INSTITUT FÜR INFORMATIK
Datenbanken und Informationssysteme

Universitätsstr. 1 D–40225 Düsseldorf

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

Verfahren zur Dimensionsreduktion

Philipp Helo Rehs

Bachelorarbeit

Beginn der Arbeit:	21. Mai 2014
Abgabe der Arbeit:	21. August 2014
Gutachter:	Prof. Dr. Stefan Conrad
	Prof. Dr. Michael Leuschel

Erklärung

Hiermit versichere ich, dass ich diese Bachelorarbeit selbstständig verfasst habe. Ich habe dazu keine anderen als die angegebenen Quellen und Hilfsmittel verwendet.

Düsseldorf, den 21. August 2014

_____

Philipp Helo Rehs

Zusammenfassung

Im Rahmen des Social Engineering und anderen Bereichen der Datenverarbeitung fallen immer größer werdende Datenmengen an. Diese Daten werden im Prozess des Data-Mining ausgewertet und für Geschäftsprozesse, Marktanalysen und Informationsgewinnung genutzt. Jedoch lassen sich Datenmengen im Bereich von mehreren Terabyte schlecht verarbeiten, da viele Systeme dafür nicht geeignet sind solch große Datenmengen im Arbeitsspeicher zu behalten oder da die Algorithmen für solche Daten nicht ausgelegt sind. Zudem kommt es bei hochdimensionalen Daten, also Daten mit vielen Attributen, zum Fluch der Dimensionalität. Dies bedeutet, dass die Unterschiede zwischen den Daten immer geringer werden, sodass man diese schwer auseinander halten kann.

Um diesem Problem entgegenzuwirken wurden Verfahren zur Dimensionsreduktion entwickelt, welche in der Lage sind Attribute ohne Informationsverlust zu entfernen. Einige dieser Verfahren wählen die Attribute mit der höchsten Varianz zur Weiterverarbeitung aus. Andere hingegen versuchen die Daten durch Kombination der Attribute in niedrigere Dimensionen zu überführen. Jedoch ist davon auszugehen, dass alle diese Verfahren gewisse Vor- und Nachteile besitzen.

In dieser Bachelorarbeit wird versucht diese Vor- und Nachteile zu untersuchen indem die Algorithmen auf verschiedenste Datensätze angewendet werden. Danach wurden die reduzierten Daten mit k-Means und DBSCAN geclustert und die Ergebnisse mit Hilfe von mehreren Clustering-Bewertungsverfahren analysiert. Dabei wurde eine Unterscheidung zwischen generierten Daten mit bestimmten Eigenschaften wie sich überlappende und nicht überlappende Clustern, sowie Datensätzen aus anderen Forschungsarbeiten getroffen.

Um die Clusteringergebnisse zu bewerten werden mehrere Verfahren wie der Silhouettenkoeffizient und das F_1-Maß genutzt. Dabei bekommt man eine eine Bewertung zwischen 0 und 1, je nachdem ob das Clustering gut zu den Daten passt.

Die Auswertung der Ergebnisse von allen Datensätzen zeigt, dass sich besonders die Hauptkomponentenanalyse (PCA) eignet um eine Dimensionsreduktion durchzuführen,weil diese im Großteil der Fälle ein bessere Ergebnis liefert. Dies ist durch die Vorgehensweise von PCA zu erklären, da die Daten in eine neue Basis transformiert wird, welche genau den Richtungen der größten Varianzen entsprechen.

Inhaltsverzeichnis

1 Einleitung

1.1 Was ist Data-Mining?

Das Data-Mining (dt. Datengewinnung) ist ein Verfahren bei dem mit Hilfe von statistischen Verfahren versucht wird in großen Datenbeständen neue Erkenntnisse und Modelle zu entdecken. Data-Mining ist nur ein Schritt im Prozess des „Knowlegde Discovery in Database"(KDD), bei dem zunächst die Daten aufbereitet werden für das eigentliche Data-Mining und danach weiter ausgewertet werden. Beim Data-Mining ist es das Ziel aus den Daten noch unbekannte Informationen und Zusammenhänge zu extrahieren[5].

1.2 Was ist Clustering?

Clustering ist ein Verfahren um Datensätze mit Ähnlichkeiten zu Gruppen (bzw. Clustern) zusammenzufassen. Diese Cluster werden anschließend für eine weitere Verarbeitung in Verfahren zur automatischen Klassifizierung oder zur Mustererkennung genutzt. Die Ergebnisse hängen stark von der Vorverarbeitung (z. B. der Normierung der Daten bzw. dem Entfernen von Löchern) ab, sodass es oft nötig ist, die Daten mit unterschiedlichen Parametern zu testen. Bei der Vorverarbeitung wird versucht, dass man keine Löcher oder unbekannte Werte in den Daten hat, da die Algorithmen damit nicht arbeiten können. Zudem werden die Daten oftmals normiert, sodass sich die Attribute im selben Intervall befinden. Die ist nötig, damit Distanzfunktionen beim Clustering funktionieren. Die meist verbreitetsten Verfahren basieren auf Dichteverteilungen bzw. Abständen der Datenpunkte zueinander. Eins dieser Verfahren ist DBScan[4], das Verfahren benötigt zwei Parameter, welche das Verhalten des Algorithmus beeinflussen. Zum einen wird die Mindestanzahl an Punkten für einen Cluster benötigt und der maximale Abstand zwischen zwei benachbarten Punkten. Anderen Verfahren wird eine Menge an Clustern vorgegeben und die Daten werden dem bestmöglichen Cluster zu geordnet. Zu diesen Verfahren gehört k-Means[6], welches 1973 von J. A. Hartigan entwickelt wurde. Dabei werden die Daten den k Clustern zugeordnet unter der Bedingung, dass die Abstandsquadrate aller Punkte zu ihrem jeweiligen Clusterzentrum minimal werden.

1.3 Fluch der Dimensionalität

Der Fluch der Dimensionalität wurde von Richard Bellmann[2] erstmals 1961 erwähnt und beschreibt die rapide Volumenzunahme durch das Ansteigen der Dimensionen. Dies lässt sich mit dem folgenden Beispiel beschreiben:
Wenn man 100 zufällige Werte aus dem eindimensionalen Raum zwischen 0 und 1 nimmt, lässt sich dieser Bereich sehr gut abdecken. Wenn man jedoch einen 10-dimensionalen Raum mit Werten zwischen 0 und 1 nimmt und wieder 100 zufällige Werte zieht, dann ist die Abdeckung viel geringer. Will man eine ähnliche Abdeckung wie im eindimensionalen Raum erreichen benötigt man $100^{10} = 10^{20}$ Zufallswerte. Dies hat zur Folge, dass der Abstand zwischen der größten und der kleinsten Distanz innerhalb eines hochdimensonalen Raums gegen 0 geht:

$$\lim_{n\to\infty} \frac{dist_{max} - dist_{min}}{dist_{min}} \to 0 \,.$$

Durch diese Eigenheit kommt es beim Clustering mit Distanzfunktionen zu Problemen.

Um diesem Effekt entgegenzuwirken versucht man, irrelevante Attribute aus den Daten vor dem Clustering zu entfernen und damit die Anzahl der Dimensionen zu reduzieren.

1.4 Dimensionsreduktion

Bei der Dimensionsreduktion versucht man einen hochdimensionalen Datensatz in einen niedrig dimensionaleren Raum zu überführen ohne dabei relevante oder wertvolle Informationen für das weitere Data-Mining zu verlieren. Dies geschieht auf verschiedene Arten. Zum Beispiel wird versucht die Gesamtvarianz möglichst zu erhalten mit einer möglichst geringen Anzahl von Attributen oder man entfernt beispielsweise schwach korrelierte Attribute. Zu den bekanntesten Vertretern gehört die Hauptkomponentenanalyse, kurz PCA. Doch all diese Verfahren haben ihre Stärken und Schwächen, welche in dieser Arbeit behandelt werden.

1.5 ELKI

Für den gesamten Prozess des Data-Mining wird an der Ludwig-Maximilian-Universität München eine Open-Source Software Names „ELKI: Environment for Developing KDD-Applications Supported by Index-Structures“[1] in Java entwickelt. Die Software beinhaltet einen Großteil der in dieser Arbeit verwendeten Algorithmen und ist darauf ausgelegt schnell um neue Verfahren erweitert zu werden.
ELKI ist sehr modular aufgebaut, sodass die Schritte über das Einlesen der Daten, dass Verwalten im Speicher und das Anwenden von verschiedenen Algorithmen unabhängig voneinander ist. Dadurch ist es möglich die Daten aus verschiedensten Dateiformaten auszulesen und dennoch dieselben Algorithmen zu verwenden. Im Rahmen der Bachelorarbeit werden für alle Datensätze einfache CSV-Dateien genutzt.

2 Verfahren

Die Verfahren zur Dimensionsreduktion lassen sich in die Kategorien Feature Selection und Feature Extraction aufteilen. Bei der Feature Selection wird eine Teilmenge der Attribute bestimmt, welche keine redundanten und irrelevanten Attribute mehr enthält.

2.1 Feature Extraction

Die Feature Extraction ist ein Ansatz um hochdimensionale Daten in einen niedriger dimensionalen Raum zu transferieren. Oftmals werden dabei die Dimensionen durch Linearkombinationen zusammen gefasst um die Daten weniger Dimensionen zu verarbeiten. Es gibt auch nicht-lineare Verfahren, jedoch werden diese in dieser Arbeit nicht behandelt.

2.1.1 PCA - Hauptkomponentenanalyse

Die Hauptkomponentenanalyse (englisch Principal Component Analysis, kurz PCA) ist ein Verfahren der multivariaten Statistik. Man versucht die Attribute zu ersetzen durch eine Menge von Linearkombinationen dieser, wodurch die Anzahl reduziert wird. Die Daten werden dabei durch eine Hauptachsentransformation in einen Vektorraum mit einer neuen Basis überführt und dadurch wird die Korrelation der Attribute minimiert.
Die Hauptachsentransformation besteht aus einer orthogonalen Matrix welche die Eigenvektoren der Kovarianzmatrix als Spalten besitzt. Die Kovarianzmatrix beinhaltet die Kovarianzen zwischen allen Attributen.
Sei $A = (X_1, ..., X_n)^T$ der Vektor mit allen Attributen $X_1, ..., X_n$, so ist die Kovarianzmatrix $Cov(A)$ definiert als

$$Cov(A) = \begin{pmatrix} Cov(X_1, X_1) & \cdots & Cov(X_1, X_n) \\ \vdots & \ddots & \vdots \\ Cov(X_n, X_1) & \cdots & Cov(X_n, X_n) \end{pmatrix}$$

wobei $Cov(X, Y)$ definiert ist als

$$Cov(X, Y) = E\left((X - E(X)) \cdot (Y - E(Y))\right)$$

mit dem Erwartungswert $E(X)$ des Attributs X.

Das ganze lässt sich gut anschaulich erklären, wenn man die Daten als Wolke oder Ellipse in einem n-dimensionalem Raum auffasst. Dabei wird der Ursprung des Koordinatensystems in den Schwerpunk der Wolke gelegt. Nun werden nach und nach die Achsen so ausgerichtet, dass den Richtungen mit der größten Varianz bzw. der größten Ausdehnung folgen. Dabei entspricht die erste Achse der größten Ausdehnung und die zweite Achse der zweitgrößten Ausdehnung. So bekommt man p unabhängige Vektoren, welche der neuen Basis entsprechen. Die Basis-Vektoren entsprechen dabei den Eigenvektoren von $Cov(A)$. Nun muss man sich entscheiden, ob man eine feste Anzahl an Dimensionen p bekommen will, oder so lange weitere Vektoren bestimmt, bis man einen bestimmten Prozentsatz der ursprünglichen Varianz abbilden kann.

2.1.2 MDS - Multidimensionale Skalierung

Das Ziel der Multidimensionalen Skalierung ist es, die Objekte in einem Raum möglichst exakt zu positionieren ohne dabei die Distanzen zwischen den Objekten zu beeinflussen. Meistens wird MDS genutzt um Daten in einen zwei- oder dreidimensionalen Raum abzubilden und somit besser zu veranschaulichen. Jedoch funktioniert der Algorithmus mit einem beliebig dimensionalen Raum. Dazu wird eine Matrix A mit den paarweisen euklidischen Distanzen zwischen allen Datensätzen gefüllt. Danach wird der Durchschnittswert der Spalte sowie der Zeile in der ein Element steht von diesem abgezogen und der Durchschnittswert der Matrix A hinzu addiert:

$$a_{ij} = a_{ij} - \bar{a_i} - \bar{a_j} + \bar{A} \,.$$

Zum Schluss werden die Eigenvektoren sowie die dazu gehörigen Eigenwerte der Matrix A berechnet. Um die Daten in einen n-dimensionalen Raum zu projizieren werden die n größten Eigenwerte mit den dazu gehörigen Eigenvektoren multipliziert und bilden so die neuen Spalten bzw. Attribute.

2.2 Feature Selection

Die Feature-Selection kann man als Spezialfall der Feature-Extraction bezeichnen, da man keine Kombinationen aus den Attributen bildet, sondern nur gewisse Attribute übernimmt in eine Untermenge. Bei der Feature-Selection geht man von der Annahme aus, dass die vorhandenen Daten redundant sind und somit gewisse Attribute keinen Mehrwert an Information bieten.

2.2.1 Greedy-Forward-Selection

Bei der Greedy-Forward-Selection geht man von einen leeren Untermenge von Attributen aus und fügt nach und nach Attribute hinzu, wenn sich dadurch die Ergebnisse des Lernalgorithmus verbessern. Der Lernalgorithmus überprüft dabei ob sich das Ergebnis eines Clusterings durch Hinzunahme eines neuen Attributes verbessert hat. Diese Algorithmen werden oft verwendet beim maschinellen Lernen um aus Testdaten ein Verhalten für neue unbekannte Daten anzulernen. Da es sich um einen Greedy-Algorithmus handelt, kann es passieren, dass es sich nur um ein lokales Maximum handelt und somit

nicht das beste Untermenge gefunden wird.

Eingabe : Menge D der Daten, Menge F mit den Attributen
Rückgabewert : Untermenge von F
Subset = [];
while *alte Bewertung <= neue Bewertung und F ist nicht leer* **do**
  for *Attribut a in F* **do**
    berechne neue Bewertung für die neue Attribut-Menge mit dem Attribut a
    if *neue Bewertung ist besser als das Alte* **then**
      übernehme das Attribut in das Subset
      übernehme die Bewertung für das Subset
      entferne das Attribut aus der Menge F
      break
    end
  end
end
return *Untermenge*

Algorithmus 1 : Pseudocode zur Greedy-Forward-Selection

Gleichzeitig gibt es auch noch die Backward-Elimination, bei der zu Beginn von allen Attributen ausgegangen wird und nach und nach unnötige Attribute mit wenig Varianz entfernt werden.

2.2.2 Korrelationsbasierte Feature-Selection

Bei der korrelationsbasierten Feature-Selection wird die Korrelationsmatrix zwischen allen Attributen bestimmt. Die Korrelationsmatrix ist eine spezielle Art der Kovarianzmatrix bei der alle Attribute standardisiert werden, sodass ihr Erwartungswert gleich Null ist und die Varianz Eins ist. Dies wird für ein Attribut X erreicht mit

$$Z = \frac{X-\mu}{\sigma}$$

wobei $E(X) = \mu$ und $Var(X) = \sigma^2$ ist. Dabei werden dann die Attribute ausgewählt, die hoch korreliert mit der Klasse sind und nur eine niedrige Korrelation mit anderen Attributen haben. Die Klasse entspricht dabei einem Label bzw. einem Attribut dessen Verteilung ermittelt werden soll.

Die Korrelation von zwei Attributen (X,Y) mit i Datensätzen lässt sich bestimmen durch:

$$r_{XY} = \frac{\sum_i (x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\sum_i (x_i-\bar{x})^2}\sqrt{\sum_i (y_i-\bar{y})^2}},$$

wobei $\bar{x}$ und $\bar{y}$ jeweils der Mittelwert von X und Y sind. Der Wert von r_{XY} geht gegen 1 oder -1 wenn die Attribute sehr stark korreliert sind und gegen 0 wenn sie unabhängig voneinander sind.

2.3 Clustering-Verfahren

2.3.1 k-Means

Der Algorithmus k-Means wurde in den 1960er Jahren von Lloyd entwickelt und geht davon aus, dass man das Clustering gefunden hat, wenn man jeden Punkt einem Cluster-Zentrum zugeordnet hat und die Summe über die Abstandsquadrate aller Punkte zu ihren Zentren minimal ist. Der Algorithmus benötigt die gewünschte Anzahl an Clustern k und arbeitet dabei in drei Schritten.

Eingabe : Menge D der Daten, Anzahl der Cluster k
Rückgabewert : Daten mit Zuordnung zu Clustern
1. Initialisierung: Wähle k Werte als zufällige Cluster-Zentren $m_1, ...m_k$ aus den Daten
while *alte Cluster <> neue Cluster* **do**
  2. Zuordnung: Jeder Datenpunkt wird dem Cluster-Zentrum zugeordnet dem es am nächsten liegt
  3. Neu Berechnung: Berechne alle Cluster-Zentren neu als Mittelwert aller zugeordneten Punkt
end

Algorithmus 2 : Pseudocode zur k-Means

Die Schritte 2 und 3 werden so lange wiederholt, bis sich die Zentren nicht mehr verschieben. Dies entspricht der Optimierung der Funktion

$$J = \sum_{i}^{k} \sum_{x_j \in S_i} |x_j - \mu_j|^2 ,$$

wobei S_i die Menge aller Punkte im Cluster i ist und μ_j das Zentrum des Clusters i ist.

Da die Anzahl der Cluster k meistens im Vorhinein nicht bekannt ist, muss dieser Wert durch mehrere Durchläufe experimentell bestimmt werden. Der Algorithmus beinhaltet jedoch keine Ausreißer-Erkennung und es können nur konvexe Cluster erkannt.

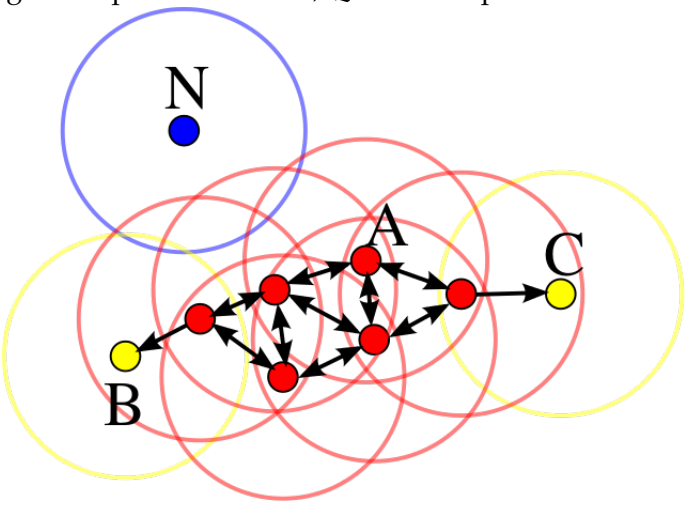
Dabei sind Ausreißer Punkte, die offensichtlich nicht zu den anderen Daten passen, möglicherweise durch fehlerhafte Datenerfassung.

2.3.2 DBScan

Der Algorithmus DBScan wurde 1996 von Martin Ester und Hans-Peter Kriegel vorgestellt[4]. Der Algorithmus ist im Gegensatz zu k-Means in der Lage selbst alle Cluster zu erkennen und Ausreißer auszusortieren. Dabei werden beliebige Formen von Clustern erkannt, solange diese eine gewünschte Dichte und Anzahl an Punkten besitzen. Dazu sind die Parameter ε und $minPts$ nötig.

Im Beispiel in Abbildung 1 ist A ein Kernpunkt vom Cluster, gleiches gilt für die anderen roten Punkte. All dies sind Punkte, bei denen in der ε-Umgebung mindestens $minPts$ weitere Punkte sind. B und C sind im gleichen Cluster, da diese dichte-erreichbar von A aus sind, sie liegen also in der ε-Umgebung eines Kernpunkts. N ist in keinem Cluster,

Abbildung 1: Beispiel für DBScan, Quelle Wikipedia:Chire CC BY-SA 3.0



da keine Verbindung zu einem Cluster besteht und auch nicht die Mindestanzahl an Punkten für ein Cluster gegeben ist, somit ist N Rauschen.

2.4 Clustering-Auswertung

Um die Ergebnisse der unterschiedlichen Dimensionsreduktionsverfahren zu vergleichen, gibt es viele Bewertungsalgorithmen. Eine Grundidee ist, dass man die Cluster, die nach dem Clustering der reduzierten Daten mit k-Means oder DBScan bestimmt hat, mit den vorhanden Labeln vergleicht. Wenn jedoch keine Label vorab bekannt waren, werden Algorithmen verwendet, welche bestimmen wie gut die Cluster voneinander getrennt sind.

Der grundsätzliche Ablauf für die Bewertung der Dimensionsreduktionsverfahren sieht so aus, dass zunächst die ursprünglichen Daten geclustert werden um einen Referenzwert zu berechnen. Danach werden die Algorithmen zur Dimensionsreduktion mit unterschiedlichen Parametern auf die Daten angewendet und die Daten danach geclustert. Dabei entstehen oftmals unterschiedliche Ergebnisse, sodass diese bewertet werden müssen um Vor- und Nachteile zu erkennen.

2.4.1 Silhouettenkoeffizient

Der Silhouettenkoeffizient ist ein Maß für die Qualität eines Clusterings in Abhängigkeit der Anzahl der Clusterzentren. Die Silhouette eines Punkts o wird mit

$$S(o) = \begin{cases} 0 & \text{wenn dist(A,o) = 0} \\ \frac{dist(B,o)-dist(A,o)}{max(dist(A,o),dist(B,o))} & \text{sonst} \end{cases}$$

berechnet, wobei A der Cluster ist, dem der Punkt o zugeordnet ist und B der nächst gelegene Cluster zu dem o nicht gehört ist. Der Silhouettenkoeffizient für einen Cluster C mit n_C Punkten ist:

$$S_C = \frac{1}{n_C} \sum_{o \in C} S(o) \, .$$

Für eine Gesamtbewertung eines Clusterings C mit k Clustern wird der Silhouettenkoeffizient berechnet mit

$$S = \frac{1}{k} \sum_{c \in C} S_c \, .$$

Wenn der Wert von S zwischen 0.75 und 1 liegt, spricht man von einer starken Strukturierung und man kann davon ausgehen, dass das Clustering gut ist. Unterhalb eines Werts von 0.5 lässt sich kaum eine Aussage treffen, da entweder die falsche Anzahl an Clustern vorhanden ist oder es keine Struktur in den Daten gibt. [7]
Keine Struktur bedeutet in diesem Fall, dass die Cluster nicht klar voneinander getrennt sind und es viele Punkte gibt, die nicht eindeutig zu einem Zentrum zugeordnet werden können. Ein Problem des Silhouettenkoeffizienten ist es, dass prinzipiell Clusterings bevorzugt werden bei dem jeder Punkt in einem eigenen Cluster ist. Da dabei die Abstände minimal werden.

2.4.2 F-Maß

Das F-Maß (engl. F-Measure) zählt zu den externen Auswertungsverfahren, da es die von außen bekannten Daten nutzt um das Ergebnis zu bewerten. Dabei wird auf die bekannten Label des Datensatzes zurück gegriffen und das harmonische Mittel über die Genauigkeit ($precision$) und die Trefferquote ($recall$) des Clusterings gebildet.

$$F_\alpha = (1 + \alpha) \cdot \frac{precision \cdot recall}{\alpha \cdot precision + recall}$$

Genauigkeit beschreibt dabei die Wahrscheinlichkeit, dass ein gefundenes Ergebnis relevant ist.

$$precision = \frac{|\text{relevantes Ergebnis} \bigcap \text{gefundenens Ergebnis}|}{|\text{gefundenens Ergebnis}|}$$

Die Trefferquote entspricht der Wahrscheinlichkeit, das ein relevantes Ergebnis gefunden wurde.

$$recall = \frac{|\text{relevantes Ergebnis} \bigcap \text{gefundenens Ergebnis}|}{|\text{relevantes Ergebnis}|}$$

Der Parameter α ist ein Korrekturfaktor um die Trefferquote stärker als die Genauigkeit zu bewerten, damit eher Ergebnisse nutzt werden, welche möglichst viele relevante Daten beinhalten.

2.4.3 Jaccard-Koeffizient

Der Jaccard-Koeffizient beschreibt die Ähnlichkeit zweier Mengen A und B durch das Verhältnis zwischen der Anzahl der Gemeinsamen und der Anzahl vereinigten Daten.

$$J(A, B) = \frac{|A \bigcap B|}{|A \bigcup B|} .$$

Der Koeffizient nimmt einen Wert zwischen 0 und 1 an, wobei 1 einer maximalen Übereinstimmung entspricht. Dieses Verfahren funktioniert jedoch nur mit Daten, die gelabelt sind, wo also die Zuordnung der Daten zu Clustern bekannt ist, da man sonst keine Übereinstimmung messen kann. Bei einem sehr genauen Clustering müsste der Jaccard-Koeffizient also gegen 1 gehen, da dann die Label der geclusterten Daten und die bekannten Label übereinstimmen.

3 Vergleich an Datensätzen

Um die Algorithmen aus Kapitel 2 zu vergleichen, werden diese auf verschiedene Datensätze angewendet. Zum einen auf generierte Datensätze mit geringen Dimensionen und auf empirische Datensätze mit hohen Dimensionen.

Dabei treten in den Grafiken folgende Abkürzungen auf:

plain Dieser Datensatz wurde vor dem Clustering nicht bearbeitet

euklid Die Daten dieses Datensatzes wurden euklidisch normiert

pca Die Daten wurden mit der Hauptkomponentenanalyse transformiert

pca_n Die Daten wurde mit der Hauptkomponentenanalyse auf n Dimensionen reduziert

mds_n Die Daten wurden mit MDS auf n Dimensionen reduziert

fs_greedy_n Die Daten wurden mit einer Greedy-Methode der Feature-Selection auf n Dimensionen reduziert

fs_correlation_n Die Daten wurden durch korrelationsbasierten Feature-Selection auf n Dimensionen reduziert

Alle Datensätze wurden mit k-Means geclustert, ausgewählte Datensätze wurden auf Grund ihrer Eigenschaften zusätzlich noch mit DBScan analysiert.

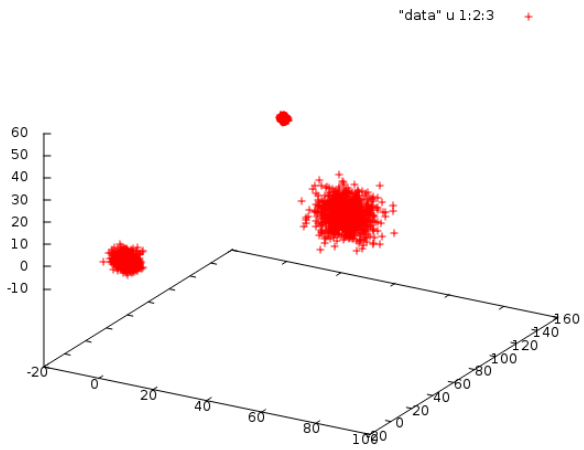
3.1 Generierte Datensätze

Zunächst wurden die Algorithmen an generierten Datensätzen mit eindeutig getrennten Clustern getestet um die Korrektheit zu überprüfen und einen ersten Vergleich ziehen zu können.

Kleine Dimension

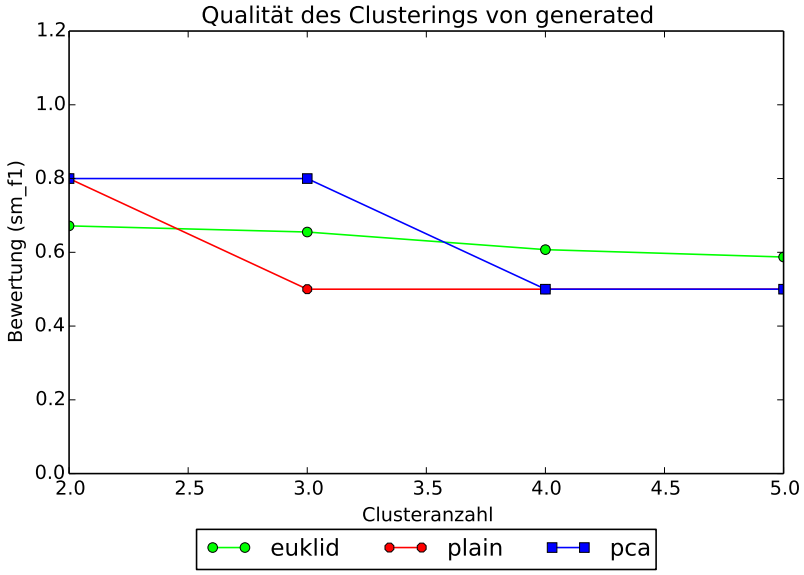
In der Abbildung 2 sieht man einen Datensatz mit drei Dimensionen in denen sich drei gauß-verteilte Cluster klar voneinander getrennt befinden. Dadurch sollten sich die Cluster einfach finden lassen durch ein Clustering.

Abbildung 2: Position der Cluster



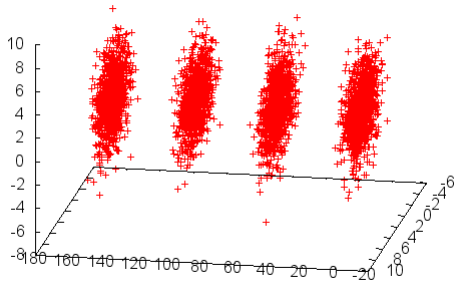
Die Ergebnisse des Clusterings in Abbildung 3 zeigen die Cluster-Bewertungen nachdem nur eine Skalierung (Euklid) bzw. eine Rotation (PCA) der Daten stattgefunden hat. Es ist zu erkennen, dass bereits eine Skalierung der Daten großen Einfluss haben kann, da nun kaum ein Unterschied in den Ergebnissen zu erkennen ist. Bei PCA gibt es zwar ein Maximum für drei Cluster, jedoch gibt es für zwei Cluster dasselbe Ergebnis. Dies lässt darauf schließen, dass es möglich ist das kleine Cluster zu einem anderen zugeordnet werden oder jedoch als eigenständiger Cluster mit gleicher Bewertung erkennt werden.

Abbildung 3: Ergebnis des Gauß-verteilten Datensatzes mit drei Dimensionen



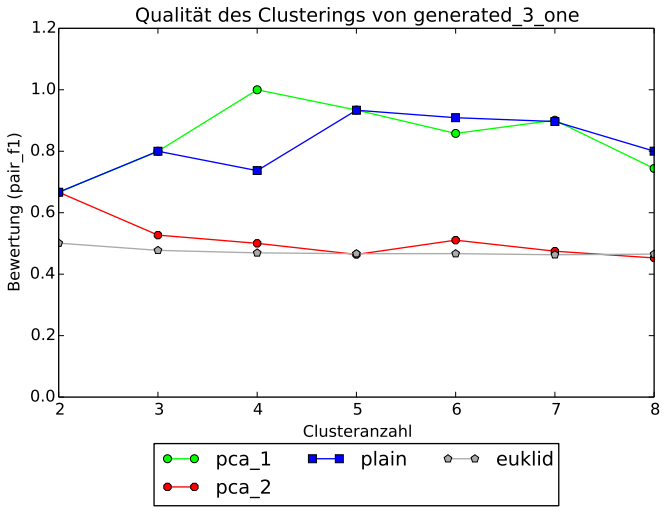
Beim zweiten Datensatz handelt es sich um vier Clustern die nebeneinander in einem drei dimensionalen Raum angeordnet sind, wie in Abbildung 4 zu sehen ist. Da die Abstände der Scheiben zueinander geringer ist als der Radius der Scheiben ist dies besonders ein Problem für Clustering-Algorithmen wie k-Means, die die Abstandsquadrate optimieren.

Abbildung 4: Position der Daten im Datensatz mit vier „Scheiben“.



In Abbildung 5 sind die Ergebnisse des Clusterings zu sehen. Die Ergebnisse der einzelnen Algorithmen unterscheiden sich sehr stark, jedoch bringt eine Dimensionsreduktion mit PCA auf eine Dimension ein sehr gutes Ergebnis. Dies hängt damit zusammen, dass sich der größte Teil der Varianz in einer Dimension befindet, sodass es einfach ist, wenn man nur in dieser Dimension die Cluster berechnet. Sobald mehr Dimensionen hinzu kommen werden die Ergebnisse mit k-Means schlechter, wie man bei der Datenreihe pca_2 sieht. Wenn man das Clustering mit DBScan macht, wobei die Cluster anhand iher Dichte erkannt werden, lassen sich ebenfalls die vier Scheiben erkennen und man erhält ein sehr gutes Clustering mit einem F_1-Wert von über 0.9. Bei DBScan ist es jedoch nicht nötig die Daten zuvor zu reduzieren.

Abbildung 5: Ergebnis des Clusterings

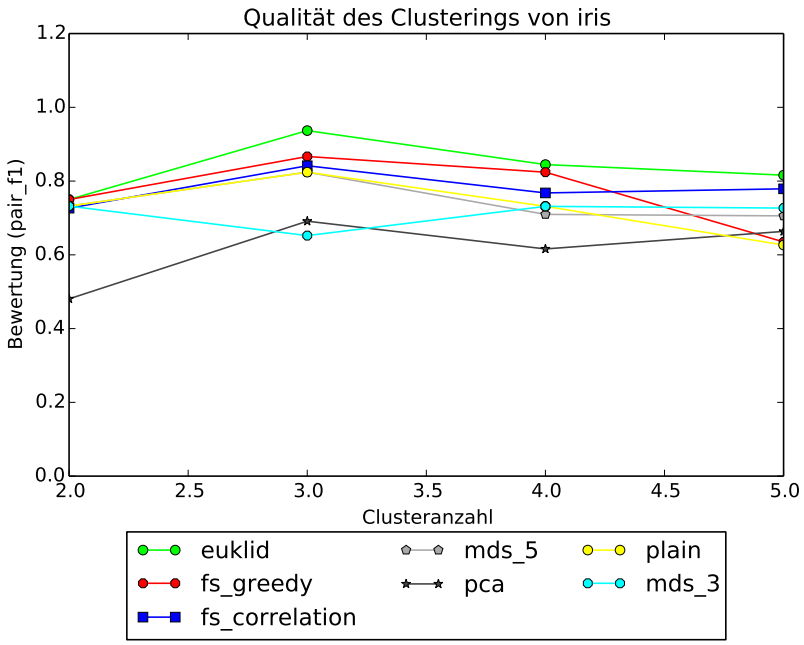


3.2 Empirische Datensätze

3.2.1 Iris Plants Database

Der Iris Datensatz von 1988 beinhaltet 150 Datensätze mit vier Attributen und einem Label. Es gab drei Werte für das Label, sodass drei Cluster bekannt waren.

Abbildung 6: Ergebnis des Iris Datensatzes



Bei diesem Datensatz führte ein Clustering auf den Originaldaten (Datensatz euklid) zum besten Ergebnis, da auf Grund der geringen Dimension schnell Daten verloren gehn. Die hohen Bewertungen der Clusterings im Bereich von vier und fünf Clustern ist zurück zu führen auf das Bilden von Unterclustern in den großen Clustern.

3.2.2 Mouse Datensatz

Der Mouse Datensatz stammt von den ELKI Entwicklern und enthält drei Cluster mit 500 Datensätzen sowie einiges an Rauschen in zwei Dimensionen in der Form eines Mäusekopfes.

Abbildung 7: Position der Datenpunkte im Mouse-Datensatz

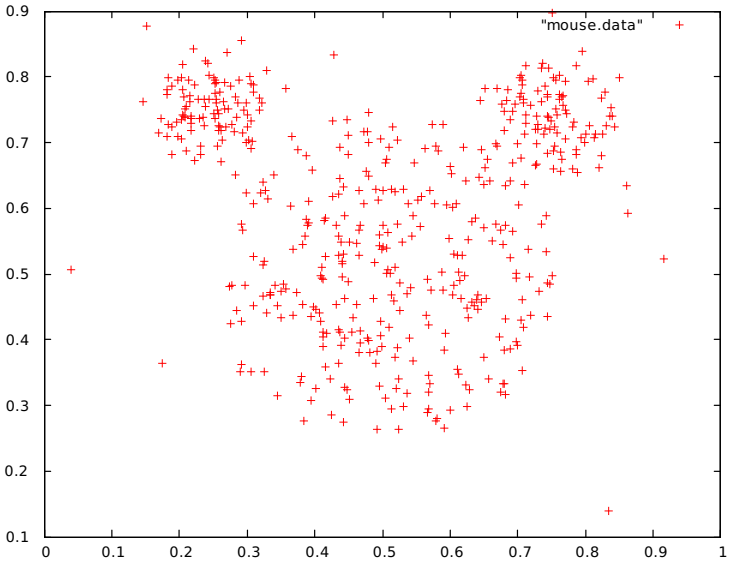
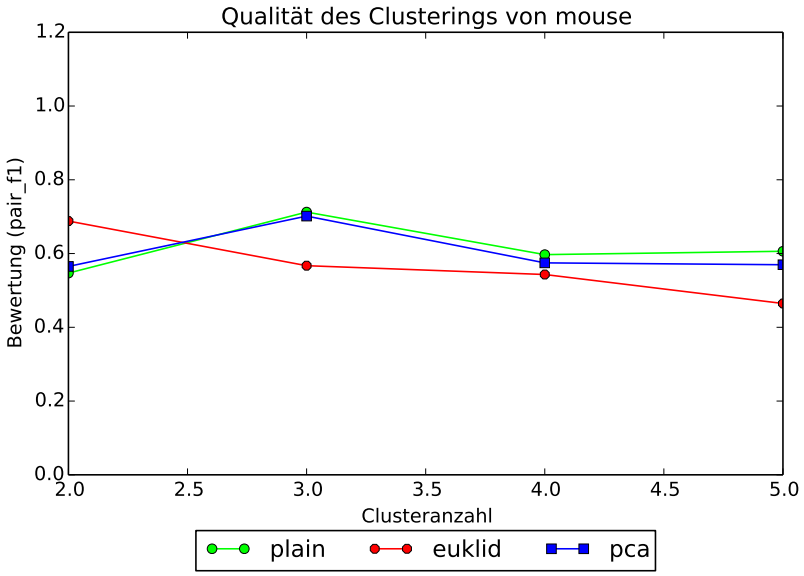


Abbildung 8: Ergebnis des Mouse-Datensatzes

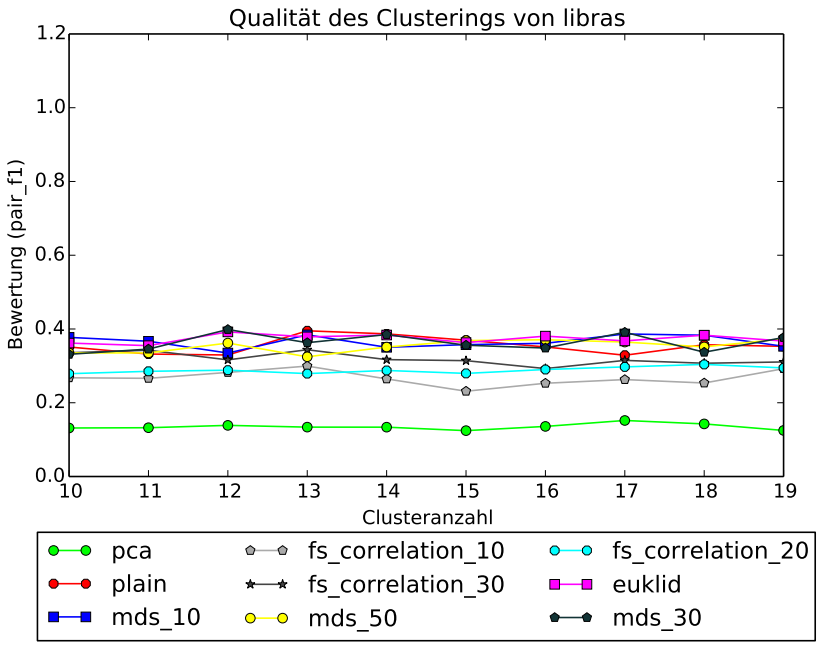


Bei diesem Datensatz wurde auf Grund der geringen Dimension mit PCA ebenfalls keine Reduktion durchgeführt, sondern nur eine Skalierung bzw. Rotation. Das in Abbildung 8 erkannbare Plateau bei 4 und 5 Clustern lässt sich dadurch erklären, dass der große Cluster sich auch in mehrere gleichgroße Cluster unterteilen lässt, welche dann eine ähnliche Größe wie die beiden oben Rechts und Links.

3.2.3 LIBRAS

Der LIBRAS (LÍngua BRAsileira de Sinais) Datensatz besteht aus 90 Features welche die Handbewegung der Zeichensprache erfassen. Es handelt sich um 15 Handbewegungen, sodass 15 Cluster-Zentren entstehen sollten.

Abbildung 9: Ergebnis des LIBRAS Datensatzes

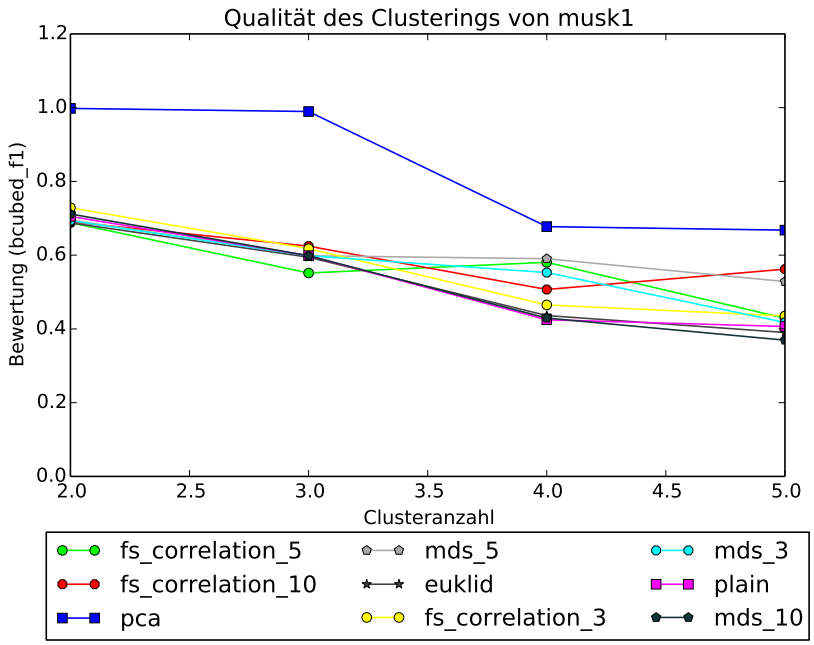


Die Ergebnisse dieses Clusterings sind bei allen getesteten Verfahren sehr ähnlich und lassen darauf schließen, dass die Daten keine erkennbare Struktur beinhalten. Dies deckt sich jedoch mit den Erkenntnissen der Ersteller des Datensatzes, da diese auch nur auf Ergebnisse von 0.35 bis 0.4 gekommen sind, wie es den Informationen zum Datensatz zu entnehmen ist. Das spricht dafür, dass die erhobenen Daten nicht reichen um die Daten eindeutig zu trennen.
Jedoch ist es auffällig, dass eine Transformation der Daten mit PCA ein sehr viel schlechteres Ergebnis liefert. Wenn die Daten mit PCA auf 10 oder 20 Dimensionen reduziert werden, kommt es zu einem Clustering, dass mit den anderen Verfahren vergleichbar ist.

3.2.4 MUSK Datensatz

Der Musk Datensatz besteht aus 166 Attributen zu 92 Molekülen welche in zwei Kategorien (Moschus und nicht-Moschus) eingeordnet worden sind. Sie sind im Rahmen eines Papers[3] von Thomas G. Dietterich entstanden.

Abbildung 10: Ergebnis des Musk1 Datensatzes



In Abbildung 10 ist klar zu erkennen, dass PCA das beste Ergebnis liefert. Da das Clustering für 2 und 3 Zentren bei fast allen Algorithmen ein ähnliches Ergebnis liefert, ist davon auszugehen, dass sich in den Daten 3 Cluster statt nur 2 wie erwartet befinden, dies ist jedoch nicht in dem Paper behandelt worden.

4 Auswertung

Um ein optimales Clustering zu erreichen, muss man sich zunächst über die Daten Gedanken machen, da besonders die Form der Daten einen großen Einfluss auf das Clustering haben kann. Bei niedrig-dimensionalen Daten ist es dabei hilfreich die Daten zu plotten um sich über die Form klar zu werden. Denn Clustering-Verfahren wie k-Means welche die Distanzquadrate optimieren haben ihre Probleme bei nicht sphärischen Daten. Dort ist ein dichte-basiertes Verfahren wie DBScan[4] besser geeignet, da es auf die Form der Daten eingeht und somit auch andere Verteilungen erkennt. Dies lies sich sehr gut an dem generierten Datensatz 4 mit vier „Scheiben"in drei Dimensionen erkennt. Dort lieferte DBScan ein nahezu perfektes Clustering wobei hingegen k-Means kein brauchbares Ergebnis liefert.

Algorithmus	F_1**-Score**	**Jaccard**
k-Means	0.65	0.50
DBScan	0.98	0.96

Tabelle 1: Ergebnisse von k-Means und DBScan am Datensatz 4.

Besonders bei Datensätzen wo die Daten sehr nah beieinander liegen und die Varianz niedrig ist, sollte man auf den Schritt der Dimensionsreduktion verzichten, da diese immer mit einem Verlust an Varianz verbunden ist. Dies wirkt sich dementsprechend stark auf solche Datensätze aus.

Algorithmus	F_1**-Score**
Plain	0,61
PCA	0,59

Tabelle 2: Auswirkung von PCA an einem Datensatz(Mouse) mit sehr nah beieinander liegenden Punkten und nur zwei Dimensionen

Ein anderes Problem das sich zeigt, ist wenn die Varianz auf alle Dimensionen (n) gleich verteilt ist, da es dann sehr schnell passiert, dass man viel Varianz verliert, wenn man die Daten in ihrer ursprünglichen Basis lässt. Wenn man dort die Dimensionsreduktion auf eine bestimmte Anzahl (m) Dimensionen beschränkt bekommt man maximal $\frac{m}{n}$ der ursprünglichen Varianz nach der Dimensionsreduktion. Dies ist besonders tragisch, wenn n viel größer ist als m. Die eigentliche Transformation der Daten in eine neue Basis wie es bei PCA geschieht ist von diesem Problem nicht betroffen, da dabei keine Varianz verloren geht. Da die neue Basis den Varianzen folgt, wird in manchen Dimensionen die Varianz größer und in anderen kleiner. Dies hat zu Folge, dass meist bei der anschließenden Reduktion weniger Varianz verloren geht, da die Dimensionen mit der geringsten Varianz entfernt wird.

Zudem kann es passieren, dass man sehr unterschiedliche Ergebnisse bekommen, wenn unterschiedliche Attribute auswählt werden. Dies kann beispielsweise passieren, wenn man die Daten mit einer Greedy-Forward-Selection auswertet und einmal mit einer Greedy-Backward-Selection. Denn dabei muss es nicht zu denselben Ergebnissen kommen, da man von unterschiedlichen Startpunkten ausgeht obwohl die Verfahren ver-

wandt sind. Deshalb empfiehlt es sich bei einer Dimensionsreduktion die Verfahren mehrfach mit unterschiedlichen Parametern durchzuführen und zu schauen, wo es zu den besten Ergebnissen kommt. Dies kostet jedoch natürlich Zeit und Rechenaufwand, beides Dinge, die man durch die Dimensionsreduktion teilweise auch reduzieren will.

Jedoch sind Dimensionsreduktions-Verfahren sehr hilfreich, wenn sich der Großteil der Varianz in einige wenige Dimensionen innerhalb von vielen Dimensionen befindet. Dabei zeigt sich besonders bei PCA, dass die Daten so transformiert werden, dass man nahezu alle Varianz wieder findet jedoch in erheblich weniger Dimensionen.

Da alle Verfahren versuchen die Varianz möglichst optimal zu halten, zeigte sich, dass wenn unterschiedliche Verfahren die Daten auf dieselbe Anzahl von Dimensionen reduzierten es zu sehr ähnlichen Ergebnissen kommt. Oftmals kommt es zu gleich guten Cluster-Bewertungen für eine bestimmte Anzahl an Clustern. Jedoch sieht auch der Verlauf in Abhängigkeit der Clusteranzahl ähnlich aus. Dies ist besonders gut am Datensatz Iris 7 zu erkennen.

Datensatz	**Bester Algorithmus**	**F1-Score**
Computerparts	PCA	0.51
„4 Scheiben“	PCA	0.99
Glass	PCA	0.96
Iris	Euklid	0.93
Libras	MDS mit 30 Dim	0.39
Mouse	Plain	0.71
Musk	PCA	0.99

Tabelle 3: Übersicht der besten Algorithmen nach Datensatz und die durchschnittliche Clustering-Bewertung

Wie man in Tabelle 3 sieht, ist nirgends eins der Feature-Selection-Verfahren am besten geeignet gewesen. Entweder war es am effektivsten die Daten direkt zu Clustern (Verfahren Plain) bzw. nur zu normieren (Verfahren Euklid) oder mit einem der Feature-Extraction-Verfahren (PCA und MDS) zu verarbeiten.

Leider zeigt sich, dass die Implementierung von MDS in ELKI sehr schlecht ist und Unmengen von Arbeitsspeicher und Rechenzeit benötigt. Ein Datensatz von 5 MB benötigte mehr als 8 GB RAM um mit MDS verarbeitet zu werden. Dies ist jedoch auch den ELKI-Entwicklern bekannt und durch folgenden Kommentar im Sourccode[1] angemerkt:

„Note: the current implementation is rather expensive, both memory- and runtime wise. Don’t use for large data sets! TODO: a contributed block Lanczos algorithm would be beneficial, to speed up MDS.“

Doch die Ergebnisse, die der Algorithmus liefert, sind korrekt und konnten weiter verarbeitet werden.

[1] http://elki.dbs.ifi.lmu.de/browser/elki/trunk/elki/src/main/java/de/lmu/ifi/dbs/elki/datasource/filter/transform/ClassicMultidimensionalScalingTransform.java

Insgesamt zeigt sich, dass bei PCA das beste Clustering sich besser erkennen lässt und meist stärker ausgeprägt ist als bei den anderen getestet Verfahren. Dies zeigt sich besonders offensichtlich beim Musk Datensatz, bei dem ein nahezu perfektes Clustering nach PCA möglich war, wohingegen bei allen anderen Verfahren sich kein klares Maximum zeigte.

5 Fazit

Ein Grundproblem bei der Datenverarbeitung ist die große Menge an Daten, da viele Algorithmen nicht für hoch-dimensionale Daten ausgelegt sind und man eine lange Rechenzeit in Kauf nehmen muss oder sogar gar keine Ergebnisse bekommt. Um dem entgegenzuwirken wurden Verfahren zur Dimensionsreduktion entwickelt.

Die in dieser Bachelorarbeit vorgestellten Verfahren zur Dimensionsreduktion finden jeden Tag ihre Anwendung und daher ist ein Vergleich der Ergebnisse dieser Verfahren interessant für die Arbeit im Bereich der Knowledge-Discovery.

Die getesteten Verfahren lassen sich in zwei Gruppen einteilen, die Feature-Extraction- und die Feature-Selection-Verfahren. Dabei zeigte sich, dass besonders die Feature-Extraction ein mächtiges Werkzeug für die Dimensionsreduktion ist, dass sie im Großteil der Fälle ein besseres Ergebnis als die Feature-Selection lieferte.

Besonders bei PCA ist es Auffälligkeit, dass eine reine Transformation der Daten in eine neue Basis ohne Dimensionsreduktion schon eine Verbesserung des Clusterings oftmals bringt. Zudem bringt PCA bei einer Reduktion der Daten meist auch eine Verbesserung des Ergebnisses gegenüber den anderen Verfahren.

5.1 Ausblick

Auch wenn die Ergebnisse bereits eine große Hilfe für den Prozess der Knowledge-Discovery ist, gibt es weitere Einflussfaktoren die ausgewertet werden sollten.

Im Rahmen einer weiteren Arbeit könnten verschiedene Clustering-Algorithmen verglichen werden, da diese ebenfalls einen sehr großen Einfluss auf das Ergebnis haben.

Besonders für das ELKI-Projekt wäre es wünschenswert, wenn ein besserer Algorithmus für MDS implementiert werden würde, welcher nicht Unmengen an Arbeitsspeicher und Rechenzeit benötigt.

Literatur

[1] Elke Achtert, Hans-Peter Kriegel, and Arthur Zimek. Elki: A software system for evaluation of subspace clustering algorithms. 2008.

[2] R. Bellman and R.E. Bellman. *Adaptive Control Processes: A Guided Tour*. 'Rand Corporation. Research studies. Princeton University Press, 1961.

[3] Thomas G. Dietterich, Richard H. Lathrop, and Tomás Lozano-Pérez. Solving the multiple instance problem with axis-parallel rectangles. *Artif. Intell.*, 89(1-2):31–71, January 1997.

[4] Martin Ester, Hans-Peter Kriegel, Jörg S, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. pages 226–231, 1996.

[5] Martin Ester and Jörg Sander. *Knowledge Discovery in Databases:*. Springer Berlin Heidelberg, 2000.

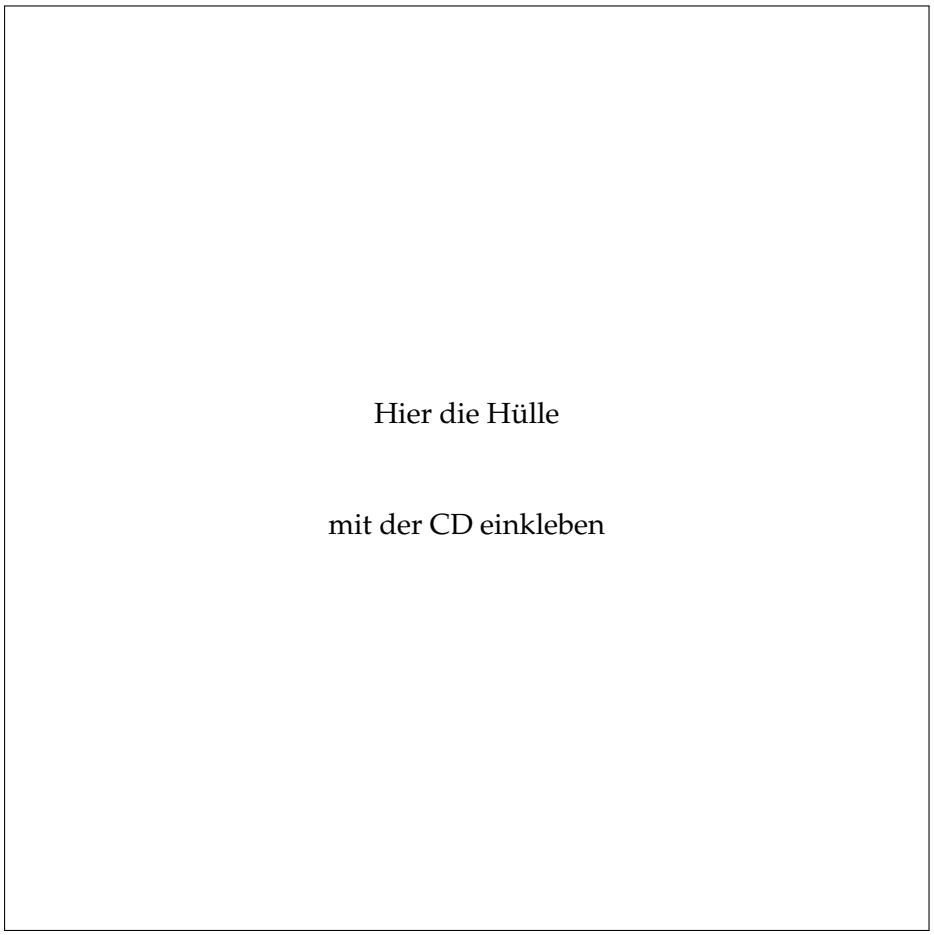
[6] John A Hartigan. Clustering. *Annual review of biophysics and bioengineering*, 2(1):81–102, 1973.

[7] Peter J. Rousseeuw. *Silhouettes - A Graphical Aid to the Interpretation and Validation of Cluster Analysis*. Department of Mathematics and Informatics, University of Technology, 1984.

Abbildungsverzeichnis

Tabellenverzeichnis

Liste der Algorithmen


Hier die Hülle
mit der CD einkleben