Seminar
Algorithmen zum Wirkstoffdesign

Clustering algorithms (Overview)

Matthias Zschunke

14. Dezember 2003

Zusammenfassung

Aufgaben wie Genomics, Proteomics oder auch die Suche nach neuen pharmakophoren Wirkstoffen produzieren riesige Datenmengen, die ausgewertet werden müssen. Diese Auswertungen umfassen die Analyse der Daten, sowie ihre Interpretation. Die Analyse großer Datenmengen wird häufig als *data mining* bezeichet. Geeignete *data mining*-Methoden helfen dabei, die Datenmengen zu reduzieren und damit *interpretierbar* zu machen.

Clustering gehört in die Gruppe der *data mining*-Methoden. Es umfasst eine Vielzahl von Verfahren, die die Daten durch Gruppieren auf die *relevanten* enthaltenen Informationen reduzieren und damit ein enthaltenes *Datenmodell* zum Vorschein bringen.

Inhaltsverzeichnis

1 Einführung

Für die Analyse großer Datenmengen benötigt man die Unterstützung geeigneter Algorithmen. Diese Verfahren dienen der Erforschung der Daten oder der Bestätigung von Annahmen, die man über die Daten gemacht hat. Das Ziel der Analyse ist es, ein zugrunde liegendes Modell für die Daten zu finden, das es einfacher macht, sie zu interpretieren.

Um dieses Modell zu finden, ist es notwendig, die Daten zu klassifizieren bzw. zu gruppieren. Klassifikation bedeutet Ähnlichkeiten und Unterschiede innerhalb der zugrunde liegenden Daten zu verdeutlichen. Sie ermöglicht das Erkennen von Mustern in den Daten. Anhand der Ähnlichkeit der Daten innerhalb gefundener Klassen und der Unterschiede zu Daten anderer Klassen lassen sich Rückschlüsse auf ihre Bedeutung ziehen.

Es gibt zwei wichtige Arten der Klassifikation: überwachte (*supervised*) und unüberwachte (*unsupervised*) Klassifikation. Während im ersten Fall die Muster vorgegeben sind und versucht wird, die Daten entsprechend diesem Muster zu gruppieren, versucht die unüberwachte Klassifikation, zu der Clustering gehört, die Daten in bedeutungsträchtige Gruppen einzuteilen. Das Muster bzw. die Bedeutung lassen sich dabei erst nach dem Klassifizieren erkennen.

Die Methoden zur Klassifikation von Daten aufgrund ihrer Ähnlichkeit bzw. Unähnlichkeit wird als Clustering bezeichnet.

„Clustering [is]... the process of organizing objects into groups whose members are similar in some way.“ [1]

Clustering-Algorithmen lassen sich in vielfältigster Weise einsetzen, so z.B. in Bioinformatik, medizinischen Diagnose-Verfahren ([2], [3]), statistischen Analysen, Computerlinguistik und Text Mining, Informationssuche in und Analyse von Datenbanken oder WWW, Computergraphik, Mustererkennung, Machine Learning, Neuronale Netzwerke, uva.

Clustering ist ein zentraler Prozess in der Datenanalyse. Es soll in den nächsten Abschnitten auf wichtige Aspekte der Datenanalyse eingegangen werden.

2 Daten

2.1 Daten und ihre Repräsentation

Als *Daten* bezeichnet man in diesem Zusammenhang eine Menge von Objekten, die untersucht werden sollen. Jedes Objekt hat bestimmte Eigenschaften, die Attribute, Merkmale oder Dimensionen genannt werden. Objekte des gleichen

Datenraums unterscheiden sich in der Ausprägung ihrer Merkmale, nicht aber in der Anzahl der Merkmale (Attribute, Dimensionen). Die Anzahl der Attribute hängt von den Daten ab und variiert von Problem zu Problem.

In der Literatur findet man häufig mathematische Bezeichnungen, die auch in dieser Ausarbeitung benutzt werden:

D, D_d: Datenraum (auch Merkmalsraum) der Dimension d (häufig $\mathbb{R}^d$)

S: Datenmenge, $S \subset D$

$\mathbf{x}_i$: i-tes Objekt aus S

$|S|$, n: Mächtigkeit von S, d.h. Anzahl der Objekte

$$\begin{array}{l} \\ \text{Objekte} \end{array} \left\{ \begin{array}{l} \overbrace{\qquad\qquad\qquad}^{\text{Attribute}} \\ \mathbf{x}_1 = (\quad x_{1,1} \quad \cdots \quad x_{1,d} \quad) \\ \mathbf{x}_2 = (\quad x_{2,1} \quad \cdots \quad x_{2,d} \quad) \\ \quad \vdots \qquad \vdots \quad \ddots \quad \vdots \\ \mathbf{x}_n = (\quad x_{n,1} \quad \cdots \quad x_{n,d} \quad) \end{array} \right.$$

Abbildung 1: Die Datenmenge als Matrix.

Jedes Objekt aus S hat d Attribute. Aufgrund der Art (des Typs) dieser Eigenschaften unterscheidet man *numerische* und *nicht-numerische* Daten. Für den ersten Fall bedeutet dies, dass die zu betrachtenden Objekte Vektoren von reellen Zahlen sind, z.B. Zeitreihen von durchgeführten Messungen [2]. Hier werden die Objekte auch häufig als Profile bezeichnet. Der zweite Fall umfasst den Rest, z.B. Datensätze aus einer Datenbank. Im Allgemeinen ist es allerdings möglich nicht-numerische Merkmale durch geeignete Codierung in numerische Merkmale zu transformieren, wobei jedes Merkmal selbst durch ein oder mehrere Attribute dargestellt wird. Im Folgenden wird deshalb nicht mehr zwischen numerisch und nicht-numerisch unterschieden.

Dieser erste Schritt ist notwendig um die Daten in ein verarbeitbares Format zu bringen, sodass Cluster-Algorithmen darauf angewendet werden können.

2.2 Dimensionsreduktion

Problematisch ist häufig die Anzahl der Attribute. Hierbei ist es so, dass einige der Attribute für ein bestimmtes Problem irrelevant, manchmal sogar irreführend

sind. D.h. man möchte bestimmte Merkmale ausklammern. Diese Reduzierung des Datenraumes auf weniger Dimensionen wird als Merkmalsauswahl (*feature selection*) bezeichnet. Merkmalsauswahl beschreibt den Prozess der Identifizierung der effektivsten Untermenge der Merkmale. Eine ähnliche Methode ist die Merkmalsextraktion (*feature extraction*), bei der durch bestimmte Transformationen einige Merkmale z.B. zusammengefasst werden. [8]

Versucht man z.B. Messreihen zu analysieren, benötigt man zusätzlich noch eine Methode zur Normalisierung der Daten. D.h. die Daten müssen so skaliert werden, dass sie dieselbe Varianz bzw. Standardabweichung (σ) und denselben Mittelwert ($\bar{\mathbf{x}}$) haben, meist $\bar{\mathbf{x}} = 0$ und $\sigma = 1$.

Diese Methoden führen zu einer geeigneten Menge von Merkmalen, die dann für den tatsächlichen Clustering-Algorithmus zur Verfügung stehen. Eine weitere Aufgabe der Merkmalsauswahl ist es, offensichtliche Muster innerhalb der Daten zu erkennen und die Repräsentation der Daten daraufhin anzupassen.

Abbildung 2 zeigt die Anordnung von Objekten um einen Mittelpunkt. Die durchschnittliche Entfernung zum Ursprung ist hierbei konstant.

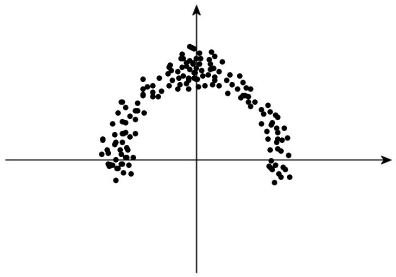

Abbildung 2: Kurvenförmiger Cluster. [8]

Wählt man für die Punkte kartesische Koordinaten, produzieren wahrscheinlich viele Clustering-Algorithmen mehrere Cluster, da die Punkte nicht *kompakt* angeordnet sind. Günstiger wäre hier die Wahl von Polarkoordinaten. (siehe Abbildung 3)

Man erkennt also, dass eine sorgfältige Analyse der möglichen Merkmale die Qualität des Ergebnisses der Klassifikation deutlich verbessert. *Bessere* Ergebnisse vereinfachen die am Ende der Datenanalyse stehende Interpretation. Ungeeignete Datenrepräsentationen führen häufig zu sehr komplexen und möglicherweise nicht interpretierbaren Clusterstrukturen. [8]

Verfahren mit denen man rein mathematisch die Reduzierung der Dimensionen vornehmen kann sind z.B. PCA (*principal components analysis, Hauptachsenanalyse* [5]) und MDS (*Multidimensionales Skalieren* [6], [7]). Beide Verfahren gehören in die Gruppe der *feature extraction*-Methoden.

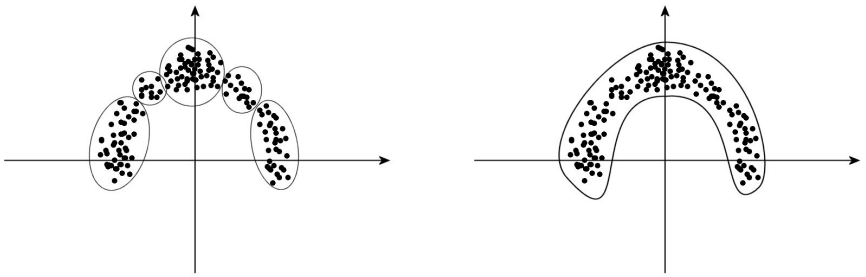

Abbildung 3: links: Annahme: kartesische Koordinaten. rechts: Annahme: Polarkoordinaten.

3 Ähnlichkeit und Unähnlichkeit

Um Objekte zu klassifizieren, muss man entscheiden können, ob zwei Objekte ähnlich zueinander sind oder nicht, bzw. benötigt man für die meisten Anwendungen sogar eine Aussage darüber, *wie* ähnlich sich die beiden Objekte sind. Unabhängig davon, ob man numerische oder nicht-numerische Objekte untersucht, braucht man ein Distanzmaß (*distance measure*) oder ein Ähnlichkeitsmaß (*similarity/proximity/affinity measure*). Das Distanzmaß definiert eine Funktion $d : D \times D \longmapsto Z$, die anhand der beiden zu vergleichenden Objekte einen Wert zurückliefert, der der Distanz (Unähnlichkeit) zwischen den Objekten entspricht. Das Ergebnis von d kann einerseits aus einer stetigen Menge (z.B. $\mathbb{R}$) oder aus einer diskreten Menge (z.B. $\{1, \cdots, m\}$) stammen. Es gibt verschiedene Möglichkeiten, Distanzmaße zu definieren.

Bei Daten, deren Merkmalsräume stetig sind, werden häufig metrische Distanzmaße eingesetzt, die folgende Eigenschaften aufweisen:

- $d(\mathbf{x}, \mathbf{y}) = d(\mathbf{y}, \mathbf{x})$
- $d(\mathbf{x}, \mathbf{y}) \geq 0, d(\mathbf{x}, \mathbf{y}) = 0 \Leftrightarrow \mathbf{x} = \mathbf{y}$
- $d(\mathbf{x}, \mathbf{z}) \leq d(\mathbf{x}, \mathbf{y}) + d(\mathbf{y}, \mathbf{z})$

  $\forall \mathbf{x}, \mathbf{y}, \mathbf{z} \in D$

Metrische Distanzmaße sind z.B.:

- euklidische Distanz: $d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{d} (x_i - y_i)^2}$
- Manhattan-Metrik: $d(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{d} |x_i - y_i|$
- Hamming-Distanz: $d(\mathbf{x}, \mathbf{y}) = \frac{1}{d} \sum_{i=1}^{d} \delta_i,\ \delta_i = \{^{0,\ falls\ x_i=y_i}_{1,\ sonst}$

Neben den metrischen Distanzmaßen lassen sich auch nicht-metrische zur Anwendung bringen, z.B.:

- Pearson'scher Korrelationskoeffizient: $d(\mathbf{x}, \mathbf{y}) = 1 - \rho(\mathbf{x}, \mathbf{y})$

$$\rho(\mathbf{x}, \mathbf{y}) = \frac{\sum_{i=1}^{d} (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{d} (x_i - \bar{x})^2 \sum_{i=1}^{d} (y_i - \bar{y})^2}}$$

- Rangkorrelationskoeffizenten
- *mutual information*-Distanz [8]

Das Distanzmaß ist eine der wichtigsten Entscheidungen, die man in Vorbereitung des Ausführens eines Clustering-Algorithmus treffen muss, da die berechneten Klassen in starkem Maße davon abhängen.

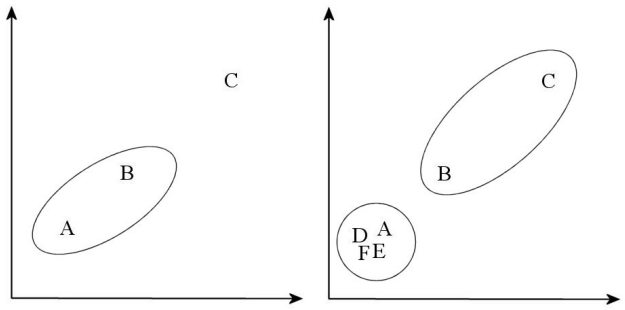

Abbildung 4: links: A und B sind ähnlicher bzgl. ihrer euklidischen Distanz. rechts: B und C sind ähnlicher als B und A bzgl. dieses Kontexts. [8]

Für bestimmte Probleme können Distanzmaße, die nur die beiden betrachteten Objekte berücksichtigen, ungeeignet sein (siehe Abbildung 4). Durch die Anwendung der euklidischen Distanz sind A und B ähnlicher als B und C. Im zweiten Fall aber kommt zusätzliche Information über nahegelegene Objekte zum tragen, sodass sich jetzt B und C ähnlicher sind als A und B. Das hier verwendete Maß wird als *mutual neighbor*-Distanz (MND) bezeichnet.

Für jedes Objekt $\mathbf{x}$ werden alle anderen Objekte entsprechend ihrer euklidischen Distanz zu $\mathbf{x}$ geordnet. $MND(\mathbf{x}, \mathbf{y}) = NN(\mathbf{x}, \mathbf{y}) + NN(\mathbf{y}, \mathbf{x})$, wobei $NN(\mathbf{y}, \mathbf{x})$ der Rang (*neighbor number*) von $\mathbf{y}$ bzgl. des Objektes $\mathbf{x}$ ist.

Da der Unterschied zwischen Distanz- und Ähnlichkeitsmaß nur in der Perspektive liegt (d.h. also große Distanz bedeutet zugleich kleine Ähnlichkeit, und umgekehrt, große Ähnlichkeit heißt kleine Distanz), wird im Folgenden nur zwischen beiden unterschieden, wenn dies unbedingt erforderlich ist.

4 Clustering

Hat man ein Distanzmaß gefunden, muss man sich für einen geeigneten Clustering-Algorithmus entscheiden.

Es werden im Großen und Ganzen zwei Gruppen von Verfahren unterschieden: *Hierarchisches Clustering* und *Partitioning*. Hierarchisches Clustering führt zu einer geschachtelten Struktur von Clustern, kleinere sind in größeren enthalten. Die kleinsten Cluster enthalten genau ein Objekt. Beim Partitioning entsteht nur eine Ebene von Clustern. Sie enthalten nur Objekte, keine Cluster.

Die Unterteilung der verschiedenen Verfahren lässt sich durch bestimmte Eigenschaften dieser Methoden noch verfeinern.

- *agglomerativ vs. divisiv*

  Die agglomerative Methode beginnt mit kleinen Clustern, die genau ein Objekt enthalten (*singeltons*). Jedes Objekt ist seinem *eigenen* Cluster zugeordnet. Anschließend werden die Cluster solange zusammengefügt, bis ein bestimmtes Abbruchkriterium erreicht ist.

  Die divisive Methode beginnt mit einem Cluster, der alle Objekte enthält. Schrittweise entstehen durch Zerteilen kleinere Cluster bis auch hier ein bestimmtes Abbruchkriterium erreicht ist.

- *monothetisch vs. polythetisch*

  Diese Unterscheidung bezieht sich auf die schrittweise oder gleichzeitige Verarbeitung der Attribute. Die meisten Algorithmen sind polythetisch, d.h. bei der Distanzberechnung zweier Objekte (oder zweier Cluster) werden alle Merkmale gleichzeitig berücksichtigt. Monothetische Algorithmen betrachten zunächst eines der Attribute, erstellen dementsprechende Cluster und betrachten anschließend das nächste Attribut, usw.

  Abbildung 5 zeigt ein Beispiel, bei dem zunächst nur das Attribut X_1 in die Clusterberechnung eingegangen ist. Es entstehen 2 Cluster links und rechts der Trennlinie V. Anschließend wird das Attribut X_2 betrachtet, wodurch

die beiden Cluster nochmal horizontal geteilt werden. Die berechneten Cluster werden durch die versetzten (!) Linien H_1 und H_2 getrennt.

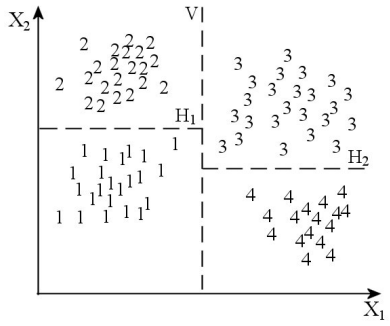

Abbildung 5: Monothetisches Partitioning. [8]

- *hard vs. fuzzy*

  Strikte (*hard*) Algorithmen ordnen jedem Objekt genau einen Cluster zu. Beim *fuzzy* (engl.: ungenau) Clustering kann jedes Objekt in mehreren Clustern enthalten sein. Dabei wird jedem Objekt ein Grad von Zugehörigkeit zu bestimmten Clustern zugewiesen.

- *incremental vs. non-incremental*

  Diese Unterscheidung ist sinnvoll, wenn die Datenmenge sehr groß ist, und Rechenzeit oder Speicherplatz zum Problem werden. *Non-incremental* Methoden versuchen die Anzahl der Durchsuchungen der Datenmenge, die Anzahl der Objekte oder die Größe der Datenstrukturen zu minimieren, die für den Algorithmus benötigt werden.

- *deterministisch vs. stochastisch*

  Stochastische Methoden lassen Zufallsverteilungen in die Berechnung einfließen. Bei *k-means* z.B. werden die Cluster durch ihre Mittelpunkte repräsentiert. Die Initialisierung der Mittelpunkte erfolgt zufällig. Über die Wahl der Mittelpunkte, die von der gewählten Zufallsverteilung abhängig ist, kann Einfluß auf die resultierenden Cluster genommen werden.

4.1 *k-means* Clustering

K-means ist einer der beliebtesten Partitioning Algorithmen. Seine Implementierung ist recht einfach und er liefert für einfache Anwendungen gut interpretierbare Ergebnisse.

Dieser Algorithmus versucht eine Partition der Datenmenge S in k Cluster zu finden. Dabei muss k vom Benutzer gewählt werden und verändert sich während der Berechnung nicht mehr. Die Cluster werden gebildet, indem ein bestimmtes Maß für die Homogenität der Cluster minimiert wird. Dies ist i.Allg. das sog. Sum-of-squares-Kriterium. Sei für jeden Cluster C_i $c(C_i)$ seine Kostenfunktion: Ziel ist es

$$c(C_i) = \sum_{r=1}^{|C_i|} \sum_{s=1}^{|C_i|} (d(\mathbf{x}_r^i, \mathbf{x}_s^i))^2$$

$\mathbf{x}_r^i$: das r-te Element des Clusters C_i

zu minimieren.

4.1.1 Algorithmus

1. Wähle zufällig k Cluster-Zentren $\mu_1, \ldots, \mu_k$.
2. Berechne für jedes $\mathbf{x} \in S$, zu welchem Clustermittelpunkt μ_i es am nächsten liegt.
3. Berechne für jeden Cluster C_i seine *Kosten*, d.h. berechne für jedes C_i die Kostenfunktion
$$c(C_i) = \sum_{r=1}^{|C_i|} (d(\mu_i, \mathbf{x}_r^i))^2$$
4. Berechne für jeden Cluster C_i seinen neuen Mittelpunkt μ_i:
$$\mu_i = \frac{1}{|C_i|} \sum_{r=1}^{|C_i|} \mathbf{x}_r^i$$
5. Wiederhole 2.), 3.) und 4.) solange, bis sich die $c(C_i)$ $(\forall i)$, bzw. die Clusterzuordnung nicht mehr ändert.

Nicht in allen Datenräumen ist der Mittelwert definiert, deshalb gibt es einen sehr ähnlichen Algorithmus, der anstelle der Mittelwerte die Mediane berechnet, genannt *k-medioids* bzw. *Partitioning Around Medioids* (*PAM*).

4.1.2 Vorteile

- Einfachheit.

4.1.3 Nachteil

- Neigung zu sphärischen Clustern.
- Filterung von *Noise*[1] nicht möglich.

4.1.4 Verbesserung

Für sehr große Datenmengen kann der Speicherplatz oder die Rechenzeit schnell zum Problem werden. Es gibt verschiedene Möglichkeiten *k-means* zu verbessern.

Mit der Datenstruktur *kd-tree* lassen sich die große Anzahl von Distanzberechnungen reduzieren und geeignetere Initialisierungen der k Clustermittelpunkte finden. [10]

Eine andere Verbesserung ist die effiziente Schätzung der Clusteranzahl. [11]

4.2 Hierarchisches Clustering

Hierarchisches Clustering wird angewandt, wenn es in der Datenmenge S keine offensichtliche Partition in wohlseparierte Cluster gibt. Dieses Verfahren erstellt einen Baum, dessen Knoten Teilmengen von S darstellen. S ist dabei die Wurzel des Baumes. Die einzelnen Objekte in S sind die Blätter. Innere Knoten stellen die Vereinigungsmenge der jeweiligen Kindknoten dar.

Hier wird grob zwischen zwei Verfahren unterschieden: *agglomerative* (*bottom-up*) und *divisive* (*top-down*) Algorithmen. Erstere gehen von einelementigen Clustern aus und verbinden diese stufenweise. Divisive Algorithmen erzeugen schrittweise kleinere Cluster durch Teilen größerer Cluster. Am häufigsten werden die agglomerativen Methoden eingesetzt.

4.2.1 Algorithmus

Agglomerative Methoden beginnen mit einer Zuordnung von je einem Cluster zu je einem Objekt und verbinden anschließend schrittweise diese Cluster zu größeren.

1. Beginne mit n Clustern $C_1, \ldots, C_n$; wobei $C_i = \mathbf{x}_i$.
2. Minimiere eine Kostenfunktion $c(C_i, C_j)$, um die *günstigste* Vereinigung $C_i \cup C_j$ zu finden.

[1] Als *Noise* werden Objekte aus S bezeichnet, die nur schwer mit anderen Objekten clustern.

3. Ersetze C_i und C_j durch $C_i \cup C_j$.

4. Wiederhole 2.) und 3.) bis alle Cluster zusammengefasst sind.

Der Hauptunterschied zwischen den verschiedenen Algorithmen, die diese Verfahren implementieren, ist die Kostenfunktion $c(C_i, C_j)$

Die häufigsten sind: ([1],[4])

- *Single-link* (Bsp.: SLINK)

$$c(C_i, C_j) = \min_{\mathbf{x} \in C_i, \mathbf{y} \in C_j} d(\mathbf{x}, \mathbf{y})$$

- *Average-link* (Bps.: UPGMA, Voorhees' method)

$$c(C_i, C_j) = \frac{1}{|C_i|\,|C_j|} \sum_{\mathbf{x} \in C_i} \sum_{\mathbf{y} \in C_j} d(\mathbf{x}, \mathbf{y})$$

- *Complete-link* (Bsp.: CLINK)

$$c(C_i, C_j) = \max_{\mathbf{x} \in C_i, \mathbf{y} \in C_j} d(\mathbf{x}, \mathbf{y})$$

Im Folgenden ist ein einfaches Beispiel von 7 Objekten angegeben.

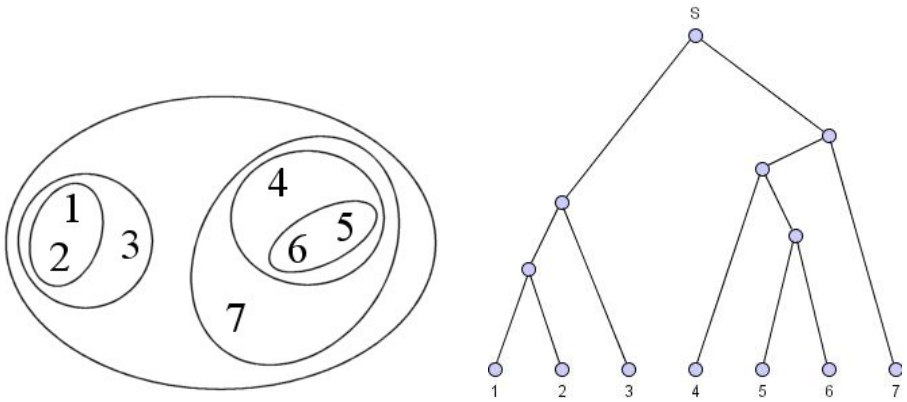

Abbildung 6: Hierarchisches Clustern.

Bei hierarchischen Clustering Algorithmen ist die Datenmatrix meist von geringerer Bedeutung, meist wird die Distanzmatrix als Eingabe verwendet. Die Distanzmatrix wird dabei in einem Schritt vor dem eigentlichen Clustering Algorithmus berechnet.

Die Distanzmatrix hat die Dimension $n \times n$, wodurch sie bei großen n schnell zum Speicherplatz-Problem werden kann. Deshalb gibt es einige Möglichkeiten, die Distanzmatrix zu reduzieren. Dabei werden z.B. einige Einträge, die kleiner als ein festzulegender Schwellenwert sind, einfach weggelassen. Oder es wird nur eine Menge von repräsentativen Elementen verwendet. Eine andere Möglichkeit ist es, mit jedem Objekt nur eine bestimmte Anzahl von nächsten Nachbarn zu assoziieren.

4.2.2 Vorteile

- Baumstruktur, d.h. keine Festlegung auf eine bestimmte Clusteranzahl.
- Flexibilität im Grad der Clusterung.
- Nutzbarkeit auf sämtlichen Datenräumen, vorausgesetzt es lässt sich ein Distanzmaß definieren.

  Anwendbarkeit auf Attribute beliebigen Typs.

4.2.3 Nachteile

- Tendenz zu sphärischen Clustern bei average- und complete-link.
- Möglicherweise unerwünscht längliche Cluster bei single-link.

4.3 Self-organizing Maps (*SOM*)

Beim SOM Clustering wird der d-dimensionale Eingaberaum auf ein 1- oder 2-dimensionales Gitter projiziert. Die Anzahl der Zeilen und Spalten im Gitter gibt dabei die Anzahl der Cluster wieder.

Die Anzahl der Cluster wird hierbei wieder vorgegeben. Die Knoten des Gitters werden als Referenzvektoren bezeichnet. Die Referenzvektoren werden während des Algorithmus in einem iterativen Prozess in Richtung der Eingabevektoren dirigiert.

Abbildung 7 zeigt das Prinzip des SOM Clustering anhand eines 3×2-Gitters. Die Gitterknoten sind von 1 bis 6 durchnummeriert. Die kleineren schwarzen Punkte stellen die Daten dar. Die Pfeile geben eine mögliche Richtung der Annäherung zu den Datenpunkten an.

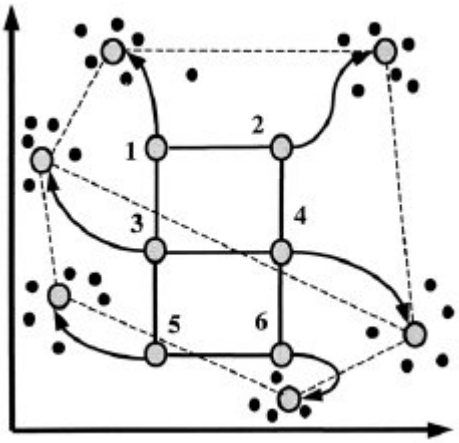

Abbildung 7: Prinzip des SOM-Algorithmus [9]

4.3.1 Algorithmus

1. Wähle ein Gitter mit $k = k_v \times k_h$ Knoten v, $v \in \{1, \ldots, k\}$.
2. Initialisiere k d-dimensionale Vektoren $f_0(v)$, durch zufällige Wahl von Objekten $\mathbf{x} \in S$ oder vollständig zufällig.
3. Iterationsschritte i:
   a) Für jedes Objekt $\mathbf{x} \in S$ bestimme den Knoten $v_\mathbf{x}$, für den $f_i(v_\mathbf{x})$ am nächsten zu $\mathbf{x}$ liegt.
   b) Aktualisiere alle Referenzvektoren wiefolgt:

$$f_{i+1}(v) = f_i(v) + \eta(d(v, v_x), i) \cdot (\mathbf{x} - f_i(v))$$

   $\eta(d, i)$: Lernrate. Die Lernrate nimmt mit der Distanz zwischen den Knoten $d(v, v_x)$ und der Iterationszahl i ab.

   c) Wiederhole a) und b) bis keine Veränderung mehr eintritt.

4.3.2 Beispiel: ALL/AML-Klassifikation

In [3] wird SOM zur Klassifikation von Leukemie-Krebszelltypen (ALL: akute lymphoplastische Leukemie, AML: akute myelitische Leukemie) eingesetzt. Das Problem beim klinischen Klassifizieren von Krebstypen liegt in der schwierigen Anwendung der klinischen Verfahren. Erfahrene Krebsspezialisten müssen anhand von Tumormorphologie, Histochemie, Immunodetektion und Zellanalysen

über den Krebstyp entscheiden. Die verschiedenen Schritte erfordern hoch spezialisierte Labore und sind sehr teuer.

Das in [3] verwendete Grundprinzip stützt sich auf die Analyse von DNA-Microarray-Daten. Aus Krebszellen wird die RNA extrahiert, mit Microarrays hybridisiert und die Expressionslevel werden gemessen. Die in einer Zelle vorhandene RNA spiegelt genau die Aktivität von Genen wider.

Das Ziel von [3] ist es, die Klassifikation von Krebstypen anhand der veränderten Genexpression in befallenen Zellen durchzuführen.

Es werden zwei Verfahren beschrieben: *class prediction* und *class discovery*. Ersteres soll Zellproben von Patienten in bereits gefundene Klassen einordnen. Zweiteres soll neue Klassen finden können.

Das Prinzip ist es, aufgrund einer Trainingsmenge von Knochenmark-Proben von 38 Patienten (27 ALL und 11 AML) einen *Schätzer* zu erstellen, der unbekannte Proben klassifizieren kann. Dabei wurde anhand der Expressionsprofile der Gene eine Korrelation zwischen den Klassen und *informativen* Genen gemessen. Getestet wurden 34 unabhängige Proben, bei denen 29 zu 100 % richtig klassifiziert wurden.

Die zweite Frage war, ob auch neue Klassen gefunden werden können. Auf die Expressionsprofile der verschiedenen Proben wurde SOM Clustering einmal mit 2 Clustern und dann mit 4 Clustern angewandt.

- 2 Cluster

  Die beiden Cluster entsprachen gerade der Unterteilung in AML und ALL

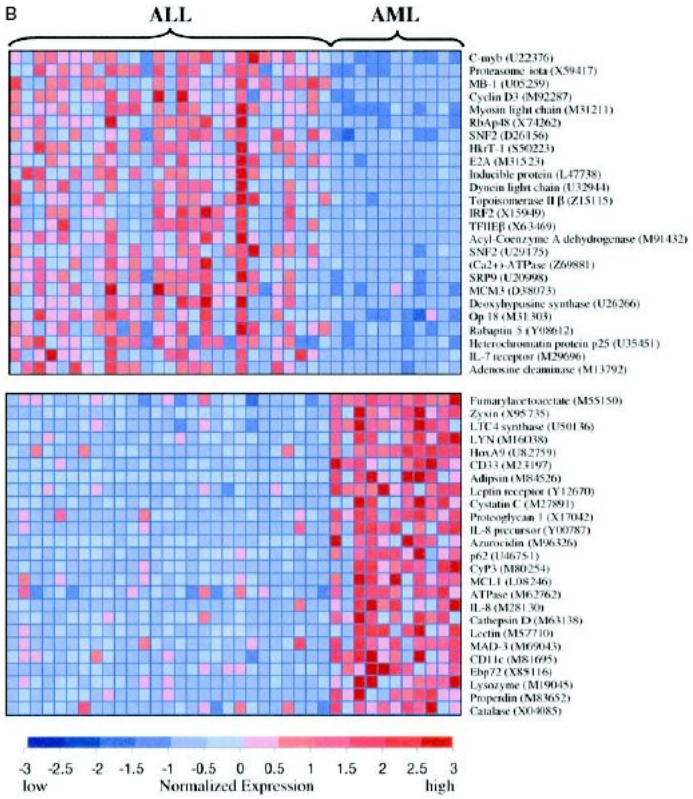

Abbildung 8: Heatmap der 50 *informativen* Gene und 38 Trainingsproben

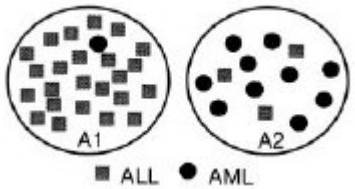

Abbildung 9: Gefundene Cluster mit 2×1-SOM

- 4 Cluster

  Hierbei wurden zwei weitere Klassen – Subtypen von ALL – gefunden. Dabei handelt es sich um die Unterscheidung von T-Zell- und B-Zell-ALL.

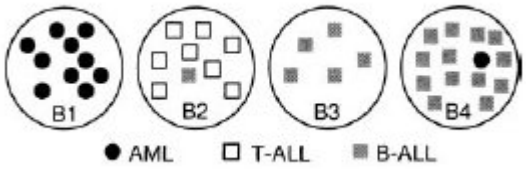

Abbildung 10: Gefundene Cluster mit 2×2-SOM

4.4 *Graph-based* Clustering

Graphen-basierte Clustering-Verfahren verstehen die Distanzmatrix als Graph, dessen Knoten den Objekten aus S entsprechen und dessen (*gewichtete*) Kanten die Distanz zwischen den Objekten darstellen.

Es wird versucht, Partitionen des so erhaltenen Graphen zu finden, die eine Clusterung der Objekte darstellen. Meist werden Bipartitionen des Graphen gebildet, die in einem rekursiven Prozess weiter geteilt werden. Die Graphen werden als Ähnlichkeitsgraphen bezeichnet.

4.4.1 *Clique-based* Clustering

Diese Verfahren stellen ihre Berechnungen aufgrund von Cliquengraphen an. Die grundlegende Idee ist, dass idealerweise alle Elemente eines Clusters ähnlich zueinander sind und unähnlich zu Elementen anderer Cluster. D.h. ein Ähnlichkeitsgraph sollte knotendisjunkte Cliquen enthalten. Dieser Graph wird Cliquengraph genannt. Jeder Knoten repräsentiert ein Objekt und jede Clique repräsentiert einen Cluster. Die Kanten in solch einem Graph geben also an, dass die Objekte an den inzidenten Knoten ähnlich zueinander sind. [1]

In der Praxis lassen sich die Ähnlichkeitsbeziehungen nur näherungsweise in Cliquengraphen ausdrücken, d.h. es können bestimmte Kanten fehlen, oder einige Kanten können zuviel sein.

Ein Modell ist das sog. *corrupted clique graph* Modell [1]. Dabei wird als Eingabe ein Cliquengraph angenommen, dessen Kanten mit Wahrscheinlichkeiten

gewichtet werden. Es wird angegeben, mit welcher Wahrscheinlichkeit eine Kante entfernt werden kann oder hinzugefügt werden kann.

Der Algorithmus versucht ausgehend vom *corrupted* Cliquengraph den orginalen Cliquengraph zu finden, der eindeutig die Cluster repräsentiert.

4.5 Spectral Clustering

Spektrale Clustering-Algorithmen nutzen die Eigenvektoren der Ähnlichkeitsmatrix um die Objekte zu partitionieren. [12], [13]

Die Anzahl der Cluster k muss hierbei vom Benutzer gewählt werden.

4.5.1 Algorithmus

Gegeben: Datenmenge $S = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ im $\mathbb{R}^d$; k: Clusteranzahl

1. Berechne Ähnlichkeitsmatrix $A_{n\times n}$

$$A_{ij} = \begin{cases} e^{\frac{-\left|\mathbf{x}_i - \mathbf{x}_j\right|^2}{2\sigma^2}}, & \text{falls } i \neq j \\ 0, & \text{sonst} \end{cases}$$

σ^2: Skalierungsfaktor

2. Berechne Diagonalmatrix $D_{n\times n}$

$$D_{ij} = \begin{cases} \sum_{l=1}^{n} A_{il}, & \text{falls } i = j \\ 0, & \text{sonst} \end{cases}$$

D.h. in der Diagonale von D stehen die Zeilensummen von A.

Berechne $L = D^{-1/2} A D^{-1/2}$.

$$D^{-1/2} := \begin{cases} \frac{1}{\sqrt{D_{ij}}}, & \text{falls } i = j \\ 0, & \text{sonst} \end{cases}$$

3. Finde $\mathbf{v}_1, \ldots, \mathbf{v}_k$, die k größten Eigenvektoren von L, so dass alle $\mathbf{v}_i$ paarweise orthogonal sind. Erstelle daraus die Matrix

$$X_{n\times k} = [\mathbf{v}_1, \ldots, \mathbf{v}_k] \in \mathbb{R}^{n\times k}$$

4. Konstruiere Matrix $Y_{n\times k}$ durch Normalisierung von X, z.B.:

$$Y_{ij} = \frac{X_{ij}}{\sqrt{\sum_j X_{ij}^2}}$$

5. Jede Zeile Y_i von Y ist ein Punkt im $\mathbb{R}^k$. Clustere diese Punkte mit einem beliebigen (am Besten *einfachen*) Clusteralgorithmus wie *k-means*.

6. Weise jedem Orginalpunkt $\mathbf{x}_i$ den Cluster j genau dann zu, wenn die Zeile Y_i im Cluster j liegt.

Die Frage, die sich stellt ist, warum nicht direkt *k-means* auf die Ausgangsdaten angewandt wird. Folgende Abbildung zeigt die Vorteile dieses Verfahrens.

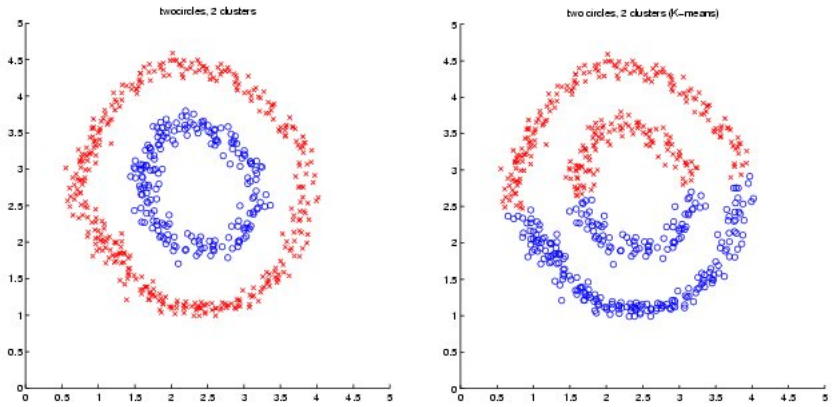

Abbildung 11: Vergleich von *spectral* (links) vs. *k-means*(rechts) Clustering [12]

K-means hat die Eigenschaft kompakte Cluster zu finden und versagt bei der Anwendung auf diese Daten, deren Punkte in konzentrischen Kreisen angeordnet sind.

4.5.2 Vorteile

- Clustering unabhängig von bestimmter Form (*shape*).
- Einfache Implementierung mit Statistik-Sprachen wie S-Plus, Matlab oder R.

4.5.3 Nachteile

- Wahl der Clusterzahl k.

4.5.4 Weitere Beispiele

Auch andere Daten unterschiedlichster Strukturen konnten mit dem beschriebenen Algorithmus geclustert werden.

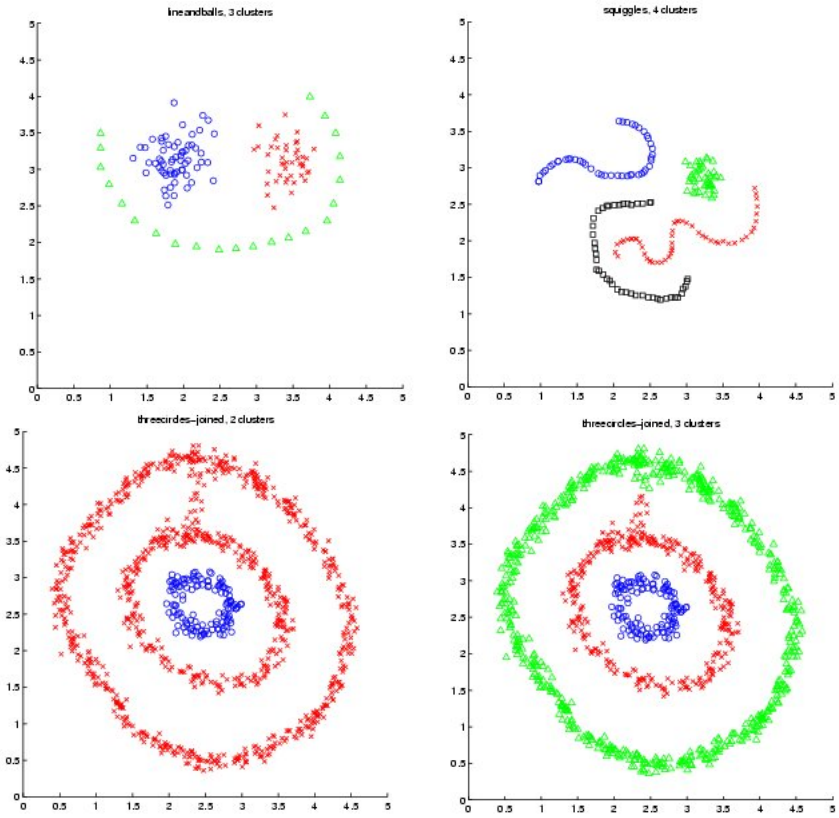

Abbildung 12: Clustering von verschiedenen Strukturen [12]

5 Clustering-Ergebnisse

5.1 Bewertung der Ergebnisse

Wenn ein Clusteralgorithmus eine Menge von Clustern ausgibt, muss ein Maß gefunden werden, um die Güte der Cluster zu bestimmen.

Möglichkeiten für die Bewertung sind z.B.:

- Minimale Varianz

$$q(C) = \sum_i q_i$$

$q(C)$: Qualität der Clustermenge $C = C_1, \ldots, C_k$
q_i: Qualität des Clusters C_i, $q_i = \sum_{\mathbf{x} \in C_i} d(\mathbf{x}, \mu_i)$
μ_i: Mittelpunkt des Clusters C_i

- Dichte der Cluster

$$q(C) = \sum_{i=1}^{k} \frac{1}{|C_i|} \sum_{\mathbf{x} \in C_i} d(\mathbf{x}, \mu_i)$$

- Homogenität H
  - mittlere: $H = \frac{1}{n} \sum_{\mathbf{x} \in S} d(\mathbf{x}, \mu_{C(\mathbf{x})})$
  - maximale: $H = \max_{\mathbf{x} \in S} d(\mathbf{x}, \mu_{C(\mathbf{x})})$
- Trennung T
  - mittler Trennung: $T = \frac{1}{\sum_{i \neq j} |C_i||C_j|} \sum_{i \neq j} |C_i|\,|C_j|\, d(\mu_i, \mu_j)$
  - minimale Trennung: $T = \min_{i \neq j} d(\mu_i, \mu_j)$

5.2 Visualisierung

Für die Interpretation der Ergebnisse ist es hilfreich, eine geeignete Form der Darstellung der Ergebnisse zu finden.

Daten von Meßreihen, wie z.B. Expressionsprofile von Microarray-Experimenten (siehe Abschnitt 4.3.2), lassen sich durch sog. Heatmaps darstellen.

Werden hierarchische Clustering-Methoden angewandt, kann die resultierende Clusterhierarchie als Dendrogramm dargestellt werden (siehe Abschnitt 4.2.1). Das folgende Dendrogramm gibt die Beispiel-Clusterung der 7 Objekte an. Das Dendrogramm hat, im Gegensatz zur Baumdarstellung in Abschnitt 4.2.1, den Vorteil, dass die Distanzen der Punkte zu den anderen Punkten im jeweiligen Cluster abgelesen werden kann.

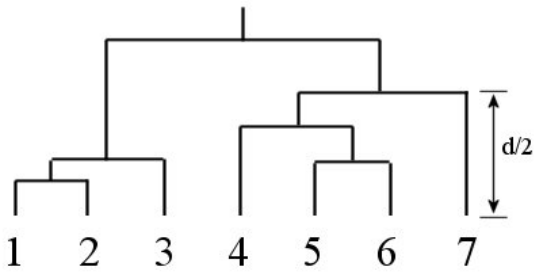

Abbildung 13: Dendrogramm zum Beispiel aus Abschnitt 4.2.1.

6 Zusammenfassung

In dieser Ausarbeitung wurden Clusteralgorithmen als Methoden unüberwachter Klassifikation vorgestellt. Clustering-Verfahren spielen eine zentrale Rolle bei der Datenanalyse verschiedenster Fachgebiete. Sie helfen sehr große Datenmengen zu analysieren und deren Informationsgehalt zu filtern und zu interpretieren.

Es existieren verschiedene Möglichkeiten das Problem des Klassifizierens zu lösen. Wichtige, und häufig angewandte, Verfahren sind *k-means*-Clustering und hierarchisches Clustern.

Die Güte der Cluster und ihre Interpretierbarkeit hängen sehr stark von verschiedenen Parametern, wie z.B. Distanzmaß, Clusteranzahl und vor allem Clustering-Algorithmus, ab.

Bei der Wahl der Algorithmen müssen bestimmte Überlegungen getroffen werden, z.B.: Lässt sich auf den Daten ein Distanzmaß definieren? Kann man die Daten sinnvoll als Vektoren darstellen? Kann der Algorithmus mit der Dimension und der Anzahl der Objekte effizient umgehen? Werden Ausreißer in den Daten berücksichtigt? Wie werden die Ergebnisse präsentiert?

6.1 Ausblick

Clusteralgorithmen liefern gute Ergebnisse, wenn sie mit *geeigneten* Daten und *geeigneten* Parametern arbeiten. Die Wahl dieser Parameter ist meist eine große Herausforderung und resultiert aus Annahmen, die man über die Daten machen muss.

Mögliche zukünftige Arbeiten müssen sich daher mit der Eliminierung von *a priori* Annahmen, wie z.B. fest gewählte numerische Eingaben, beschäftigen. Nützlich wären auch intelligente Parametersuchen. Eine gute Lösung wäre deshalb ein Al-

gorithmus, der als Eingabe ausschließlich die zu clusternden Objekte nutzt, und anhand dieser Daten selbstständig entscheiden kann, welche Parameter für ein gutes Clustering gewählt werden müssen.

Ein weiteres großes Problem ist die Effizienz der Algorithmen. So können z.B. Verfahren durch effiziente Aktualisierung der Clusterzuordnungen in ihrer Laufzeit verbessert werden. [1]

Literatur

[1] Daniel Fasulo. *An Analysis of Recent Work on Clustering Algorithms*. 1999

[2] Javed Kahn, Jun S. Wei, Markus Ringner, Lao H. Saal, Marc Ladanyi, Frank Westermann, Frank Berthold, Manfred Schwab, Cristina R. Antonescu, Carsten Peterson & Paul S.Meltzer. *Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks*. Nature Medicine. 2001

[3] T. R. Golub, D. K. Solnim, P. Tamayo, C. Huard, et al. *Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring*. Sience Vol. 286. 1999

[4] K. Nieselt-Struwe. Skript zur Vorlesung *Microarrays und Genexpression*. Uni Tübingen, SS 2003

[5] J. Quackenbush. *Computational analysis of microarraydata*. Nat Rev Genet, 6: 418-427, 2001

[6] J. B. Kruskal, M. Wish. *Multidimensional Scaling*. Sage University Paper series on Quantitative Applications in the Social Sciences, number 07-011. Sage Publications, Newbury Park, CA., 1978

[7] J. B. Kruskal, `http://www.cis.hut.fi/~sami/thesis/node15.html`

[8] A. K . Jain, M. N. Murty, P. J. Flynn. *Data Clustering: A Review*. ACM Computing Surveys, Vol. 31, No. 3, 1999

[9] P. Tamayo, D. Slonim, J. Mesirov, Q. Zhu, S. Kitareewan, E. Dimitrovsky, E.S. Lander, and T.R. Golub. *Interpreting patterns of gene expression with self-organizing maps: methods an application to hemotopoietic differentiation*. Proc Natl Acad Sci USA, 96:2907-2912, 1999

[10] D. Pelleg, A. Moore. *Accelerating Exact k-means Algorithms with Geometric Reasoning*. School of Computer Science, Carnegie Mellon University, Pittsburgh. 1999

[11] D. Pelleg, A. Moore. *X-means: Extending K-means with Efficient Estimation of the Number of Clusters*. School of Computer Science, Carnegie Mellon University, Pittsburgh. 2000

[12] A. Y. Ng, M. I. Jordan, Yair Weiss. *On Spectral Clustering: Analysis and an algorithm*. In NIPS 14. 2002

[13] C. J. Alpert, S. Yao. *Spectral Partitioning: The More Eigenvectors, The Better*. ACM 0-89791-756-1/95/0006. 1995