Clustering

Hauptseminar Machine Learning
Wintersemester 2003/2004

Can Önder

15. Januar 2004

Inhaltsverzeichnis

1 Einleitung

Grob gesagt versteht man unter Clustering die Einteilung von mehr oder weniger großen Datenmengen in verschiedene Gruppen bzw. Klassen.
Will man Daten clustern, dann hat man dazu ein riesiges Repertoire an verschiedenen zur Verfügung stehenden Verfahren, die man anwenden kann. Innerhalb dieser Verfahren gibt es dann wieder unterschiedliche und sehr vielfältige Algorithmen sowie Bestandteile von Algorithmen, zwischen denen man wählen kann. Somit ergeben sich insgesamt sehr vielfältige Möglichkeiten für das Clustering. Ganz grob unterscheidet man zwischen drei unterschiedlichen Verfahren:

- Partitionierendes Clustering
- Hierarchisches Clustering
- Wahrscheinlichkeitsbasiertes Clustering

Was genau versteht man aber nun unter einem Cluster?

Definition:
Ein Cluster ist eine Menge von Objekten, die untereinander eine hohe und zu anderen Objekten außerhalb des Clusters eine möglichst geringe Ähnlichkeit aufweisen.

Ziel von Clustering-Verfahren ist es auch oft, sog. *natürliche Gruppen* in den Daten zu finden. Dies bedeutet, man versucht „Wissen“ aus den Daten zu gewinnen, also etwas über ihre natürliche Struktur und über ihre spezifischen Eigenschaften herauszufinden oder diese zu entdecken.
Man unterscheidet dabei zwei verschiedene Vorgehensweisen:

1. Segmentierung

Unter Segmentierung versteht man eine *zweckmäßige* Unterteilung der Daten. „Zweckmäßig“ ist hierbei relativ zu sehen, da die Anforderungen je nach Anwendungsgebiet unterschiedlich sein können.

Beispiel 1:
Sie möchten TShirts für Männer herstellen. Sie überlegen sich, Ihre Zielgruppe in drei unterschiedliche Kategorien (Cluster) einzuteilen: dicke Männer, sehr kleine Männer und Männer von durchschnittlicher Statur. Anhand dieser Vorgaben können Sie nun Ihre TShirts hinsichtlich verschiedener Attribute wie z.B. Ärmellänge, Weite und Länge einteilen, und zwar derart, dass die TShirts mit den entsprechend passenden Eigenschaften die jeweiligen Zielgruppen so gut wie möglich abdecken.

Beispiel 2:
Sie sind Inhaber eines Lieferservices und möchten die verschiedenen Häuser einer Stadt mit Ihrer Ware beliefern. Sie können nun die Häuser bzgl. ihrer Lage in verschiedene, nah beieinander liegende Häuserblöcke einteilen, und diese dann beliefern lassen. Ein Mitarbeiter beliefert also die Häuser im Süden, ein anderer die Häuser im Osten, usw.
Auch dies wäre eine zweckmäßige Einteilung von Objekten (hier: Häuser).

2. Clusteranalyse

Anders sieht es bei der Clusteranalyse aus. Hierbei sollen die Daten nicht nach einem vorgegebenen Zweck unterteilt werden, sondern man interessiert sich für die natürlichen Strukturen, die den Daten zugrunde liegen (natürliche Klassen). Um obiges Beispiel mit dem Lieferservice weiterzuführen (Beispiel 2), so würde man hier beispielsweise nicht mehr nach Lage der Häuser differenzieren, sondern man würde die Häuser einteilen in kleine Häuser, Mehrfamilienhäuser und große Villen. Eine derartige Einteilung spiegelt dann vielmehr die natürlichen Gegebenheiten von Häusern wider. Man möchte also nach vorhandenen Eigenschaften und Strukturen in den Daten suchen, und nicht selten damit auch Statistiken aufstellen. Anwendung findet die Clusteranalyse in unzähligen Bereichen, unter anderem in den folgenden:

Medizin:

- Entdecken von Krankheitsbildern
- Zuordnung von Symptomen zu unterschiedlichen Krankheitsstadien

Biologie:

- Gehören auf den ersten Blick gleich aussehende Pflanzen oder Tiere vielleicht zu unterschiedlichen Spezies?

Marktsegmentierung & Kaufverhalten:

- demographische Unterschiede
- unterschiedliche Verhaltensweisen von Kreditkarteninhabern

2 Clustering-Verfahren

Im Folgenden sollen nun einige bekannte Clustering-Verfahren vorgestellt und erläutert werden.

2.1 Grundlagen

Hierzu ist es wichtig, sich vorerst mit einigen essentiellen Grundlagen und Hilfsmitteln vertraut zu machen, auf denen Clustering-Algorithmen basieren und aufbauen.

2.1.1 Abstandsfunktionen

Alle Verfahren und Algorithmen, die im Folgenden beschrieben werden, sind abhängig von einer Art von „Abstand“. In der oben aufgeführten Definition wird von Ähnlichkeit von Objekten gesprochen. Was aber bedeutet es, zu sagen, dass ein Objekt näher an einem Objekt dran ist als an einem anderen Objekt, bzw. dass es zu einem Objekt ähnlicher ist als zu einem anderen? Wie kann man diese Ähnlichkeit messen?
Um diese Frage zu beantworten, wird der Begriff der Abstandsfunktion oder Distanzfunktion eingeführt. Darunter versteht man eine mathematische Formel, mit deren Hilfe man den Abstand zweier Objekte zueinander messen kann. Es existieren auch hier sehr viele unterschiedliche Funktionen und Ansätze, die auch ständig verbessert und weiterentwickelt werden. Man muss auch je nach Anwendung unterscheiden. Vergleicht man lediglich Zahlen miteinander, so misst man deren Distanz anders als wenn man Farben oder Wörter zueinander in Bezug bringt.
Man unterscheidet demnach also zwischen sog. quantitativen/numerischen (Zahlen, Größen) und nicht-quantitativen/nominalen (Farbe, Geschlecht) Datenattributen.
Die Wahl der Abstandsfunktion ist in hohem Maße entscheidend für das Ergebnis des Clusterings. Meist ist sie sogar wichtiger als der Clustering-Algorithmus selbst. Einige Beispiele für verschiedene Abstandsfunktionen sind:

Euklidischer Abstand:

Der Abstand zweier Objekte berechnet sich aus der Wurzel der Summe der quadrierten Differenzen der jeweiligen Objektkomponenten:

$$dist(x,y) = \sqrt{(x_1 - y_1)^2 + \ldots + (x_n - y_n)^2} \quad = \quad \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \quad = \quad \|x_i - {y_i}^2\|$$

Abstandsfunktionen basierend auf Attributen:

Objekte können mehrere Attribute haben, z.B. kann eine Hose bezüglich Länge,

Qualität und Farbe kategorisiert werden. Im Folgenden sei x_{ij} die Darstellungsweise des Objektes x_i bezüglich seines j'ten Attributs. Der Unterschied zweier Objekte wird dann dargestellt als die Summe der Unterschiedlichkeiten ihrer Attribute, indem man über alle möglichen Attribute summiert. Es ergibt sich folgende Berechnungsformel:

$$D(x_i, x_{i'}) = \sum_{j=1}^{p} d_j(x_{ij}, x_{i'j}); \quad d_j(x_{ij}, x_{i'j}) = (x_{ij} - x_{i'j})^2$$

Möchte man die einzelnen Attribute gewichten, so erhält man:

$$D(x_i, x_{i'}) = \sum_{j=1}^{p} \omega_j \cdot d_j(x_{ij}, x_{i'j}); \quad \sum_{j=1}^{p} \omega_j = 1$$

Proximity Matrices:

Man kann nun alle paarweisen Objektabstände in einer N $\times$ N-Matrix speichern. Dabei stelle d_{ij} den Abstand von Objekt i zum Objekt j dar. Die Matrix ist symmetrisch, die Diagonalelemente sind 0 (Definition des Abstandes eines Objektes zu sich selbst). Die resultierende sog. Proximity Matrix dient den Cluster-Algorithmen dann als Input:

$$\mathbf{P} = \begin{pmatrix} 0 & d_{12} & \dots & d_{1N} \\ d_{21} & 0 & \dots & d_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ d_{N1} & \dots & \dots & 0 \end{pmatrix}$$

Nearest-Neighbor:

Möchte man den Abstand zwischen zwei Clustern messen, so betrachtet man den minimalen Abstand zweier Objekte/Punkte x, y, wobei x aus dem einen und y aus dem anderen Cluster stammt, und nimmt diesen Abstand repräsentativ als den Abstand der zwei Cluster her:

$$D(C_i, C_j) = \min\{d(x, y) | x \in C_i, y \in C_j\}$$

Furthest-Neighbor:

Die Furthest-Neighbor Methode macht es genau andersherum. Als Abstand zweier Cluster wird der maximale Abstand zweier Objekte/Punkte x, y aus den jeweiligen Clustern genommen. Zwei Cluster werden somit als nah zueinander angesehen, wenn alle Objekte in ihrer Vereinigung relativ nah zueinander sind:

$$D(C_i, C_j) = \max\{d(x, y) | x \in C_i, y \in C_j\}$$

Beispiel für nominale Attriute:

Hat man nominale Attribute, so kann man den Abstand messen, indem man die Anzahl der Attribute zählt, in denen sich zwei Objekte unterscheiden. Diese Anzahl wird dann als der Abstand der zwei Objekte angesehen:

$$dist(x,y) = \sum_{i=1}^{n} \delta(x_i, y_i); \quad \delta(x_i, y_i) = \begin{cases} 0 & \text{falls } (x_i = y_i) \\ 1 & \text{sonst} \end{cases}$$

Als weitere Abstandsfunktionen sind u.a. der Hamming-Abstand, Zentrumsmessungen zwischen Clustern und viele weitere bekannt.
Abstandsfunktionen müssen stets den folgenden drei Regeln gehorchen:

1. Nur positive Abstände:	$\forall x, y \in D : dist(x,y) \in \mathbf{R}^+$
2. Gleichheit:	$dist(x,y) = 0 \Leftrightarrow x = y$
3. Symmetrie:	$dist(x,y) = dist(y,x)$

2.1.2 Bewertungsfunktionen

Bevor man sich den verschiedenen Clustering-Verfahren zuwendet, muss man sich erst noch die sog. Bewertungsfunktionen ansehen. Jeder Algorithmus verwendet eine soche Bewertungsfunktion, um eine gute Partitionierung der Daten zu finden. Die Bewertungsfunktion drückt die Qualität eines Clusterings aus, d.h. sie liefert einen Wert, der eine Aussage darüber macht, ob eine gefundene Einteilung vorteilhaft ist oder nicht. Dazu wird die Bewertungsfunktion minimiert und nach einem Extremum gesucht. Da es jedoch keine eindeutige Lösung zum Optimieren der Bewertungsfunktion gibt, muss man systematisch nach möglichen Clustern im Datenraum suchen.
Beim partitionierenden Custering hat man in erster Linie zwei Anforderungen an die Cluster:

- Die Cluster sollen kompakt sein
- Die Cluster sollen untereinander so weit wie möglich voneinander entfernt sein

Man erhält damit zwei mathematische Formalismen, welche diese Kriterien ausdrücken.

Within Cluster Variation

Die sog. *Within cluster variation* $wc(C)$ dient als Maß für die Kompaktheit der Cluster. Das heißt, man schaut auf die Streuung der Objekte *innerhalb* eines Clusters. Ist die Streuung gering, so ist das Cluster kompakt. Hierfür wird zuerst für jedes Cluster ein repräsentatives Zentrum berechnet (arithmetisches Mittel aller Punkte im jeweiligen Cluster). Anschließend berechnet sich die Streuung aus der Summe Abstände der Punkte zu ihrem Clusterzentrum:

$$r_k = \frac{1}{n_k} \sum_{x \in C_k} x; \qquad Clusterzentrum$$

$$wc(C) = \sum_{k=1}^{K} wc(C_k) = \sum_{k=1}^{K} \sum_{x_i \in C_k} d(x, r_k)^2$$

Between Cluster Variation

Die sog. *Between cluster variation* $bc(C)$ dient als Maß für die Streuung der Cluster untereinander, d.h. ob die unterschiedlichen Cluster alle relativ nah beieinander liegen oder nicht. Dazu betrachtet man die Abstände der verschiedenen Clusterzentren untereinander:

$$bc(C) = \sum_{1 \leq j < k \leq K} d(r_j, r_k)^2, \qquad j \neq k$$

Als Bewertungsfunktion nimmt man meistens monotone Kombinationen von $wc(C)$ und $bc(C)$ wie z.B. $bc(C)/wc(C)$. Diese Bewertungsfunktion gibt dann eine Aussage über die Gesamtqualität des Clusterings.
Es gibt noch viele weitere Ansätze für Bewertungsfunktionen wie z.B. Kovarianz-Matrizen oder das sog. *minimum distance criterion*, welches für jedes Cluster den minimalen Abstand aller in ihm befindlichen Punkte bestimmt. Am Ende wird das Maximum all dieser Abstände als $wc(C)$ betrachtet.

2.2 Partitionierendes Clustering

Beim partitionierenden Clustering möchte man eine Menge von Daten in eine *vorgegebene* Anzahl von Clustern einteilen. Das bedeutet, wir haben eine Datenmenge $D = \{x_1, x_2, \ldots, x_n\}$, die wir in k disjunkte Partitionen einteilen möchten. Wir erhalten dann daraus eine resultierende Clustermenge $C = \{C_1, C_2, \ldots, C_k\}$ mit k Clustern. Dabei gehört jeder Punkt x_i in ein eindeutiges Cluster C_k. Jedes Cluster besteht aus mindestens einem Objekt, wobei jedes Objekt in höchstens einem Cluster enthalten sein darf (disjunkte Cluster).
Die Anzahl der Cluster muss im Voraus feststehen. Woher aber weiß man, welcher Wert von k optimal ist. Man könnte die Anzahl der Cluster k variieren, beobachten, wie sich die Bewertungsfunktion ändert, und sich dann für das beste Ergebnis entscheiden. Dies liefert jedoch kein aussagekräftiges Ergebnis, da der Wert der Bewertungsfunktion immer besser wird, je mehr Cluster man hat. Das lässt sich durch den Aspekt der Streuung erklären:
Im Extremfall haben wir so viele Cluster, dass jedes Objekt in seinem eigenen Cluster zu liegen kommt. Dann ist logischerweise die *within cluster variation* am geringsten, da das Objekt mit dem Zentrum übereinstimmt. Also liefert auch die Bewertungsfunktion einen guten Wert.

Man könnte auch kombinatorisch vorgehen. Man könnte alle möglichen Cluster-Einteilungen durchprobieren und sich am Ende für die beste entscheiden. Dies hat sich jedoch schon sehr bald als ineffizient herausgestellt, da wir exponentielle Laufzeit benötigen. Bei N Objekten und k Clustern bekommen wir $\binom{N}{k}$ Möglichkeiten. Für große N ist dies schon bald nicht mehr effizient brechenbar.

Die heute am weitesten verbreiteten Verfahren sind iterative Verfahren.
Dabei wird mit einem initialen Clustering begonnen. Anschließend werden die Objekte iterativ immer wieder neu zugeordnet, sodass die Bewertungsfunktion optimal ist, und das wiederum solange, bis das Clustering optimal ist.
Ein populärer Algorithmus, der dieses Verfahren implementiert, ist der sog. **K-means-Algorithmus**. Der k-means-Algorithmus ist bereits seit mehreren Jahrzehnten im Einsatz und wird unter anderem auch zur Bildkompression verwendet. Bereits im Jahre 1979 wurde das Buch *A K-Means Clustering Algorithm* von J.A. Hartigan und M.A. Wong verfasst.
Der Algorithmus arbeitet wie folgt:

1. K = Anzahl der Cluster wird festgelegt
2. Es werden zufällig k Clusterzentren ausgewählt
3. Jeder Punkt im Datenraum wird dem ihm am naheliegendsten Zentrum zugewiesen (verwende euklidische Abstandsfunktion)
4. Es werden die neuen Clusterzentren berechnet
5. Wiederhole solange iterativ bei 3), bis die Zentren stabil sind

In Pseudo-Code ausgedrückt:

FOR $k = 1, \ldots, K$ sei r_k ein zufälliger Punkt aus D;
WHILE es ergeben sich Änderungen im Cluster C_k **DO**
    Cluster bilden:
    FOR $k = 1, \ldots, K$ **DO**
        $C_k = \{x \in D | d(r_k, x) \leq d(r_j, x) \forall j = 1, \ldots, K; j \neq k\}$;
    END;
    Neue Clusterzentren bilden:
    FOR $k = 1, \ldots, K$ **DO**
        r_k = Mittel der Punkte im Cluster C_k;
    END;
END;

Der k-means-Algorithmus erzeugt disjunkte, runde, kompakte Cluster. Er ist einfach und intuitiv, was ihn so beliebt macht und der Grund dafür ist, das er in so vielen Gebieten Anwendung findet. Er liefert stets ein mindestens lokales Extremum und läuft mit Komplexität $O(KnI)$, wobei K die Anzahl der Cluster ist, n die Anzahl der Objekte und I die Anzahl der Iterationen.

Von Nachteil ist, dass man keine Aussage über das globale Extremum erhält, das heißt man weiß am Ende nicht, ob evtl. ein noch besseres Clustering möglich gewesen wäre. Außerdem ist das Ergebnis und die Qualität des Clusterings stark abhängig von k sowie von den anfangs zufällig gewählten Clusterzentren. Wird hier ungünstigerweise eine schlechte Startkonfiguration gewählt, so kann es passieren, dass ein gutes Clustering übersehen wird. Der Algorithmus funktioniert desweiteren nur für numerische Attribute und hat Probleme mit weit außen liegenden Punkten, sog. Ausreißern.
Deshalb lässt man den k-means-Algorithmus meistens mehrmals durchlaufen mit unterschiedlichen Startkonfigurationen, sodass man am Ende eine bessere Chance hat, ein globales Extremum zu finden.

Die Abbildung auf der nächsten Seite zeigt die Arbeitsweise des Algorithmus. Dabei zeigt Bild 1) die zu Beginn zufällig gewählten Clusterzentren. In Bild 2) sieht man bereits, wie sich die Zentren unter dem Einfluss der Datenpunkte auf die Datenansammlungen zubewegen, bis die Zentren schließlich in Bild 3) ziemlich gut in der Mitte der Datenhaufen zu liegen kommen und eine stabile Situation erreicht wird.

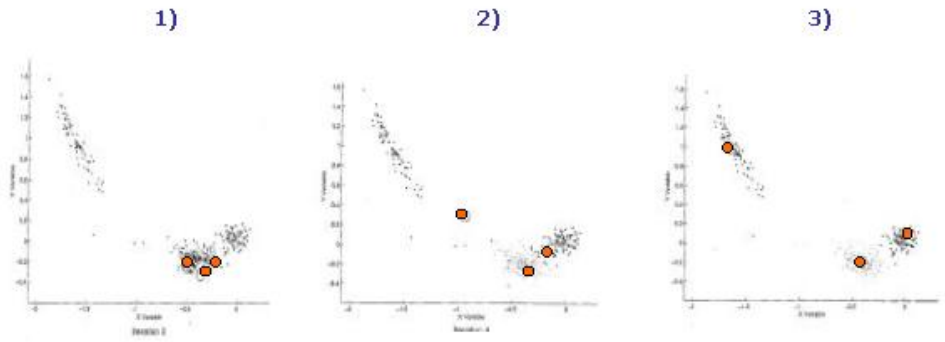

Abbildung 1: Arbeitsweise des k-means-Algorithmus

Der k-means-Algorithmus kann auch erweitert werden. Man kann beispielsweise auch überlappende Cluster zulassen, wenn man nicht auf Disjunktheit besteht. Ein Verfahren, welches dem k-means-Algorithmus sehr ähnlich ist, jedoch weniger Probleme mit „Ausreißern“ hat und auf beliebige Datenattribute anwendbar ist, ist das sog. *K-Menoids-Clustering*. Hierbei wird als Clusterzentrum nicht das arithmetische Mittel der Punkte im Cluster verwendet, sondern derjenige Punkt im Cluster, der diesem arithmetischen Mittel am nächsten ist. Dieser Punkt wird dann *Menoid* genannt. K-Menoids hat allerdings einen höheren Rechenaufwand als k-means. Ein anschauliches Java-Applet zum k-means-Algorithmus findet man unter
www.delft-cluster.nl/textminer/theory/kmeans/kmeans.html.

2.3 Hierarchisches Clustering

Im Gegensatz zum partitionierenden Clustering interessiert man sich beim hierarchischen Clustering in erster Linie dafür, *Clusterstrukturen* zu entdecken. Die Daten werden hierbei rekursiv unterteilt, d.h. man schaut sich die einzelnen Objekte an und mischt die beiden ähnlichsten zusammen in ein Cluster. Dieses Verfahren wird rekursiv solange fortgesetzt, bis man letztendlich eine sehr tiefreichende baumartige Struktur von entstehenden Clustern bekommt. Dabei umfasst die Wurzel des Baumes das gesamte Daten-Set. Die einzelnen inneren Knoten stellen die verschiedenen Gruppen bzw. Cluster dar, und an den Blättern befinden sich die einzelnen Objekte.
Ein Grund für die Beliebtheit hierarchischer Clustering-Verfahren ist die gute graphische Darstellungsmöglichkeit der entstehenden Baumstruktur mithilfe sog. *Dendogramme*.
Man unterscheidet beim hierarchischen Clustering zwei verschiedene Ansätze:

- Agglomeratives Clustering
- Divisives Clustering

Diese beiden Verfahren unterscheiden sich lediglich in der Art der Vorgehensweise. Agglomerative Methode arbeiten sich von den Blättern des Baumes bis zur Wurzel nach oben durch, während divisive Methoden an der Wurzel beginnen und weiterarbeiten, bis sie die Blätter erreicht haben. Genaueres wird hierzu weiter unten im Text erläutert.
Die nächste Abbildung zeigt ganz grob die Darstellungsweise eines Dendogramms:

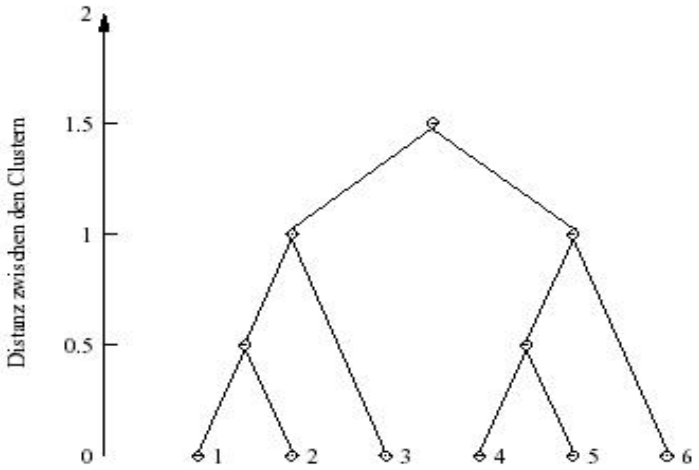

Abbildung 2: Dendogramm in einfacher Form

Beim hierarchischen Clustering gibt es keine globale Bewertungsfunktion. Vielmehr verwendet man vor jedem Mischen oder Teilen der Daten mehrere lokale *Scores* bzw. Berechnungsformeln, die Aufschluss darüber geben, welche Daten man gemeinsam in ein Cluster werfen sollte und welche nicht. Eine Variante einer solchen Berechnungsformel ist die sog. *Category Utility*, die später noch genauer erläutert wird. Beim hierarchischen Clustering wird die Anzahl der Cluster erst zur Laufzeit festgelegt. Die Unterschiedlichkeit der verschiedenen Cluster steigt mit höherem Level, wobei die Höhe eines Knotens angesehen werden kann als ein Maß für die interne Variation seiner Kinderknoten. Höher liegende Cluster bzw. Gruppen sind demnach auch bessere Kanditaten für *natürliche Gruppen*.
Man muss allerdings beachten, dass man Dendogramme nicht zwingend als eine 1:1-Abbildung der zugrundeliegenden Daten ansehen sollte. Dies liegt zum einen daran, dass abhängig von den verwendeten Methoden (z.B. agglomerativ oder divisiv) völlig unterschiedliche Dendogramme reultieren können, obwohl die Daten dieselben sind. Zum anderen ist die Entstehung der Baumstruktur durch den Algorithmus zwingend vorgegeben, d.h. man wird ganz unabhängig von den zugrundeliegenden Daten immer eine solche Baumstruktur erhalten, egal ob sie die tatsächliche Struktur der Daten widerspiegelt oder nicht. Eher ist das Resultat als eine Relation der $N(N-1)/2$ paarweisen Objektunterschiedlichkeiten anzusehen.

2.3.1 Agglomeratives Clustering

Das agglomerative Clustering stellt die sog. *Bottom-up*-Methode des hierarchischen Clusterings dar. Sie basiert auf der Abstandsmessung zwischen Clustern. Zu Beginn liegt jedes Objekt in seinem eigenen Cluster. Anschließend werden immer die zwei ähnlichsten Cluster zusammen in ein neues Cluster gemischt, welches dann ein Level höher im Baum zu liegen kommt. Damit wird auch die Anzahl der Cluster um eins reduziert. Bei diesem Mischvorgang kann es evtl. zu einer größeren Umstrukturierung des Baumes, ausgehend von der entsprechenden betroffenen Stelle, kommen. Die Entscheidung, welche Cluster zusammengemischt werden, kann z.B. mittels der bereits vorhin angesprochenen *Category Utility* erfolgen.
Agglomerative Methoden brechen ab, sobald sie an der Wurzel angelangt sind und demnach alle Cluster zu einem einzigen Cluster vermischt haben.
Wir haben also eine Menge von Daten $D = \{x_1, x_2, \ldots, x_n\}$ und eine gegebene Abstandsfunktion $D(C_i, C_j)$ für zwei Cluster C_i und C_j. Dann lautet ein agglomerativer Algorithmus wie folgt:

FOR $i = 1, \ldots, n$ sei $C_i = \{x_i\}$;
WHILE es gibt noch mehr als ein Cluster **DO**
    Seien C_i und C_j die Cluster mit geringstem Abstand $D(C_i, C_j)$ von allen Clusterpaaren;
    $C_i = C_i \cup C_j$;
    Entferne Cluster C_j;
END;

Folgende Abbildung zeigt nochmals ein Dendogramm, diesmal realistischer, resultierend aus einem tatsächlich an Daten durchgeführten Clustering. An den Blättern befinden sich die einzelnen Objekte, repräsentiert durch Zahlen.

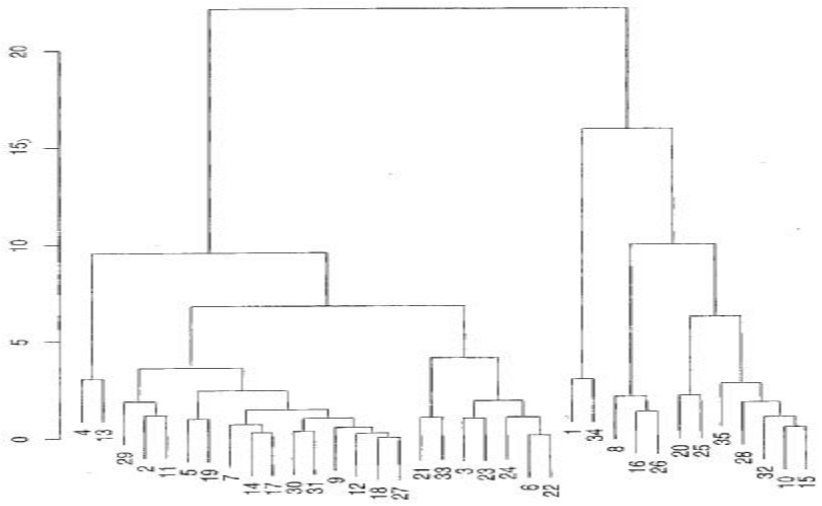

Abbildung 3: Dendogramm

Agglomerative Methoden verwenden als Abstandsfunktion oft die *Nearest Neighbor (single linkage)* -Methode:

$$D(C_i, C_j) = \min\{d(x, y) | x \in C_i, y \in C_j\}$$

Dabei entspricht also der Absand zwischen zwei Clustern dem Abstand der zwei sich am nächsten liegenden Punkten aus den zwei Clustern.
Ein Nachteil dieses Verfahrens ist, dass der Aspekt der Kompaktheit der Cluster verletzt werden kann. Es werden nämlich nur die nahest liegenden Punkte berücksichtigt, es wird jedoch keine Aussage über die Lage der anderen Punkte gemacht, was bedeutet, dass diese auch sehr versteut liegen könnten und damit die Clusterformen in die Länge ziehen. So haben auch weit außen liegende Punkte einen hohen Einfluss.

Alternativ kann die *Furthest Neighbor (complete linkage)* -Methode verwendet werden:

$$D(C_i, C_j) = \max\{d(x, y) | x \in C_i, y \in C_j\}$$

Als Maß wird also der größte Abstand zweier Punkte aus den jeweiligen Clustern genommen. Was allerdings hier passieren kann ist, dass Objekte, die im selben Cluster liegen, unterschiedlicher zueinander sind als zu Objekten, die ganz woanders im Baum platziert sind.
Um die Nachteile beider Methoden etwas auszugleichen, kann man einfach den Durchschnitt beider Verfahren bilden. Man nennt das dann auch *Group-Coverage (average linkage).*

Nachfolgende Abbildung zeigt den Unterschied der jeweiligen Methoden auf. Man erkennt deutlich die Unterschiedlichkeit der resultierenden Dendogramme, was auch noch einmal ein Indiz dafür ist, dass man das graphische Ergebnis nicht direkt auf die Stuktur der Daten projizieren sollte. Man hat nämlich dieselbe Datenmenge in allen drei Abbildungen, und man verwendet auch denselben agglomerativen Algorithmus. Lediglich die verwendete Abstandsfunktion ist unterschiedlich.

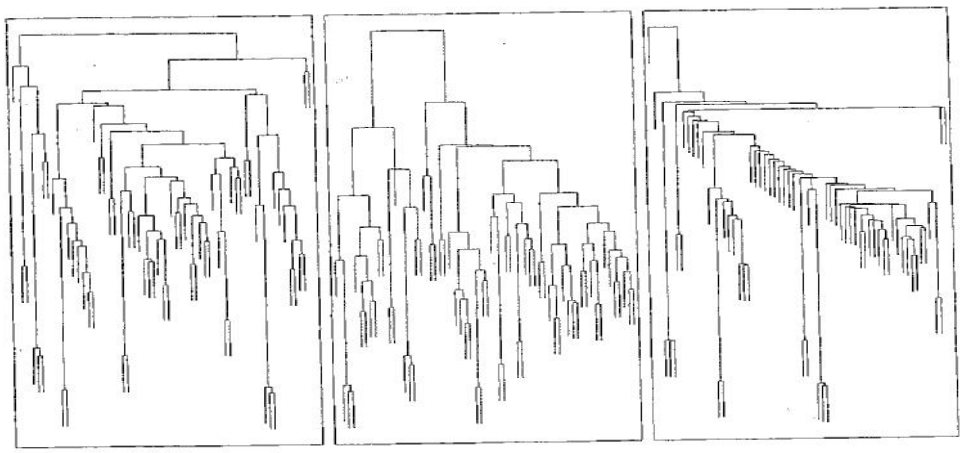

Abbildung 4: Dendogramme für Group-Coverage (links), Complete Linkage (Mitte) und Single Linkage (rechts)

2.3.2 Category Utility

Sehen wir uns nun die Berechnungsformel, die uns entscheidet, ob und welche Cluster gemischt werden, etwas genauer an. Die Category Utility berechnet den Wert der Wahrscheinlichkeit eines Attributes, dass dieses Attribut in ein bestimmtes Cluster gehört und stellt damit eine Qualitätsmessung für eine Unterteilung von Objekten/Instanzen in Cluster dar.
Sie wird wie folgt berechnet:

$$CU(C_1, \ldots, C_k) = \frac{\sum_l Pr[C_l] \sum_i \sum_j (Pr[a_i = v_{ij}|C_l]^2 - Pr[a_i = v_{ij}]^2)}{k}$$

Die a_i stellen dabei die verschienen Attribute dar, die v_{ij} sind die Werte, die diese Attribute annehmen können. Hat man beispielsweise ein Attribut Farbe, so könnten die belegenden Werte blau, grün und gelb sein. Bei der inneren Differenz wird die Wahrscheinlichkeit, dass ein Attribut einen bestimmten Wert annimmt subtrahiert von der Wahrscheinlichkeit, dass ein Attribut diesen Wert annimmt und zu einem bestimmten Cluster gehört. Der Wert dieser Differenz soll möglichst groß werden, damit man eine Aussage darüber treffen kann, ob eine gewisse Einteilung in Cluster sinnvoll ist oder nicht. Hätte man nur ein Cluster, so wäre die Wahrscheinlichkeit immer 1, und die Differenz wäre stets 0. Wir haben dann keine qualitative Aussage. Die innere Differenz ist sozusagen ein Maß dafür, in welchem Maße uns die Zusatzinformation, dass eine Instanz in einem bestimmten Cluster liegt, die Cluster also mit berücksichtigt werden, von Nutzen ist. Diese Wahrscheinlichkeiten werden dann in der inneren Doppelsumme einmal über alle Attribute und über alle möglichen Attributbelegungen summiert. Die äußere Summe rechnet dann noch die Clusterwahrscheinlichleiten mit ein und summiert über alle Cluster. Am Ende wird noch durch k dividiert, um sog. *Ovefitting* zu vermeiden. Overfitting bedeutet, dass man zu viele Cluster erhält. Die Formel ergibt nämlich immer dann den bestmöglichen Wert, wenn jedes Objekt in seinem eigenen Cluster liegt. Wir hätten somit keinen Anreiz, überhaupt Cluster zu bilden. Darum teilt man durch k, und kann dann auch wieder qualitative Unterscheidungen bzgl. bestimmter Einteilungen in Cluster treffen.
Obige Formel gilt für nominale Attribute, kann jedoch auch auf numerische Attribute ausgeweitet werden. Man nimmt dann für die Attribute Normalverteilung an und rechnet im kontinuierlichen Raum. Nach viel Rechnerei und Umformungen erhält man:

$$CU(C_1, \ldots, C_k) = \frac{1}{k} \sum_l Pr[C_l] \frac{1}{2\sqrt{\pi}} \sum_i (\frac{1}{\sigma_{il}} - \frac{1}{\sigma_i})$$

Um den Problemfall für $\sigma = 0$ abzufangen, führt man eine Mindest-Standardabweichung ein.

Eine ausführliche Herleitung der Gleichung findet man in der im Anhang angegebenen Literatur, insbesondere im *Data Mining*.

Im Folgenden soll anhand der Category Utility ein hierarchisches Verfahren beschrieben werden. Dabei betrachten wir nach und nach neue Objekte, und suchen für diese Objekte einen guten „Host“ oder „Parent“. Das ist eine Instanz, die laut Category Utility ein guter Kandidat ist, um mit der neuen Instanz in ein gemeinsames Cluster gemischt zu werden. Wird kein guter Host gefunden, so bleibt die Instanz in ihrem eigenen Cluster.

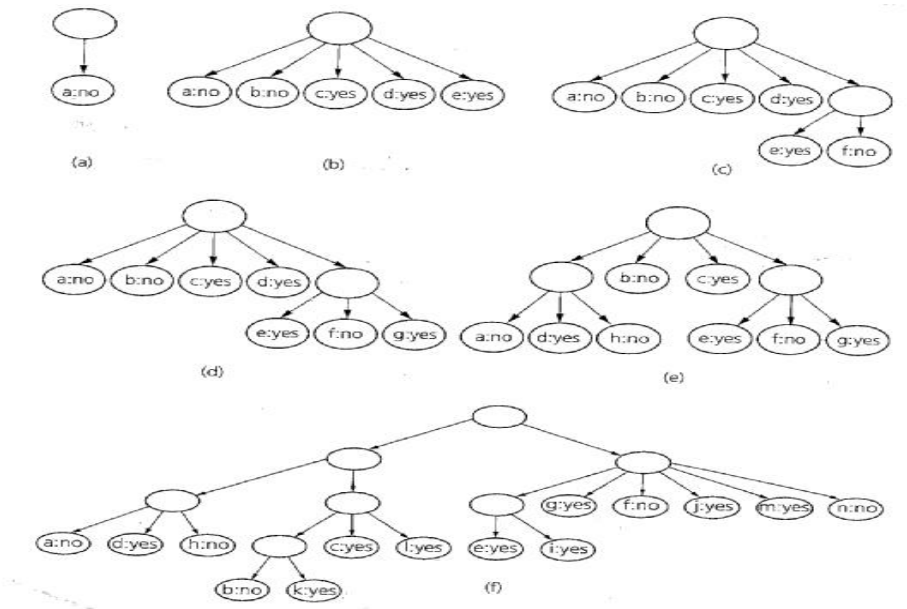

Abbildung 5: Hierarchisches Clustern mit der Category Utility

Zu Beginn betrachten wir Instanz a, welche in ihrem eigenen Cluster liegt (a). Im nächsten Schritt (b) kommen die Instanzen b, c, d und e hinzu. Die Auswertung der Category Utility ergibt, dass kein guter Host existiert, also bleiben alle Instanzen vorerst in ihrem eigenen Cluster. Anschließend (c) kommt Instanz f hinzu. Als passender Host findet sich Instanz e. Also wird ein neues gemeinsames Cluster für e und f gebildet. Danach (d) folgt Instanz g. Die Auswertung der Category Utility ergibt, dass das zuletzt neu gebildete Cluster ein guter Host für g ist. g wird also auch in das neue Cluster mitaufgenommen. Im nächsten Schritt (e) wird eine Umstrukturierung des Baumes nötig: der beste Host für Instanz h ist a, der zweitbeste ist d. Es stellt sich heraus, dass die Category Utility einen besseren Wert liefert, wenn man zuerst ein neues Cluster mit den Instanzen a und d bildet und anschließend Instanz$h$ addiert, als wenn man Instanz h direkt mit Instanz a in ein Cluster mischt. Folglich wird der Baum entsprechend neu strukturiert. Dieses Verfahren wird fortgesetzt, bis

das Dendogramm vollständig ist (f).

Die Komplexität (sowohl Zeit als auch Speicher) von agglomerativen Methoden beträgt $O(n^2)$, da alle paarweisen Abstände von Clustern überprüft (und gespeichert vorliegen) müssen.
Die Konsequenz davon ist, dass man für großes n einen sehr hohen Rechenaufwand zu verzeichnen hat. Außerdem wird die Baumstruktur der Dendogramme irgendwann sehr unübersichtlich, da ja am Ende alle Objekte an den Blättern in ihrem eigenen Cluster liegen.

2.3.3 Divisives Clustering

Das divisive oder auch teilende Clustering bildet die Umkehrung des agglomerativen Clusterings. Es handelt sich dabei um die *Top-down*-Variante, d.h. es wird oben an der Wurzel begonnen und es werden immer die zwei unterschiedlichsten Cluster gesucht, und anschließend geteilt.
Die Vorgehensweise ist wie folgt:

1. Starte mit dem obersten Cluster, welches alle Daten enthält (Top-Level-Cluster)
2. Suche anschließend nach dem Objekt mit dem größten Abstand zu allen anderen Objekten und bilde mit ihm ein neues, separates Cluster
3. Berechne dann für jedes verbliebene Objekt im ersten Cluster die Differenz aus dem durchschnittlichen Abstand zu allen Objekten im neuen Cluster und dem durchschnittlichen Abstand zu allen verbleibenden Objekten im ersten, ursprünglichen Cluster
4. Füge das Objekt mit dem dafür größten Ergebniswert dem neuen Cluster hinzu
5. Wiederhole diesen Vorgang so lange, bis die Differenz negativ wird, also so lange, bis es keine Objekte im ursprünglichen Cluster mehr gibt, die zu Objekten im neuen Cluster ähnlicher sind als zu Objekten im eigenen Cluster. Dann ist das neu gebildete Cluster fertig
6. Wende dieses rekursive *Splitting* auf beide Unter-Cluster an, und fahre so lange fort, bis jedes Objekt in einem eigenen Cluster liegt

Divisives Clustering ist insgesamt weniger weit verbreitet als agglomeratives Clustering, da es höheren Rechenaufwand erfordert.
Ein sehr großer Vorteil der hierarchischen Vorgehensweise ist, dass die Anzahl der Cluster im Voraus nicht feststehen muss, sondern während der Laufzeit automatisch bestimmt wird. Ein Hauptproblem der anderen Verfahren ist es nämlich, eine gute Wahl für diese Anzahl k zu treffen.

2.4 Wahrscheinlichkeitsbasiertes Clustering

Beim wahrscheinlichkeitsbasierten Clustering wird angenommen, dass die Objekte mit einer bestimmten Wahrscheinlichkeit zu einem Cluster gehören. Das bedeutet, die Daten kommen von einer Wahrscheinlichkeitsdichtefunktion, die den Daten zugrunde liegt. Die Grundlage dieser Verfahren stellen die sog. *Gaussian Mixture Models* dar. In diesem Fall ist eine Mixture als ein Satz von Wahrscheinlichkeiten anzusehen, die die k Cluster darstellen. Dabei hat jedes Cluster seine **eigene** Wahrscheinlichkeitsverteilung, wobei jede Instanz zu genau einem Cluster gehört. Damit man besser rechnen kann, wird oft die Gaussverteilung oder die Normalverteilung sowie Unabhängigkeiten der einzelnen Instanzen angenommen.

Beispiel:
Gegeben seien zwei Cluster A und B mit zugrundeliegender Normalverteilung und deren fünf unbekannten Parametern $\mu_A, \mu_B, \sigma_A, \sigma_B$ und p_A, welche die Erwartungswerte und die Standardabweichungen sowie die Wahrscheinlichkeit für Cluster A darstellen. Angenommen, man weiß, aus welchen Clustern die Instanzen stammen, dann lassen sich die Parameter sehr einfach wie folgt berechnen:

$$\mu = \frac{x_1 + x_2 + \ldots + x_n}{n}, \qquad \sigma^2 = \frac{(x_1 - \mu)^2 + (x_2 - \mu)^2 + \ldots + (x_n - \mu)^2}{n - 1}$$

Das heißt man berechnet den Erwartungswert durch das arithmetische Mittel aller Datenpunkte und dementsprechend auch die Varianz. Die Wahrscheinlichkeit p_A berechnet sich einfach aus dem Verhältnis der aus Cluster A stammenden Punkte (ist laut Annahme bekannt) und der Anzahl aller Punkte. Die Wahrscheinlichkeit für Cluster B, also p_B berechnet sich aus $1 - p_A$, da wir ja nur zwei Cluster haben.
Die nächste Abbildung zeigt die beiden Normalverteilungen der beiden Cluster A und B.

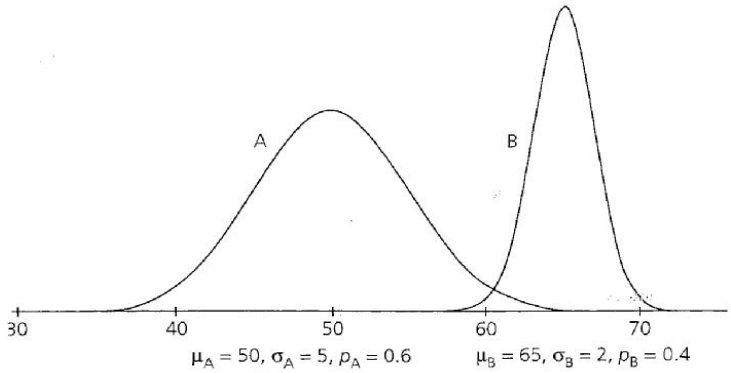

Abbildung 6: Mixture Model für $k = 2$ Cluster A, B

Wenn der entgegengesetzte Fall eintritt, man also die Parameter der Verteilung kennt, jedoch nicht die Cluster-Herkunft, so kann man die entsprehenden Wahrscheinlichkeiten wie folgt berechnen:

$$Pr[A|x] = \frac{Pr[x|A] \cdot Pr[A]}{Pr[x]} \quad = \quad \frac{f(x; \mu_A; \sigma_A) \cdot p_A}{Pr[x]} \quad (1)$$

mit

$$f(x; \mu; \sigma) = \frac{1}{\sqrt{2\pi\sigma}} \cdot e^{\frac{(x-\mu)^2}{2\sigma^2}} \quad (2)$$

Gleichung (1) berechnet also mithilfe der Formel von Bayes die Wahrscheinlichkeit, dass, hat man Instanz x gegeben, diese Instanz ins Cluster A gehört. $Pr[A]$ entspricht p_A, welche wir als gegeben vorausgesetzt haben. Der Ausdruck $Pr[x|A]$ kann mit der Normalverteilung für A abgeschätzt werden. Gleichung (2) stellt die Normalverteilung dar, deren Parameter μ und σ wir ebenfalls gegeben haben. $Pr[x]$ müssen wir nicht direkt berechnen. Wir werten einfach den Zähler von Gleichung (1) sowohl für Cluster A als auch für Cluster B aus und normalisieren, indem wir durch die Summe teilen, anstatt durch $Pr[x]$. Somit haben wir alle Parameter gegeben und können die Wahrscheinlichkeiten berechnen.

Meistens hat man jetzt aber das Problem, dass man weder die Verteilung samt ihrer Parameter noch die Cluster-Herkunft gegeben hat. In diesem Fall scheint das Problem unlösbar, jedoch kann man sich mit dem sog. *EM-Algorithmus* behelfen. Mit dem *EM-Algorithmus* kann man die Parameter der Verteilung nämlich schätzen und iterativ recht genau berechnen.
Der EM-Algorithmus durchläuft folgende Schritte:

1. Starte mit geschätzten Parametern für die Verteilung
2. Verwende diese Parameter zum Errechnen der Cluster-Wahrscheinlichkeiten (siehe Gleichung (1)) (E-Schritt)
3. Verwende wiederum die Wahrscheinlichkeiten, um erneut die Parameter zu berechnen (M-Schritt)
4. Wiederhole iterativ bei 2) usw...

Das klassische Werk über die erste detaillierte Beschreibung des EM-Algorithmus ist das Buch von A.P. Dempster, N.M. Laird und D.B. Rubin: *Maximum likelihood from incomplete data via the EM-algorithm*, wo weitere ausführlichere Informationen nachgelesen werden können.

Man muss erwähnen, dass man mit dem EM-Algorithmus nicht die definitive Cluster-Herkunft berechnet, sondern lediglich die Cluster-Wahrscheinlichkeiten.

Diese Wahrscheinlichkeiten verhalten sich jedoch wie Gewichte. Damit kann man dann wieder die Parameter berechnen.
Sei also ω_i die Wahrscheinlichkeit dafür, dass Instanz i zum Cluster A gehört. Dann berechnen sich Erwartungswert und Varianz wie folgt:

$$\mu_A = \frac{\omega_1 x_1 + \omega_2 x_2 + \ldots + \omega_n x_n}{\omega_1 + \omega_2 + \ldots + \omega_n} \quad (1)$$

und

$$\sigma_A^2 = \frac{\omega_1 (x_1 - \mu)^2 + \omega_2 (x_2 - \mu)^2 + \ldots + \omega_n (x_n - \mu)^2}{\omega_1 + \omega_2 + \ldots + \omega_n} \quad (2)$$

Man beachte, dass in Gleichung (1) nicht nur die Objekte berücksichtigt werden, die ins Cluster A gehören, sondern es wird über den gesamten Datenraum summiert. Gleiches gilt bei der Berechnung der Varianz (2). Gleichung (1) + (2) bilden den (M)-Schritt des EM-Algorithmus. Im (E)-Schritt werden dann wiederum aus den Parametern die Wahrscheinlichkeiten berechnet (wie oben bereits beschrieben).

Der EM-Algorithums konvergiert, ganauso wie der k-means-Algorithmus, gegen ein Maximum. Im Gegenteil zum k-means-Algorithmus allerdings erreicht der EM-Algorithmus dieses Maximum nie, sondern kommt ihm nur beliebig nah. Darum bricht man die Berechnungen nach „genügend“ vielen Iterationsschritten ab. Wie gut unsere Berechnung bereits ist, kann man mit **Maximum Likelihood** abschätzen. Diese Funktion nimmt mit jeder Iteration einen besseren Wert an.
Für genauere Informationen über Maximum Likelihood und dessen Funktionsweise sei an dieser Stelle auf das im Anhang angegebene *Data Mining* sowie auf Literatur von *Sir Ronald A. Fisher (1890-1962)*, dem Begründer von Maximum Likelihood und vieler verwandter Techniken im Bereich der Statistik, verwiesen.

Man beachte wiederum, dass auch der EM-Algorithmus ein lokales Maximum liefert.
Darum wird man auch hier mehrere Durchläufe vornehmen, bevor man sich dann für das beste Ergebnis entscheidet.

Führen wir nun unser Beispiel mit den zwei Clustern weiter. Möchte man nämlich mehr als zwei Cluster haben, wie es in der Realität auch meistens der Fall sein wird, so kann man das *Mixture Model*, also in diesem Fall unseren wahrscheinlichkeitsbasierten Algorithmus, auf beliebig viele Klassen ausweiten. Abgesehen von erheblich höherem Rechenaufwand und entsprechend komplizierteren Berechnungsformeln für die Parameter und die Wahrscheinlichkeiten handelt es hierbei jedoch um exakt dieselbe Basis-Methode. Diese wird einfach auf mehrere Klassen angewendet. Die Wahrscheinlichkeiten jedes Attributes werden dabei multipliziert, um die Gesamtwahrscheinlichkeit für eine Instanz

zu erhalten. Wenn man korrelierte Attribute hat, kann man keine Unabhängigkeit der Instanzen mehr annehmen. Mit n unabhängigen Attributen erhält man für jedes Attribut einen Erwartungswert und eine Standardabweichung. Mit n covarianten Attibuten dagegen erhält man $n + n(n + 1)/2$ Parameter und eine $n \times n$ Covarianzmatrix. Ein nominales Attribut mit v möglichen Werten wird als Vektor mit v Einträgen charakterisiert, welche die Wahrscheinlichkeiten darstellen.

Insgesamt kann die Arbeitsweise des wahrscheinlichkeitsbasierten Clusterings in folgenden Schritten beschrieben werden:

- Bestimme die Anzahl der Cluster k
- Wähle für jedes Cluster eine Wahrscheinlichkeitsverteilung
- Benutze den EM-Algorithmus, um die Verteilungsparameter und die Clusterwahrscheinlichkeiten zu schätzen

Wahrscheinlichkeitsbsiertes Clustering hat den Vorteil, dass es eine total verteilungsbasierte Beschreibung fur jede Komponente liefert. Da jedes Cluster seine eigene Verteilung besitzt, kann auch jedes Cluster eine ganz individuelle Form annehmen, d.h. man hat innerhalb eines Clusterings unterschiedlich kleine und kompakte, längliche und elliptische, zerstreute oder sehr große Cluster, was einen relativ großen Spielraum bietet. Eine gute Möglichkeit, die Probleme mit weit außerhalb liegenden Punkten loszuwerden, bietet die Einführung einer $k + 1$-ten Komponente als *Garbage Collector*. Das bedeutet, dass alle Objekte, die so weit außerhalb verstreut liegen, sodass sie keinem der gefundenen Cluster zugehörig zu sein scheinen, einfach in diese zusätzliche Komponente mit aufgenommen und damit quasi aus dem Verkehr gezogen werden.

Von Nachteil ist, dass das Modell auf Annahmen beruht, was die Verteilungen anbelangt. Man sollte Hintergrundwissen über die vorliegenden Daten haben, um mit einer gewissen Bestimmtheit sagen zu können, dass die getroffenen Annahmen auch halbwegs sinnvoll und realistisch sind. Anderenfalls kann man kaum mit qualitativ guten Ergebnissen rechnen. Ein weiterer Nachteil ist die hohe Komplexität der verwendeten Schätzalgorithmen wie z.B. der EM-Algorithmus.

Abbildung 7 zeigt ein paar Diagramme eines wahrscheinlichkeitsbasierten Clustering-Verfahrens mithilfe des EM-Algorithmus. Man kann sehr schön erkennen, dass die zwei entstehenden Cluster aufgund ihrer unterschiedlichen Verteilungen auch sehr unterschiedliche Formen haben.

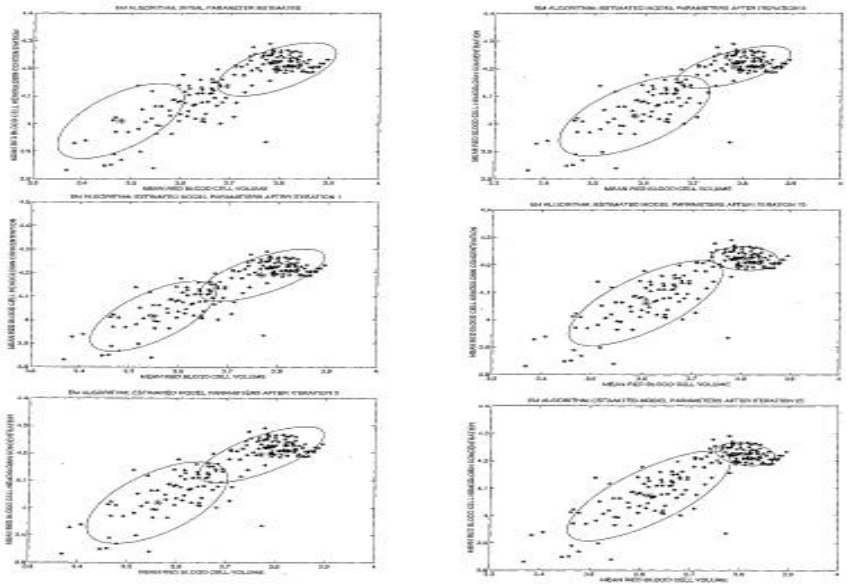

Abbildung 7: Wahrscheinlichkeitsbasiertes Clustering

Abschließend muss man noch erwähnen, dass, unabhängig von dem gewählten Clustering-Verfahren, sei es nun partitionierend mittels k-means-Algorithmus, hierarchisch oder wahrscheinlichkeitsbasiert, die Bewertung des Ergebnisses immer eine subjektive Sache ist. Es ist sehr schwer zu sagen, ob ein resultierendes Clustering gut oder schlecht ist, und wo die Grenzenn dazwischen liegen. Dies ist meist sehr anwendungsabhängig und muss vom Anwender selbst eingeschätzt werden. Es wird auch immer sehr schwierig bleiben, ein perfektes Ergebnis zu erhalten, da jedes Verfahren mit Annahmen der einen oder anderen Art arbeitet. Sei es die Annahme über die vermeintlich beste Wahl von k, also der Anzahl an Clustern, sei es die Annahme über die heirarchische Strukturierung der Cluster oder sei es die Annahme über ein Wahrscheinlichkeitsmodell, welches den Daten zugrunde liegt.

Literatur

[1] David Hand, Heikki Mannila, Padhraic Smyth:
Principles of Data Mining

[2] Trevor Hastie, Robert Tibshirani, Jerome Friedman:
The Elements of Statistical Learning

[3] *Data Mining - Practical Machine Learning Tools and Techniques with Java Implementations*