarchiestufe 0.5; Die Kugeln repräsentieren die Merkmale und die Würfel die Klassen.

Abb. 5.E). Auch auf dieser Ebene befinden sich die Objekte genau wie ihre Oberklasse in Abb. 5.D nahe den Merkmalen „MPG“ und „Drive-Ratio“.

5.2 Der Schachspielerdatensatz

In diesem Abschnitt sollen Klassifikationen des Schachspielerdatensatzes kurz vorgestellt werden. Das Interessante an diesem Datensatz besteht darin, daß hier gemischte Merkmale vorkommen. Für die 46 Schachspieler liegen die drei nominal mehrstufigen Merkmale „Land“, „Titel“ und „Verein“ und die beidem ratioskalierten Merkmale „ELO-Zahl“ (Bewertung der Spielstärke) und „+/-“ (Veränderung der ELO-Zahl im letzten Quartal) vor.

Diese Daten wurden bei automatischer Auswahl des verallgemeinerten M-Koeffizienten (3.37) für die mehrstufigen Merkmale und bei Auswahl der L_3-Distanz für die ratioskalierten Merkmale mit dem Ward-Verfahren klassifiziert. In Abbildung 5.F sieht man die sich bildende Struktur mit den einzelnen Spielern. Beschriftet man nun die einzelnen nichttrivialen Klassen des Baumes, erhält man eine sehr gute Übersicht über den Datensatz (Abb 5.G). Grundsätzlich erkennt man auf der obersten Hierarchiestufe 0.7 eine Aufspaltung in vier Hauptklassen. Dabei bildet sich eine Großmeister-Klasse mit den drei stärksten Mecklenburg-Vorpommeranischen Spielern. Die nächstgrößere Klasse repräsentiert die starken Spieler des VfL Neuklo-

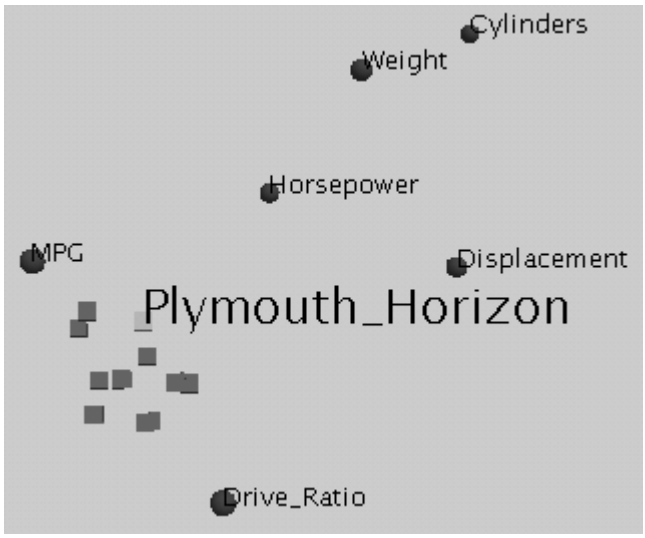

Abbildung 5.E: KOAN-Darstellung des Autodatensatzes erzeugt mit dem Ward-Verfahren auf der Objektebene (Hierarchiestufe 0.0) ; Die Kugeln repräsentieren die Merkmale und die Würfel die Objekte.

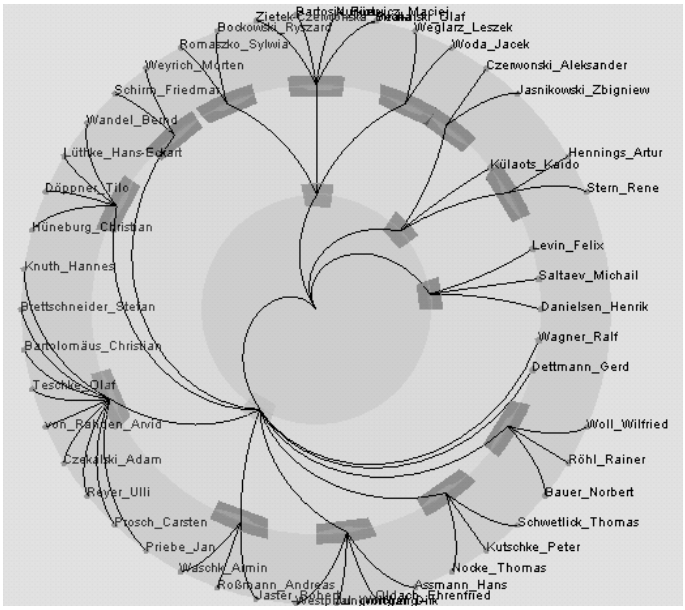

Abbildung 5.F: Namensbeschriftete MagicEyeView-Darstellung des Schachspielerdatensatzes durch einem Hierarchiebaum mit drei Heterogenitätsstufen $0.7, 0.3$ und 0; Basis:Ward-Verfahren

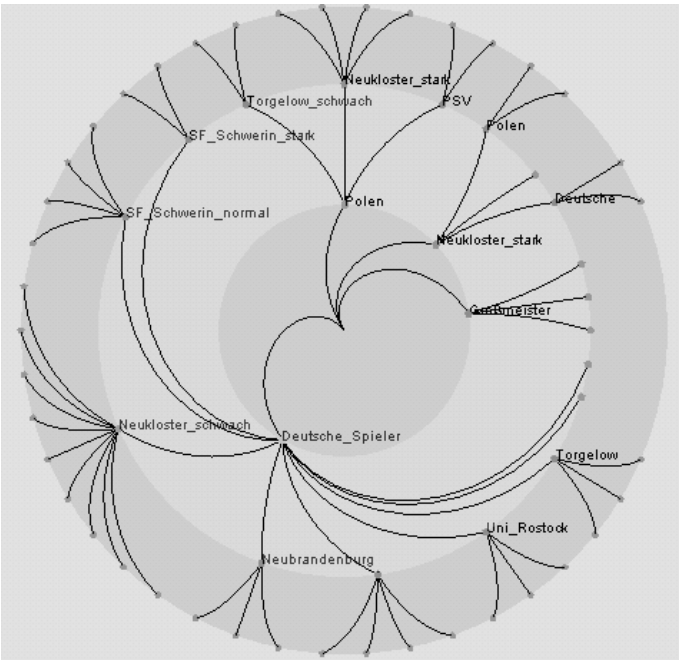

Abbildung 5.G: Knotenbeschriftete MagicEyeView-Darstellung des Schachspielerdatensatzes durch einem Hierarchiebaum mit drei Heterogenitätsstufen $0.7, 0.3$ und 0; Basis:Ward-Verfahren

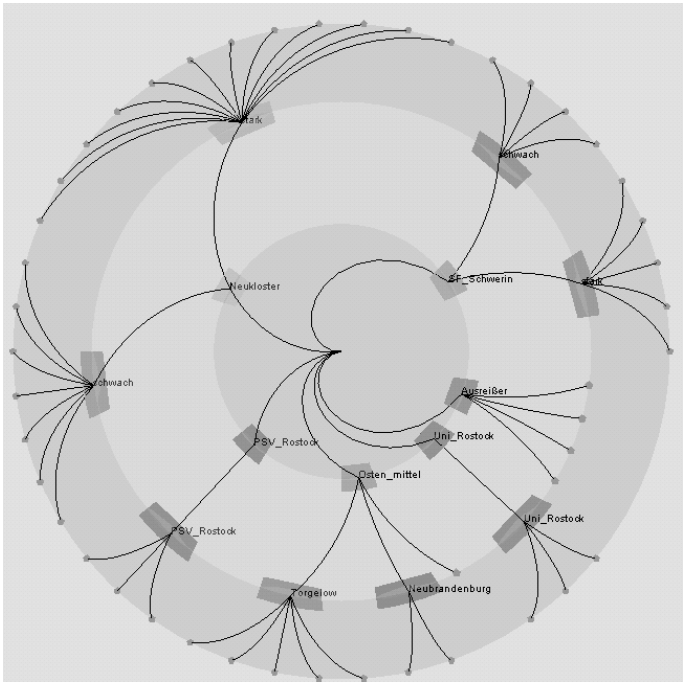

Abbildung 5.H: MagicEyeView-Darstellung des Schachspielerdatensatzes durch einem Hierarchiebaum mit drei Heterogenitätsstufen 0.7, 0.3 und 0 unter Auswahl der Merkmale „ELO-Zahl“ und „Verein“; Basis:Ward-Verfahren

ster, die sich wiederum in 3 Unterklassen aufspaltet. Die dritte, noch größere Klasse bildet die Klasse von polnischen Spielern in Mecklenburger Vereinen. Die letzte und größte Klasse besteht einzig aus deutschen Spielern mittlerer und unterer Spielklasse. Diese Klasse zerfällt vor allem aufgrund der Vereinszugehörigkeit in weitere Klassen.

Weiterhin fällt bei der Interpretation dieser Klassifikationshierarchie auf, daß das Merkmal „+/-“ kaum Einfluß auf das Klassifikationergebnis zu haben scheint.

Um die Möglichkeit der Auswahl von Merkmalen vorzustellen, wurde in diesem Datensatz eine Reduktion auf die beiden Merkmale „ELO-Zahl“ und „Verein“ durchgeführt. Die entstandene Klassifikation in Abbildung 5.H gibt diesen Sachverhalt wieder.

Kapitel 6

Zusammenfassung, Ergebnisse und Ausblick

In diesem Kapitel sollen die durchgeführten Untersuchungen noch einmal zusammengefaßt, die erzielten Ergebnisse bewertet und ein Ausblick gegeben werden.

Grundgedanke dieser Arbeit war es, die Probleme von Visualisierungssystemen bei der Darstellung großer Datenmengen zu vermindern, da diese oft erhebliche Mängel in Bezug auf die Übersichtlichkeit und Navigationsfähigkeit solcher Datenmengen aufweisen. Aus diesem Grunde wurde die Idee der Vorverarbeitung aufgegriffen und ein Konzept entwickelt, das es ermöglicht, die Eingangsdaten der Visualisierung flexibel zu strukturieren und zu klassifizieren.

Hierfür wurde eine Literaturrecherche durchgeführt, um bereits vorhandene Verfahren auf ihre Nutzbarkeit für die Visualierung zu untersuchen und relevante Eigenschaften unter dem Aspekt der Visualisierung zu extrahieren. Auf den Erkenntnissen dieser Untersuchungen aufbauend wurde das Werkzeug „InfoSonne“ entwickelt, welches sowohl der Anforderung Flexibilität als auch den Anforderungen Modularität und Erweiterbarkeit entspricht.

Die mit diesem Werkzeug erzielten Ergebnisse bestätigen, daß die vorverarbeitende Strukturierung die Darstellung und Interpretierbarkeit großer Datenmengen wesentlich verbessert.

Ausblickend soll darauf hingewiesen werden, daß das Tool „InfoSonne“ aufgrund seiner Flexibilität leicht um neue Verfahren erweitert werden kann. Beispielsweise könnten Verfahren aus der aktuellen Forschung und Strukturierungen mit neuronalen Netzen integriert werden. Weiterhin könnten bei Bedarf neue Maße für die Untersuchung neuer Datensätze mit bisher nicht integrierten Merkmalseigenschaften eingefügt werden.

Abschließend bleibt festzustellen, daß die vorverarbeitende Strukturierung breite Anwendungsperspektiven in der Visualisierung verspricht.

Literaturverzeichnis

[BEPW96] Klaus Backhaus, Bernd Erichson, Wulff Plinke, and Rolf Weber, *Multivariate analysemethoden. eine anwendungsorientierte einführung.*, 8 ed., Springer, 1996.

[Boc74] Hans Hermann Bock, *Automatische klassifikation*, Vandenhoeck & Ruprecht, Göttingen, 1974.

[Goo64] D.W. Goodall, *A probabilistic similarity index*, Nature (1964), no. 203, 1098.

[Ham61] U. Hamann, *Merkmalsbestand und verwandtschaftsbeziehungen der farinose. ein beitrag zum system der monokotyledonen.*, Willdenowia (1961), no. 2, 639 – 768.

[LW67] G.N. Lance and W.T. Williams, *Mixed-data classificatory programs i. agglomerative systems.*, Australian Computer J. (1967), no. 1, 15–20.

Tabellenverzeichnis

Abbildungsverzeichnis

Anhang A

Beispieldatensätze

A.1 Der Autodatensatz

```
6 38
Car MPG Weight  Drive_Ratio Horsepower  Displacement    Cylinders
Buick_Estat_Wagon   16.9    4.36    2.73    155.00  350.00  8.00
Ford_Country_Squire_Wagon   15.5    4.05    2.26    142.00  351.00  8.00
Chevy_Malibu_Wagon  19.2    3.60    2.56    125.00  267.00  8.00
Chrysler_LeBaron_Wagon  18.5    3.94    2.45    150.00  360.00  8.00
Chevette    30  2.15    3.70    68.00   98.00   4.00
Toyota_Corona   27.5    2.56    3.05    95.00   134.00  4.00
Datsun_510  27.2    2.30    3.54    97.00   119.00  4.00
Dodge_Omni  30.9    2.23    3.37    75.00   105.00  4.00
Audi_5000   20.3    2.83    3.90    103.00  131.00  5.00
Volvo_240_GL    17  3.14    3.50    125.00  163.00  6.00
Saab_99_GLE 21.6    2.79    3.77    115.00  121.00  4.00
Peugeot_694_SL  16.2    3.41    3.58    133.00  163.00  6.00
Buick_Century_Special   20.6    3.38    2.73    105.00  231.00  6.00
Mercury_Zephyr  20.8    3.07    3.08    85.00   200.00  6.00
Dodge_Aspen 18.6    3.62    2.71    110.00  225.00  6.00
AMC_Concord_D/L 18.1    3.41    2.73    120.00  258.00  6.00
Chevy_Caprice_Classic   17  3.84    2.41    130.00  305.00  8.00
Ford_LTD    17.6    3.72    2.26    129.00  302.00  8.00
Mercury_Grand_Marquis   16.5    3.95    2.26    138.00  351.00  8.00
Dodge_St_Regis  18.2    3.83    2.45    135.00  318.00  8.00
Ford_Mustang_4  26.5    2.58    3.08    88.00   140.00  4.00
Ford_Mustang_Ghia   21.9    2.91    3.08    109.00  171.00  6.00
Mazda_GLC   34.1    1.97    3.73    65.00   86.00   4.00
Dodge_Colt  35.1    1.91    2.97    80.00   98.00   4.00
AMC_Spirit  27.4    2.67    3.08    80.00   121.00  4.00
VW_Scirocco 31.5    1.99    3.78    71.00   89.00   4.00
```

```
Honda_Accord_LX 29.5    2.13    3.05    68.00   98.00   4.00
Buick_Skylark   28.4    2.67    2.53    90.00   151.00  4.00
Chevy_Citation  28.8    2.59    2.69    115.00  173.00  6.00
Olds_Omega  26.8    2.70    2.84    115.00  173.00  6.00
Pontiac_Phoenix 33.5    2.55    2.69    90.00   151.00  4.00
Plymouth_Horizon    34.2    2.20    3.37    70.00   105.00  4.00
Datsun_210  31.8    2.02    3.70    65.00   85.00   4.00
Fiat_Strada 37.3    2.13    3.10    69.00   91.00   4.00
VW_Dasher   30.5    2.19    3.70    78.00   97.00   4.00
Datsun_810  22  2.81    3.70    97.00   146.00  6.00
BMW_320i    21.5    2.60    3.64    110.00  121.00  4.00
VW_Rabbit   31.9    1.92    3.78    71.00   89.00   4.00

6 38
Car MPG Weight  Drive_Ratio Horsepower  Displacement    Cylinders
Buick_Estat_Wagon   16.9    4.36    2.73    155.00  350.00  8.00
Ford_Country_Squire_Wagon   15.5    4.05    2.26    142.00  351.00  8.00
Chevy_Malibu_Wagon  19.2    3.60    2.56    125.00  267.00  8.00
Chrysler_LeBaron_Wagon  18.5    3.94    2.45    150.00  360.00  8.00
Chevette    30  2.15    3.70    68.00   98.00   4.00
Toyota_Corona   27.5    2.56    3.05    95.00   134.00  4.00
Datsun_510  27.2    2.30    3.54    97.00   119.00  4.00
Dodge_Omni  30.9    2.23    3.37    75.00   105.00  4.00
Audi_5000   20.3    2.83    3.90    103.00  131.00  5.00
Volvo_240_GL    17  3.14    3.50    125.00  163.00  6.00
Saab_99_GLE 21.6    2.79    3.77    115.00  121.00  4.00
Peugeot_694_SL  16.2    3.41    3.58    133.00  163.00  6.00
Buick_Century_Special   20.6    3.38    2.73    105.00  231.00  6.00
Mercury_Zephyr  20.8    3.07    3.08    85.00   200.00  6.00
Dodge_Aspen 18.6    3.62    2.71    110.00  225.00  6.00
AMC_Concord_D/L 18.1    3.41    2.73    120.00  258.00  6.00
Chevy_Caprice_Classic   17  3.84    2.41    130.00  305.00  8.00
Ford_LTD    17.6    3.72    2.26    129.00  302.00  8.00
Mercury_Grand_Marquis   16.5    3.95    2.26    138.00  351.00  8.00
Dodge_St_Regis  18.2    3.83    2.45    135.00  318.00  8.00
Ford_Mustang_4  26.5    2.58    3.08    88.00   140.00  4.00
Ford_Mustang_Ghia   21.9    2.91    3.08    109.00  171.00  6.00
Mazda_GLC   34.1    1.97    3.73    65.00   86.00   4.00
Dodge_Colt  35.1    1.91    2.97    80.00   98.00   4.00
AMC_Spirit  27.4    2.67    3.08    80.00   121.00  4.00
VW_Scirocco 31.5    1.99    3.78    71.00   89.00   4.00
Honda_Accord_LX 29.5    2.13    3.05    68.00   98.00   4.00
Buick_Skylark   28.4    2.67    2.53    90.00   151.00  4.00
Chevy_Citation  28.8    2.59    2.69    115.00  173.00  6.00
```

```
Olds_Omega  26.8    2.70    2.84    115.00  173.00  6.00
Pontiac_Phoenix 33.5    2.55    2.69    90.00   151.00  4.00
Plymouth_Horizon    34.2    2.20    3.37    70.00   105.00  4.00
Datsun_210  31.8    2.02    3.70    65.00   85.00   4.00
Fiat_Strada 37.3    2.13    3.10    69.00   91.00   4.00
VW_Dasher   30.5    2.19    3.70    78.00   97.00   4.00
Datsun_810  22  2.81    3.70    97.00   146.00  6.00
BMW_320i    21.5    2.60    3.64    110.00  121.00  4.00
VW_Rabbit   31.9    1.92    3.78    71.00   89.00   4.00
```

A.2 Der Schachspielerdatensatz

```
5 46
Name                    Land Titel Elo  +/- Verein
Danielsen_Henrik        DEN  GM    2506 -4  SF_Schwerin
Saltaev_Michail         UZB  GM    2504 -21 SF_Schwerin
Levin_Felix             DEU  GM    2496 +16 SF_Schwerin
Stern_Rene              DEU  IM    2462 -3  VfL_Blau-Weiß_Neukloster
Külaots_Kaido           EST  IM    2426 -19 VfL_Blau-Weiß_Neukloster
Jasnikowski_Zbigniew    POL  IM    2411 +11 VfL_Blau-Weiß_Neukloster
Czerwonski_Aleksander   POL  IM    2397 +7  VfL_Blau-Weiß_Neukloster
Woda_Jacek              POL  FM    2385  0  SYC_Rostock
Weglarz_Leszek          POL  FM    2379 -16 SYC_Rostock
Weyrich_Morten          DEU  FM    2354 +9  SF_Schwerin
Schirm_Friedmar         DEU  FM    2330  0  VfL_Blau-Weiß_Neukloster
Wandel_Bernd            DEU  TL    2330  0  SF_Schwerin
Michalski_Olaf          POL  TL    2325 +20 VfL_Blau-Weiß_Neukloster
Hennings_Artur          DEU  IM    2314 -6  VfL_Blau-Weiß_Neukloster
Nurkiewicz_Maciej       POL  TL    2301 +16 VfL_Blau-Weiß_Neukloster
Knuth_Hannes            DEU  TL    2275  0  VfL_Blau-Weiß_Neukloster
Bockowski_Ryszard       POL  TL    2264 -11 Torgelower_SV_Greif
Bartosik_Pjotr          POL  TL    2260  0  VfL_Blau-Weiß_Neukloster
Nocke_Thomas            DEU  TL   2250  0  HSG_Uni_Rostock
Waschk_Armin            DEU  TL   2245  0  SG_Eintracht_Neubrandenburg
Hüneburg_Christian      DEU  TL   2238 +28 SF_Schwerin
Brettschneider_Stefan   DEU  TL   2230  0  VfL_Blau-Weiß_Neukloster
Bartolomäus_Christian   DEU  TL   2230.001  0  VfL_Blau-Weiß_Neukloster
Bauer_Norbert           DEU  TL   2210 -15 Torgelower_SV_Greif
Jaster_Robert           DEU  TL   2200  0  SYC_Rostock
Lüthke_Hans-Eckart      DEU  TL   2200  0  SF_Schwerin
Döppner_Tilo            DEU  TL   2190  0  SF_Schwerin
Dettmann_Gerd           DEU  TL   2184 -16 Post_Güstrow
Woll_Wilfried           DEU  TL   2180  0  Torgelower_SV_Greif
```

Roßmann_Andreas	DEU	TL	2175	0	SG_Eintracht_Neubrandenburg
Teschke_Olaf	DEU	TL	2174	+14	VfL_Blau-Weiß_Neukloster
Röhl_Rainer	DEU	TL	2174	-21	Torgelower_SV_Greif
von_Rahden_Arvid	DEU	TL	2166	-9	VfL_Blau-Weiß_Neukloster
Czekalski_Adam	DEU	TL	2163	0	VfL_Blau-Weiß_Neukloster
Kutschke_Peter	DEU	TL	2155	0	HSG_Uni_Rostock
Westphal_Wolfgang	DEU	TL	2137	0	1.Schweriner_SV
Reyer_Ulli	DEU	TL	2127	-3	VfL_Blau-Weiß_Neukloster
Jungmichel_Dirk	DEU	TL	2108	0	FHS_Stralsund
Wagner_Ralf	DEU	TL	2105	-30	SF_Schwerin
Prosch_Carsten	DEU	TL	2093	+3	VfL_Blau-Weiß_Neukloster
Priebe_Jan	DEU	TL	2088	0	VfL_Blau-Weiß_Neukloster
Zietek-Czerwonska_Beata	POL	TL	2085	0	VfL_Blau-Weiß_Neukloster
Romaszko_Sylwia	POL	TL	2079	+14	Torgelower_SV_Greif
Oldach_Ehrenfried	DEU	TL	2058	+8	Ostsee_Warnemünde
Assmann_Hans	DEU	TL	2035	+5	TSV_1860_Stralsund
Schwetlick_Thomas	DEU	TL	2030	0	HSG_Uni_Rostock

Anhang B

Datendefinitionen

B.1 Definition der Klassen Vector und VectorArray: „VectorArray.hpp“

```
// Datei "VektorArray.hpp"
// enthält die Definition der Klassen VectorArray und Vector.


#ifndef VECTORARRAY_HPP
#define VECTORARRAY_HPP

#include "main.hpp"

/*************************************************************
*
*   class Vector
*
*         - Definition von Objektvektoren für die Klassifikation
*
*************************************************************/

class Vector
{
public:
    // Memberfunktionen
    Vector(ui NrDims = 0);          // Konstruktor mit Angabe der
                                    // Vektordimension
    Vector(Vector&);                // Copy-Konstruktor
    ~Vector();                      // Destruktor
```

```
    ValueType& operator[](ui x);    // Zugriffsoperator auf den Wert
                                    // der x.ten Dimension

    void operator=(Vector& v);      // Vergleichsoperator
    ui getDim();                    // Zugriff auf die Dimension
private:
    // Membervariablen
    ValueType* Data;           // Datenwerte des Vektors; ValueType
                               // wurde als Typ double definiert
    ui Dim;                    // Vektordimension

};

/*************************************************************
*
*   class VectorArray
*
*         - Definition der Objekt-Merkmals-Matrix
*
*************************************************************/

class VectorArray
{
    friend class FileIO;

private:
    // Membervariablen
    ui NrObjects;                       // Anzahl der Objekte
    ui NrAttr;                          // Anzahl der Merkmale
public:
    ValueType** Data;                   // Objekt-Merkmals-Matrix

    char** ObjNames;                    // Objektnamen
    char** AttNames;                    // Merkmalesnamen

     std::string ObjTypeName;           // Name des Objekttyps

    // Memberfunktionen
    VectorArray(ui argNrAttr = 0, ui argNrObj = 0); // Konstruktor
    VectorArray(VectorArray&);                      // Copy-Konstruktor
    ~VectorArray();                                 // Destruktor

    ValueType operator()(ui ObjNr, ui AttrNr); // Zugriffsoperator
                                    // auf Werte der
```

```
                                        // Objekt-Merkmals-Matrix
    Vector getVec(ui x);                // Zugriff auf einen Objektvektor

    ui getNrObjects();                     // Anzahl der Objekte
    ui getNrAttr();                        // Anzahl der Merkmale

    void loesche(ui VecNr);                // Löschen eines Objektes

    void writeVAtoFile(char* filename); // Hilfsfunktion: Ausgabe der
                                           // Matrix in Datei
};



#endif // VECTORARRAY_HPP
```

B.2 Definition der Klasse TriangleMatrix: „TriangleMatrix.hpp“

```
// Datei "TriangleMatrix.hpp"
// enthält die Definiertion der Klasse TriangleMatrix.

#ifndef TRIANGLEMATRIX_HPP
#define TRIANGLEMATRIX_HPP

/*****************************************************
*
*   Includes
*
*****************************************************/

#include "main.hpp"
#include <vector>

/*****************************************************
*
*   class TriangleMatrix
*
*       - Definition einer oberen bzw. unteren
*           Dreiecksmatrix als Kontainer für Distanz-
*           und Ähnlichkeitsmatrizen
*
*****************************************************/
```

```
class TriangleMatrix
{
public:
    // Memberfunktionen
    TriangleMatrix(ui size);        // Konstruktor
    ~TriangleMatrix();              // Destruktor

    double& operator()(ui j, ui k); // Zugriffsoperator
                          // auf den Ähnlichkeitswert s_jk
                          // oder den Distanzwert d_jk

    void deleteLineAndColumn(ui xy);// Löschen einer Spalte
                                // und der zugehörigen Zeile

     double getSize();              // Aktuelle Matrixgröße?

    // Hilfsfunktionen
    void writeToScreen();           // Bildschirmausgabe
    void writeToDatei(char* FileName); // Ausgabe in Datei

private:
    // Membervariable
    double** array;               // Matrixwerte
    ui max_size;                  // Anfangsgröße der Matrix
    std::vector<ui> validLinesAndColumns; // nicht gelöschte
                                  // Zeilen und Spalten
    ui size;                      // aktuelle Matrixgröße
    double equalVar;
};


#endif // TriangleMatrix_HPP
```

B.3 Definition der Klassen HierachyTree, TreeIter und Node: „Klassifikation.hpp“

```
// Datei "Klassifikation.hpp"
// enthält die Definiertion der Klassen HierachyTree, Node
// und  TreeIter.


#ifndef KLASSIFIKATION_HPP
#define KLASSIFIKATION_HPP
```

```
#include "main.hpp"
#include "VectorArray.hpp"
#include <list>
#include <fstream>// file stream operations


#include <deque>
#pragma warning (disable: 4786)

using namespace std;


/*****************************************************
*
* Data structures for classification
*
*
*****************************************************/

/*****************************************************
*
* class Node
*
* - represents a class(group), sons are part classes
* and leaves are Objects
* - Nodes are collected in HierachyTrees
*
*****************************************************/
class Node
{
public:
// member variables
list<Node*> sons; // childclasses of Node
double heterogenity; // Heterogenity of class
ui leafInfo; // contains reference key to object in
// vectorArray;
// if Node is not a leaf, value is MaxUi!

list<ui> Objects; // contains all objects (references) of
  // this class

// member functions
```

```
Node(ui argLeafInfo = MaxUi, double heterogenity = 0.0);
// constructor1
Node(double heterogenity); // constructor2
~Node(); // destructor

double& Heterogenity(); // access to heterogenity
ui getNrLeafs(); // number of class objects

class TreeIter begin(); // delivers begin iterator for subtree
class TreeIter end(); // delivers end iiterator of subtree

void fillObjects(list<ui>*); // fills variable Objects

void normHeterogenitiesInSubTree(double maxHeterogenity);
// norm of heterogenities: root h = 1, leaf h = 0

void makeHomogenitiesToHeterogenitiesInSubTree(
double minHomogenity, double maxHomogenity);
// change homogenity of son classes to herogenity
double findOutMaxTreeHomoValue(); // max. homogenity value
  // in subtree?


void heterFkt(list<Node*>& FatherBuildingList,
list<double>& Thresholds,
list<double>::iterator actThreshold);
// build HierachyTree of threshold levels on this node
// and subnodes

// output functions to KOAN
void writeNodeInfoToOneTypOFile(ofstream&, ofstream&, ui&,
VectorArray*, ui Level);
void writeNodeInfoToTypOHierachy(const char* childFileName,
class HierachyTree& tree, ui Level);

private:
// helping output function to KOAN
void writeObjOrAttTypOInfo(ofstream& datei1,
ui actObjOrAttId, VectorArray* va,
ui Level, const char* SubFileName = NULL);

};

/*********************************************************
```

```
*
* TreeIter
*
* -Iterator other all leaf objects of a HierachyTree Node
*
************************************************************/
class TreeIter
{
public:
friend Node;

// member functions
TreeIter();   // Konstructor
~TreeIter() {}; // Destruktor
TreeIter(TreeIter&); // Copy

void operator= (TreeIter&); // Assign
TreeIter& operator++(); // Increment
ui       operator* () // Access

bool operator!= (TreeIter&) ; // Not compare
bool operator== (TreeIter&) ; // Compare

private:

// helping member variables
typedef  list<Node*>::const_iterator NodeListIter;

deque<NodeListIter> mIterList;

list<Node*> mTreeNodeList;
}; // end iterator-class


/************************************************************
*
* class HierachyTree
*
* - is a tree collecting hierachic, disjunctiv and not
* disjunctive classification results
* - can be dendrogram (binary tree) or complex hierachy
* with heterogenity levels (n-ner tree)
* - consists of Nodes
* - Iteration other subclasses via TreeIter
```

```
*
************************************************************/
class HierachyTree
{
public:
friend Node;
// member variables
VectorArray* vectorArray; // related VectorArray
Node* root;   // root of tree

// member functions
HierachyTree(VectorArray*); // contructor
~HierachyTree(); // destructor

Node* generateFatherOfTwoNodes(Node* son1, Node* son2);
// helping function for building tree

void buildHeterogenityHierachyFromDendrogram(
const HierachyTree& Dendrogramm,
list<double>& heterogenLevel);
// constructs Hierachy with heterogenity levels

void normTreeHeterogenities(); // norms heterogenities
// of Tree to root h=1 and leaf h=0

// output functions to write KOAN files
void writeOneTypOFile(const char* filename);

void writeTypOHierachy(const char* rootFileName);

};


#endif // KLASSIIFIKATION_HPP
```

B.4 Definition der Deskriptoren DatenDeskriptor, NutzerDeskriptor und ProzessDeskriptor: „deskriptor.hpp“

```
// Datei "VektorArray.hpp"
// enthält die Definition der Klassen DatenDeskriptor,
// Nutzerdeskriptor, ProzessDeskriptor sowie die
```

```
// Definition zugehöriger Enums.

#ifndef DESKRIPTOR_HPP
#define DESKRIPTOR_HPP

/*******************************************************
*
*   Includes
*
*******************************************************/
#include "Precompiled.hpp"

#include "disjunktivKl.hpp"
#include "Klassifikation.hpp"
#include "AehnDistFuncs.hpp"
#include "HierachischeKl.hpp"

using namespace std;

/*******************************************************
*
*   enums
*
*******************************************************/
typedef enum { Aehnlichkeit, Distanz} ProximitaetsMass;

typedef enum { NullEinsTrafo, VarianzTrafo}
                                    StandardisierungsArt;

typedef enum {DisjunktClustern, NichtDisjunktClustern,
                        HierachischClustern} ClusterArt;

typedef enum {allEqual, hybrid} Equality;

typedef enum {rund, kette} KlassenForm;

/*******************************************************
*
*   Datendeskriptor
*
*******************************************************/
class DatenDeskriptor
{
public:
```

```
DatenDeskriptor();      // Konstruktor
~DatenDeskriptor();     // Destruktor


class ObjektDeskriptor // Beschreibung der Objekt-
                       // eigenschaften
{
public:
    bool bVerwendungAllerObjekte; // alle Objekte
                                  // verwenden?
    vector<ui> ZuVerwendendeObjekte;
        // Nicht alle verwenden ? -> welche sollen
        // verwendet werden?
} ObDesk;

class AttributEigenschaft // Beschreibung der Merk-
                          // malseigenschaften
{
public:
    SkalenTypen SkalenTyp;// Skalentyp des Merkmals
    bool bRelevanz;     // Soll Merkmal in die
                        // Kl. einbezogen werden?
    double Gewichtung;  // Wichtung des Merkmals
     bool bInfoLuecken; // fehlen Datenwerte?

    AttributEigenschaft() {}; // Konstruktor
};

class AttributDeskriptor // Zusammenfassung aller
                         // Merkmalseigenschaften
{
public:
    Equality equality;  // hybride Merkamle oder alle
                        // gleich?
    AttributEigenschaft allEqualType; // Alle Merkmale
                    // gleich -> Abspeicherung der
                    // Gesamtmerkmalseigenschaften

    map<ui, AttributEigenschaft> AttMap; // Map zur
                           // Speicherung der Merkmals-
                           // eigenschaften

    map<SkalenTypen, ui> TypAnzahlen; // Anzahlen der
```

```
                                // einzelnen Skalentypen

        ui getNrOfRelevantAttributes(); // Anzahl relevanter
                                        // Merkmale
    } AttrDesk;

    VectorArray* va; // zu den Metadaten zugehörige Objekt-
                     // Merkmals-Matrix
};

extern DatenDeskriptor datenDeskriptor; // globale Variable
                                // für den Zugriff auf den
                                // DatenDeskriptor

/*************************************************************
*
*   class ProzessDeskriptor
*
*************************************************************/
class NutzerDeskriptor
{
public:
    bool bAbsoluteDistanzen;    // -> nein: Profilverläufe
    ClusterArt clusterArt;      // hierarchisch, disjunkt
                                // oder nicht_disjunkt
    bool bAusreisserIdentifizieren; // soll das Verfahren
                                // Ausreißer kenntlich machen?
    bool bAusreisserEleminieren; // Sollen Außreißer in einem
                                // Präprozeßschritt eleminiert
                                // werden?
    KlassenForm klassenForm;    // welche Klassenform soll das
                                // Verfahren erkennen?

    vector<ui> EinzubeziehendeDimensionen; // Welche Merkmale
                                // sollen in die Klassifikation
                                // einbezogen werden
};

extern NutzerDeskriptor nutzerDeskriptor; // globale Variable
                                // für den Zugriff auf den
                                // NutzerDeskriptor
```

```
/************************************************************
*
*   class ProzessDeskriptor
*
************************************************************/
class ProzessDeskriptor
{
public:
    class StandardisierungsDeskriptor // Beschreibung der
                                      //Standardisierungsart
    {
    public:
        bool bStandardisieren;  // Soll eine Normierung
                                // durchgeführt werden?
         StandardisierungsArt StandardTyp;// Art der Normierung
        set<SkalenTypen> NichtZuStandardisierendeMerkmale;
                // Welche Mermale sollen ausgeschlossen werden?
                // z.B. nominale und ordinal Merkmale
        bool bEliminierenVonAusreissern; // sollen Ausreißer
                                         // eliminiert werden?
    } StandDesk;

    class AehnlichkeitsDistanzDeskriptor
    {
    public:
        ProximitaetsMass Type; // Distanz oder Ähnlichkeitsmaß?

        // Funktionen für die einzelnen Skalentypen
        ADFunktion nominalBinaerFunktion;
        ADFunktion nominalMehrstufenFunktion;
        ADFunktion ratioFunktion;

        double LrDistanzR; // Parameter für die Lr-Distanzen

  } AehnDistDesk;

    class GruppierungsAlgorithmus
    {
    public:
        ClusterArt AlgoGrundTyp; // hierarchisch, disjunkt
                                 // oder nicht_disjunkt

        ClusterungsAlgoFunktion ClusterungsAlgo;
           // Spezifikation des Klassifikations-Verfahrens
```

```
        ClusterungsParameter* Params;// Klassifikationparameter

    } GrupAlgo;
};


extern ProzessDeskriptor prozessDeskriptor; // globale Variable
                                // für den Zugriff auf den
                                // DatenDeskriptor

#endif //DESKRIPTOR_HPP
```

B.5 Definition der Pipeline-Klasse: „Pipeline.hpp“

```
// Datei "Pipeline.hpp"
// enthält die Definition der Klasse Pipeline zur
// Zusammenfassung von Hauptfunktionalitäten für
// die Pipelineausführung.

#ifndef PIPELINE_HPP
#define PIPELINE_HPP

#include "TriangleMatrix.hpp"
#include "klassifikation.hpp"
#include "disjunktivKl.hpp"
#include "FileIO.hpp"


class Pipeline
{
public:
static  void readData(FileIOParameters CalcParams,
                      bool verbose = false);
static  void readDeskriptors(FileIOParameters CalcParams,
                      bool verbose = false);

static  void calcProzessDeskriptor(bool verbose = false);

static  void executePipeline(HierachyTree&, bool verbose = false);

private:
static  void executePraeprozess(bool verbose = false);
```

```
static  TriangleMatrix* calculateProxyMatrix(bool verbose = false);

static  void calculateKlassifikation(HierachyTree&,
                        bool verbose = false);
};

#endif // PIPELINE_HPP
```

B.6 Die „Main“-Funktion - Ausführung der Pipeline im CommandLineTool

```
/*****************************************************************
*
* main - Funktion
*
* - lesen der Eingangsdaten (Nutzer-, Meta- und Rohdaten)
* - berechnen des Prozessdeskriptors
* - ausführen der Pipeline
* - Speichern der Ergebnisse
*
*****************************************************************/

void main(int argc, char *argv [])
{
// 0. Einlesen der exe-Argument-Parameter
FileIOParameters CalcParams;
    ProgramName = getProgramName (argv [0]);
CalcParams.ProgramName = strdup(ProgramName);

    process_args (argc, argv, CalcParams);

if (CalcParams.InPutFileName == string(""))  // kein InputFile definiert
{
cout << "Error 1: No input file defined!" << endl;
syntax();
};

if (CalcParams.OutPutFileName == string(""))  // keinOutPutFile definiert
{
cout << "Warning: No Outputfile defined!" << endl;
cout << "Taking input file name as output." << endl;
CalcParams.OutPutFileName = strdup(CalcParams.InPutFileName.c_str());
```

```
};

// 1. Einlesen der Daten und des Nutzer- und Datendeskriptors

Pipeline::readDeskriptors(CalcParams, CalcParams.verbose);
Pipeline::readData(CalcParams, CalcParams.verbose);

// 2. Erstellung des Prozessdeskriptors aus Daten- und Nutzerdeskriptor
//    - Festlegung der Verfahren und Maße für die Pipeline
// (bisher noch per Hand)

Pipeline::calcProzessDeskriptor();

// 3. Pripelineausführung

// 3.1. Erzeugen der Klassifikationsdatenstruktur
HierachyTree classification(datenDeskriptor.va);

// 3.2. Ausführung der Pipeline
Pipeline::executePipeline(classification, CalcParams.verbose);


// 4. Nachbearbeitungen

// 4.0. Schreiben eines Typ0-Files des Dendrogramms
if (CalcParams.verbose)
cout << endl <<"Writing dendrogram to typ0 file ..." << endl << endl;

string s(CalcParams.OutPutFileName);
s = s + string("Dend1.typ0");

classification.writeOneTyp0File(s.c_str());

// 4.1 Erzeugung eines n-naeren Baumes aus Dendrogramm
HierachyTree NTree(datenDeskriptor.va);
if (prozessDeskriptor.GrupAlgo.AlgoGrundTyp == HierachischClustern)
{
if (CalcParams.verbose)
cout << endl << "Creating n-ner tree from dendrogram ..." << endl;


classification.normTreeHeterogenities();

list<double> Thresholds;
```

```
if (CalcParams.ThresholdFileName != string(""))
try
{
FileIO::readThresholds(Thresholds, CalcParams.ThresholdFileName);
}
catch (FileIOException& e)
{
cout << e.getErrorString() << endl;
exit(1);
}
else
{
Thresholds.push_back(0.8);
Thresholds.push_back(0.6);
Thresholds.push_back(0.4);
Thresholds.push_back(0.2);
Thresholds.push_back(0.0);
}

NTree.buildHeterogenityHierachyFromDendrogram(classification, Thresholds);
}

// 6. Ausschreiben der Ergebnisse

if (prozessDeskriptor.GrupAlgo.AlgoGrundTyp == HierachischClustern)
{
// 6.1. Schreiben eines Typ0-Files des Dendrogramms
if (CalcParams.verbose)
cout << endl <<"Writing dendrogram to typ0 file ..." << endl << endl;

string s(CalcParams.OutPutFileName);
s = s + string("Dend2.typ0");

classification.writeOneTyp0File(s.c_str());

// 6.2. Schreiben eines Typ0-Files des n-naren Baumes
if (CalcParams.verbose)
cout << endl <<"Writing one hierachy typ0 file ..." << endl << endl;

s = string(CalcParams.OutPutFileName);
s = s + string("View.typ0");


NTree.writeOneTyp0File(s.c_str());
```

```
// 6.3. Schreiben einer Typ0-Hierachie
if (CalcParams.verbose)
cout << endl <<"Writing typ0 file hierachy ..." << endl << endl;

try
{
NTree.writeTyp0Hierachy(CalcParams.OutPutFileName.c_str());
}
catch (FileIOException& e)
{
cout << e.getErrorString() << endl;
}
}

if (CalcParams.verbose)
{
cout << "Clustering finished successfully, press a key, please." << endl;
getchar();
}
}
```