Institut für Parallele und Verteilte Systeme

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Bachelorarbeit

Metriken zur Evaluation von Teilschritten in Data Mining Analysen

Dennis Tschechlov

Studiengang:	Softwaretechnik
Prüfer/in:	PD Dr. rer. nat. habil. Holger Schwarz
Betreuer/in:	Manuel Fritz, M.Sc., Dipl.-Inf. Michael Behringer
Beginn am:	17. April 2017
Beendet am:	17. Oktober 2017
CR-Nummer:	H.3.3, I.5.3

Kurzfassung

In dieser Arbeit wurde ein Konzept entwickelt, bei der für den K-means und den K-means++ eine effiziente Berechnung der Metriken ermöglicht wurde. Mit Hilfe der Metriken wurde anschließend ein Konvergenzkriterium zur frühzeitigen Terminierung für den K-means und den K-means++ aufgestellt. In den Experimenten konnte gezeigt werden, dass sowohl für synthetische, als auch für reale Datensätze Einsparungen in den Iterationen und der Ausführungszeit von über 90% möglich waren. Zudem wurde verdeutlicht, dass eine höhere Einsparung der Ausführungszeit auch gleichzeitig mit einem höheren Qualitätsverlust verbunden ist. Des Weiteren wurden diese beiden Metriken genutzt, um geeignete Zeitpunkte für eine Visualisierung auszumachen. Dabei ergaben sich für beide Metriken, die für jeweils beide Algorithmen geprüft wurden, je unterschiedliche Werte für die Anzahl der Visualisierungen. Diese erstreckten sich von 0% bis 30% der Anzahl der Iterationen des jeweiligen Durchlaufs. Es wurde zudem aufgezeigt, dass für beide Metriken die meisten Durchläufe der Visualisierungen im Bereich von 5% bis 20% waren. Daraufhin wurden beide Ansätze kombiniert, das heißt, dass untersucht wurde wie viele Visualisierungen sich bis zum festgelegten Konvergenzkriterium ergeben. Dabei hat sich herausgestellt, dass sich dafür deutlich mehr Visualisierungen im Verhältnis von Visualisierungen und Iterationen bis zur Konvergenz ergab.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Verzeichnis der Algorithmen

1. Einleitung

Im heutigen Big Data-Zeitalter wächst die Anzahl der Daten exponentiell an. Mit jedem Jahr verdoppelt sich das jährlich weltweit generierte Datenvolumen. Laut einer Studie von Seagate wurden 2013 ungefähr 3,5 Zetabyte an Daten generiert. 2020 hingegen soll die Anzahl auf bis zu 40 Zetabyte ansteigen [RGR17]. Das entspricht einem jährlich generierten Datenvolumen von 5200 GB pro Person. Dazu zählen unter anderem alle möglichen Arten von Daten, die durch eine Interaktion mit Menschen im Internet geschehen [FAT+14]. Dies ist beispielsweise in sozialen Netzwerken wie Facebook oder Twitter, aber auch in einer *e-Commerce* Plattform wie Amazon der Fall. So fallen bei Amazon täglich mehr als 10 Millionen Transaktionen und 3 Millionen Einkäufe an [DHJ+07]. Durch eine geeignete Analyse dieser Daten ist es möglich das Kaufverhalten der Kunden zu analysieren und die Kunden zu gruppieren, wodurch den Kunden Vorschläge für einen weiteren Kauf gemacht werden können, der auf dem bisherigen Interessen basiert. Die Analyse, bei der aus einer Sammlung von Daten Informationen gewonnen werden, wird auch als *Data Mining* bezeichnet . Ein weiteres Anwendungsgebiet des Data Minings ergibt sich für Banken und Versicherungen in der Erstellung einer Risikoanalyse. Wenn zum Beispiel entschieden werden soll, ob einem Kunden ein Kredit bzw. eine Lebensversicherung angeboten werden soll[Fas]. Es kann allerdings auch in der Biologie eingesetzt werden, um verschiedene verschiedene Gene mit ähnlicher Funktion zusammen zu fassen oder um Pflanzen und Tiere zu klassifizieren [HPK11].

Durch die steigenden und komplexer werdenden Daten stoßen die Data Mining-Verfahren jedoch schnell an ihre Grenzen. Zum Einen müssen die Daten vor einer Analyse oft vorverarbeitet werden, da sie unvollständig oder fehlerbehaftet sein können. Dadurch ergibt sich ein längerer und meist auch iterativer Prozess. Des Weiteren ist das Ergebnis einer Analyse nur so gut wie es von einem menschlichen Experten interpretiert wird. Denn für die Analyse sind zahlreiche verschiedene und komplexe Beziehungen zu beachten, die eine automatisierte Auswertung der Analyse erschweren [WZWD14]. Die Data Mining-Verfahren sind zwar meist iterative Verfahren, jedoch ist es nicht möglich die Zwischenschritte dieser Verfahren einzusehen. Dadurch sind die Verfahren wie eine undurchsichtige Black-Box, da zwischen Eingabe der Parameter und der Ausgabe des Resultats keinerlei Steuerungsmöglichkeit besteht und somit oft unklar ist unter welchen Umständen Ergebnisse zustande kommen [JMF99]. Dies wird in Abbildung 1.1 dargestellt. Die transformierten Daten stellen dabei Daten dar, die vor verarbeitet wurden, um in einem für die Analyse geeigneten Format vorzuliegen. Dazu gehört zum Beispiel die Bereinigung der Daten von Ausreißern oder eine Voraggregation der Daten. Erst wenn die Analyse abgeschlossen ist, wird das Ergebnis zur Evaluation weitergeleitet. Diese wird dann von einem menschlichen Experten durchgeführt, der anhand von geeigneten

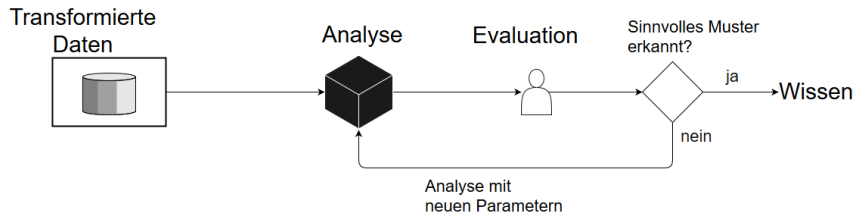


Abbildung 1.1.: Ist-Zustand des Analyseprozesses

Metriken bewertet, ob das Ergebnis sinnvoll ist oder nicht. Ist das Ergebnis sinnvoll, erhält man neues Wissen über die Daten, wenn nicht, dann muss der Analyseschritt mit neuen Parametern (und eventuell einem anderen Verfahren) erneut durchgeführt werden.

Zudem beinhalten die Analyseverfahren häufig Optimierungsfunktionen. Diese Funktionen streben einen Zielzustand an, der z. B. durch die Minimierung einer Fehlerfunktion erreicht werden soll. Dies resultiert jedoch oft in einer Langläufigkeit der Berechnungen, besonders wenn es sich um große Datenmengen handelt. Um dem entgegenzuwirken wird ein Konvergenzkriterium benötigt, welches ein approximiertes Ergebnis liefert. Für die Bewertung der Zielzustände der Verfahren gibt es zahlreiche Metriken, wodurch Rückschlüsse auf die Eignung der Parameter gezogen werden können. Dabei entstehen allerdings zwei Probleme:

i) Die Metriken sind sehr komplex und zahlreich und je nach Verfahren werden andere Metriken benötigt.

ii) Die Bewertung findet erst nach der Terminierung der Verfahren statt.

1.1. Ziele

Ein Ziel dieser Arbeit ist es, die Analyse und die Evaluation enger miteinander zu verknüpfen. Dies soll durch einen Zustand erreicht werden, der in Abbildung 1.2 dargestellt wird.

Dabei werden dem Nutzer Zwischenergebnisse der Verfahren gezeigt. Der Nutzer hat dann die

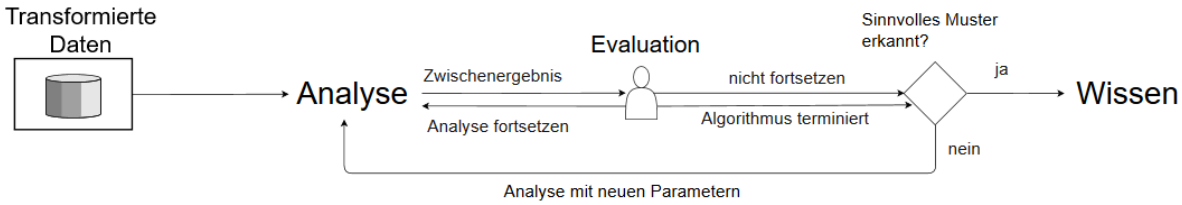


Abbildung 1.2.: Soll-Zustand des Analyseprozesses

Möglichkeit zu entscheiden, ob er die Analyse fortsetzen möchte oder nicht. Dadurch soll es für den Nutzer nachvollziehbarer werden, wie bestimmte Ergebnisse entstehen. Ein weiterer

Punkt ist, dass der Nutzer frühzeitig erkennen kann, falls eine Analyse eine unerwünschte Form annimmt und er kann das Verfahren früher abbrechen, um es mit neuen Parametern zu wiederholen. Somit wird auch Zeit eingespart, da der Nutzer nicht die komplette Analyse abwarten muss, um dann festzustellen, dass das Ergebnis nicht sinnvoll ist. Die Zeitpunkte der Visualisierung sollen dabei mit Hilfe von geeigneten Metriken bestimmt werden.

Ein weiteres Ziel dieser Arbeit zielt ebenfalls darauf ab, die Analyse-Verfahren zu verkürzen. Dabei sollen die Analyseverfahren frühzeitig terminieren, wenn bereits eine ausreichende Qualität erreicht ist und keine großen Änderungen zu erkennen sind. Dazu soll mit Hilfe von geeigneten Metriken ein Konvergenzkriterium aufgestellt werden, welches angibt, wann eine ausreichende Qualität erreicht ist. Die Ziele fokussieren sich in dieser Arbeit auf Clustering-Algorithmen, insbesondere den K-means und den DBSCAN.

1.2. Gliederung der Arbeit

Die Arbeit ist in folgender Weise gegliedert:

Kapitel 2 – Grundlagen und verwandte Arbeiten: In diesem Kapitel werden die Grundlagen für diese Arbeit geschaffen und es wird auf verwandte Arbeiten eingegangen.

Kapitel 3 – Konzept zur Berechnung der Metriken in Teilschritten: In diesem Kapitel werden geeignete Metriken ausgewählt und es wird ein Konzept für eine optimierte Berechnung dieser Metriken vorgestellt

Kapitel 4 – Experimentelle Umsetzung: In diesem Kapitel wird auf die prototypische Implementierung eingegangen und der Ablauf der Experimente wird beschrieben.

Kapitel 5 – Evaluation: In diesem Kapitel wird evaluiert welche Metriken sich für ein Konvergenzkriterium eignen und welche sich für die Visualisierung von Zwischenergebnissen eignen. Anschließend wird untersucht wie viel Zeit mit dem Konvergenzkriterium eingespart werden kann und mit welchem Qualitätsverlust dies verbunden ist. Zudem werden zwei Ansätze für die Visualisierung von Zwischenergebnissen vorgestellt und mit einander verglichen.

Kapitel 6 – Zusammenfassung und Ausblick: In diesem Kapitel wird eine Zusammenfassung der Arbeit gegeben und es wird darauf eingegangen, wo noch Potenzial für weitere Forschungen vorhanden ist.

2. Grundlagen und verwandte Arbeiten

In diesem Kapitel werden die nötigen Grundlagen für die nachfolgenden Kapitel dieser Arbeit geschaffen und es wird eine Übersicht über verwandte Arbeiten gegeben. Dazu wird ein spezifisches Verfahren der Analyse eingegangen, das Clustering, und es werden die verschiedenen Möglichkeiten der Evaluation des Clusterings aufgezeigt. Bevor es allerdings zu der Analyse kommt, müssen die Daten aus einer riesigen Datenmenge ausgewählt und meist auch noch vor-verarbeitet werden, damit überhaupt eine Analyse stattfinden kann. Der gesamte Prozess vom Auswählen der Daten bis hin zur Gewinnung von Wissen wird als *Knowledge Discovery in Databases* (KDD) [FPS96b] bezeichnet. Der KDD Prozess umfasst alle Schritte vom Auswählen der Daten, z. B. aus einer Datenbank, bis hin zur Evaluation der Analyse-Ergebnisse. In der Abbildung 2.1 sind alle Schritte des KDD Prozesses zu sehen. Dabei ist zu sehen, dass vor dem

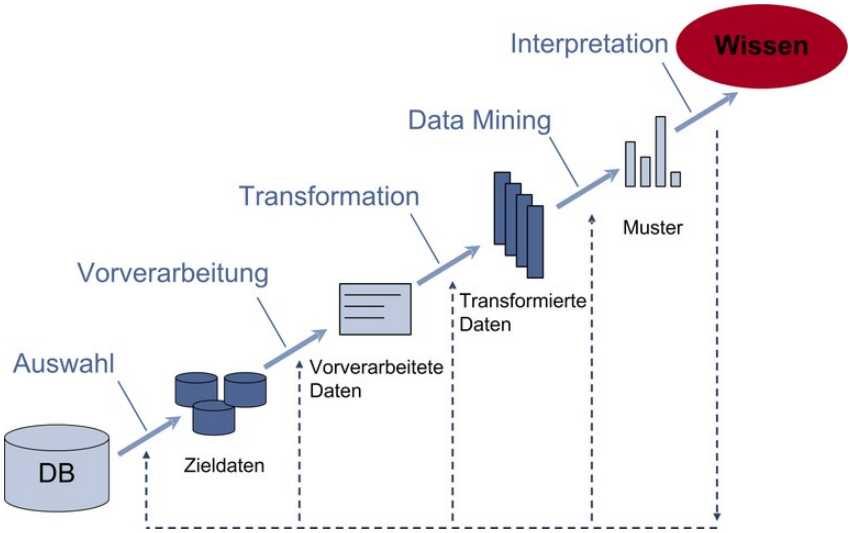


Abbildung 2.1.: Übersicht über die Schritte des KDD Prozesses [Cha16]

Data Mining Schritt zunächst Daten ausgewählt werden müssen. Diese werden anschließend vor-verarbeitet, was z. B. heißen kann, dass sie von Ausreißern befreit werden oder, dass die Daten aggregiert werden. Daraufhin werden die Daten in ein passendes Format für die Analyse umgewandelt. Das Data Mining repräsentiert dabei die eigentliche Analyse und wird in Abschnitt 2.1 thematisiert. In Abschnitt 2.2 wird eingehender auf ein spezifisches Data Mining-Verfahren, das Clustering, eingegangen. Für dieses spezifische Verfahren werden in Abschnitt 2.3 drei Algorithmen vorgestellt. Die Möglichkeiten, die es zur Evaluation dieser

Algorithmen gibt, wird in Abschnitt 2.4 dargelegt. In Abschnitt 2.5 wird auf verwandte Arbeiten eingegangen.

2.1. Data Mining

Das Data Mining beschreibt das Extrahieren von nützlichen Mustern aus großen Datenmengen [HPK11]. Dabei sind nicht alle Muster, die beim Data Mining entstehen nützlich. Daher wird es meist auch mehrfach mit unterschiedlichen Algorithmen und/ oder Parametern auf der selben Datenmenge ausgeführt. Denn wenn die Muster, die am Ende der Analyse gewonnen werden, unzureichend bzw. unbrauchbar sind, kann zu einem beliebigen vorherigen Schritt zurückgekehrt werden, um so an bessere Muster zu gelangen. Das kann beispielsweise durch schrittweises Ändern der Algorithmen oder der Parameter geschehen. Hier stößt das Data Mining auch an seine Grenzen, da unter anderem oftmals nicht klar ist, was die Ursachen für ein ungenügendes Ergebnis der Analyse sind. Dafür gibt es zahlreiche Möglichkeiten, zum einen Ursachen, die unabhängig vom Data Mining Schritt selbst auftreten, wie zum Beispiel einen Fehler in der Vorverarbeitung, schlecht ausgewählte Daten oder eine ungenügende Evaluation der Ergebnisse. Zum anderen Ursachen, die auf das Data Mining zurück zu führen sind, wie zum Beispiel, dass das ausgewählte Verfahren nicht zum Hintergrundwissen und zum Ziel der Analyse passt, dass ein ungeeigneter Algorithmus ausgewählt wurde oder, dass für den Algorithmus ungeeignete Parameter ausgewählt wurden.

Typische Data Mining-Verfahren sind das *Clustering*, eine Klassifikation, eine Regression, eine Assoziationsanalyse, eine Ausreißer Erkennung oder eine statistische Analyse [FPS96a]. Generell lassen sich die Data Mining-Verfahren in überwachte und unüberwachte Verfahren aufteilen. Unüberwacht bedeutet dabei, dass nur die Daten selbst die Eingabe für das Verfahren sind und keine zusätzlichen Informationen. Bei überwachten Verfahren hingegen enthält die Eingabe zusätzliche Informationen zur Datenmenge, wie z. B. vordefinierte Klassen [HPK11]
.

2.2. Clustering

Das Clustering ist eine unüberwachte Methode, bei der versucht wird die Daten in Gruppen aufzuteilen, sodass sich die Daten innerhalb eines Clusters möglichst ähnlich sind (hohe *intra-cluster* Ähnlichkeit) und Daten in verschiedenen Clustern möglichst unähnlich (geringe *inter-cluster* Ähnlichkeit) sind. Generell kann zwischen *hartem* und *weichem* (oder auch *fuzzy*) Clustering unterschieden werden. Beim harten Clustering gehört jeder Datenpunkt zu genau einem Cluster oder zu keinem Cluster, falls es sich bei dem Punkt um einen Ausreißer handelt. Beim weichen Clustering hingegen kann ein Punkt zu keinem, einem oder mehreren Clustern gehören [JMF99]. Für hartes Clustering kann das *Clustering Problem* wie in Definition 2.2.1 formuliert werden.

Definition 2.2.1 (Clustering Problem)
Sei eine Menge von Daten $M = \{m_1, \ldots, m_n\}$ gegeben. Zusätzlich bezeichnet A die Menge der Punkte, die als Ausreißer identifiziert wurden. Das Clustering Problem besteht darin eine Partition $C = \{C_1, \ldots, C_k\}$ aus M zu erstellen, für die folgende Bedingungen erfüllt sind:

i) $\forall i \neq j : C_i \cap C_j = \emptyset$

ii) $\bigcup_{i=1}^{k} C_i = M \setminus A$

iii) $\forall i : C_i \neq \emptyset$

Sollte ein Algorithmus keine Ausreißer behandeln, dann gilt $A = \emptyset$.

Eine Lösung des Clustering-Problems ist nicht zwangsläufig eine optimale Lösung. Die meisten Clustering-Algorithmen berechnen auch nicht die optimale Lösung, da das Finden einer optimalen Lösung des NP-hart ist [ADHP09].

Des weiteren muss ein Maß definiert werden, welches angibt wie ähnlich sich zwei Datenpunkte sind. Dazu wird meist eine Distanzmetrik herangezogen, die die Ähnlichkeit zwischen zwei Datenpunkten beschreibt. Die Wahl der Distanzmetrik ist dabei eine wichtige Entscheidung, die getroffen werden muss. Je nach Anwendungsbereich der Daten und je nach Algorithmus eignet sich eine besser als eine andere. Im Folgenden werden einige bekannte Distanzmetriken genannt [GMW07]. Die gebräuchlichste Distanzmetrik ist die *euklidische Distanz*

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}. \tag{2.1}$$

Diese ist ein Spezialfall der *Minkowski-Metrik* mit $r = 2$

$$d(x, y) = \sqrt[r]{(\sum_{i=1}^{n} (x_i - y_i)^r)}. \tag{2.2}$$

Setzt man $r = 1$ in der Minkowski-Metrik, dann erhält man die *Manhattan-Metrik* (auch Block-Distanz genannt)

$$d(x, y) = \sum_{i=1}^{n} |x_i - y_i|. \tag{2.3}$$

Eine weitere beliebte Distanzmetrik ist die *Hamming-Distanz*

$$d(x, y) = \frac{1}{n} \sum_{i=1}^{n} \delta_i \text{ mit } \delta_i = \begin{cases} 1 & \text{falls } x_i \neq y_i \\ 0 & \text{sonst} \end{cases} \tag{2.4}$$

Ein Vorteil der euklidischen Distanz ist, dass sie gut geeignet ist für Daten, die kompakte und isolierte Cluster enthalten, jedoch neigt sie dazu nur sphärische Cluster zu erkennen [MJ96]. Ein genereller Nachteil der Minkowski-Metriken ist, dass dazu tendiert wird, dass die Attribute, die am größten skaliert sind, die anderen Attribute dominieren. Dies ist beispielsweise bei der Hamming-Distanz nicht der Fall, da die Skalierung der Attribute keine Auswirkung auf diese Distanzmetrik hat. Eine Übersicht über weitere bekannte Distanzmetriken wird in [CCT10] gegeben.

2.2.1. Anforderungen an das Clustering

Vor der Wahl eines Clustering-Algorithmus sollten Anforderungen an den Algorithmus gestellt werden. Mit Hilfe dieser Anforderungen kann ein geeigneter Algorithmus ausgewählt werden. Einige dieser Anforderungen können auf Grundlage der vorhandenen Daten erschlossen werden, andere sind allerdings erst während oder nach der Analyse erkennbar. Sollten sie bereits vor der Ausführung bekannt sein, also aus den Daten ablesbar sein, kann mit deren Hilfe ein geeigneter Algorithmus ausgewählt werden. Typische Anforderungen sind [HPK11]:

Skalierbarkeit Einige Algorithmen arbeiten sehr gut auf kleinen Datenmengen. Hat man jedoch eine Datenmenge, die mehrere Millionen Datenpunkte enthält, sollte der Algorithmus auf dieser in angemessener Zeit anwendbar sein.

Typen der Attribute Viele Algorithmen können nur mit numerischen Daten umgehen, besonders Algorithmen die Distanz-basiert sind. Allerdings ist es auch möglich, dass die Attribute der Daten andere Typen enthalten, wie z. B. binäre, kategorische Daten oder eine Kombination aus diesen.

Willkürlich geformte Cluster Algorithmen, die als Distanzmetrik die euklidische Distanz benutzen, neigen dazu nur sphärische Cluster mit gleicher Größe oder Dichte zu finden. Allerdings können Cluster auch willkürliche Formen mit unterschiedlicher Größer und Dichte annehmen.

Ausreißer Behandlung In vielen Datenmengen kommt es vor, dass fehlerhafte Daten vorhanden sind oder Daten, die Extremdaten sind und sich stark von den anderen Daten unterscheiden. In vielen Algorithmen werden die Ausreißer nicht extra behandelt, wodurch das Ergebnis verzerrt werden kann.

Wahl der Parameter Für die Ausführung der Algorithmen werden meist zusätzliche Parameter benötigt. Eine schlechte Wahl der Parameter kann dabei zu einem unbrauchbarem Ergebnis führen. Daher sollte abgewägt werden, welche Informationen über die Datenmenge bekannt sind und welche Parameter eines Algorithmus dadurch abgedeckt werden können.

Dimension der Daten Die Laufzeit vieler Algorithmen ist davon abhängig, wie viele Attribute die Daten enthalten. Bei einer Vielzahl von Attributen sollte daher der Algorithmus mit Sorgfalt ausgewählt werden.

Interpretierbarkeit und Nutzbarkeit Das Interpretieren des Ergebnisses stellt eine schwierige Aufgabe dar. Da die Clustering unüberwacht ist, gibt es bei der Analyse kein richtiges oder falsches Ergebnis. Es gibt nur sinnvolle oder weniger sinnvolle Ergebnisse, wodurch eine automatisierte Bewertung schwer möglich ist, da auch immer der Anwendungsbereich und das Ziel der Analyse mit berücksichtigt werden muss. Aus diesem Grund sollte auch der Algorithmus dementsprechend gewählt werden.

2.2.2. Klassen von Clustering-Algorithmen

Es gibt eine Vielzahl von verschiedenen Klassen von Clustering-Algorithmen. Zudem gibt es auch unterschiedliche Klassifikationen. Ein Beispiel für existierende Klassen von Clustering-Algorithmen wird in Abbildung 2.2 gegeben. Dabei werden zu jeder Klasse auch einige der

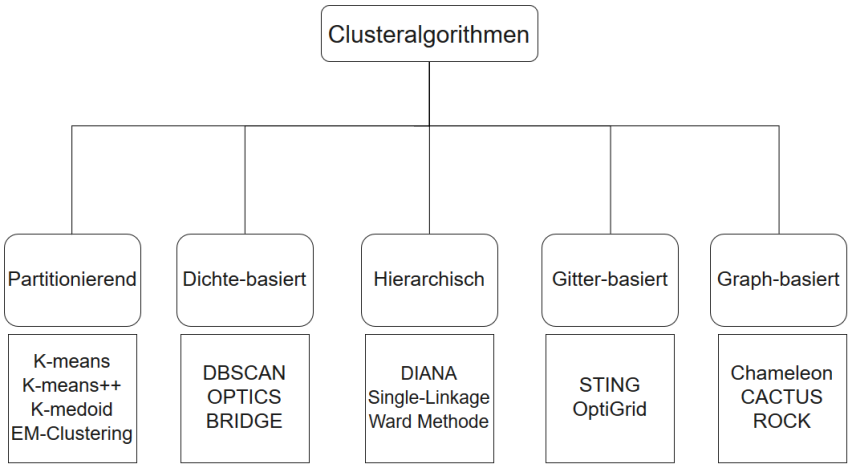


Abbildung 2.2.: Übersicht über bekannte Klassen von Clustering Algorithmen [FAT+14]

Vertreter genannt. Dies ist allerdings nur ein grober Überblick, denn es gibt zahlreiche Verfahren, die sich entweder gar nicht in eine der gezeigten Klassen einordnen lassen oder die sich in mehrere Klassen einordnen lassen [Ber+06]. In mancher Literatur wird auch nur zwischen partitionierenden und hierarchischen Verfahren unterschieden [JD88]. In diesem Abschnitt werden die partitionierenden und die dichte-basierten Verfahren vorgestellt, da diese im Laufe dieser Arbeit genauer untersucht werden.

Partitionierende Verfahren

Partitionierende Clustering Algorithmen versuchen aus einer Menge von Datenpunkten k Partitionen zu erstellen. Jede dieser Partitionen repräsentiert dabei ein Cluster. Meist wird k dabei vom Benutzer gewählt. Zunächst wird eine initiale Partition erstellt und diese wird dann versucht iterativ zu ändern, indem man die Clusterzugehörigtkeit der jeweiligen Punkte

verändert wird [JMF99]. Ein Beispiel für einen partitionierenden Clustering Algorithmus ist der K-means, der in Abschnitt 2.3.1 vorgestellt wird.

Dichte-basierte Verfahren

Dichte-basierte Verfahren beruhen darauf, ein Cluster wachsen zu lassen, solange die umliegende Dichte bestimmte Kriterien erfüllt. Das heißt z. B., dass sich in einer bestimmten Region eine Mindestanzahl an Punkten befinden muss. Ein Beispiel für einen dichte-basierten Algorithmus ist der DBSCAN Algorithmus, der in Abschnitt 2.3.3 vorgestellt wird. Der generelle Vorteil dieser Verfahren ist, dass Cluster mit willkürlichen Formen erkannt werden können und das sie Ausreißer erkennen können [Est09]. Allerdings sind diese Verfahren dafür meist mit einem höheren Berechnungsaufwand verbunden.

2.3. Clustering-Algorithmen

In diesem Abschnitt werden drei Clustering-Algorithmen vorgestellt. Einmal der weit verbreitete partitionierende Algorithmus K-means und der dichte-basierte Algorithmus DBSCAN, sowie der K-means++, welcher eine Erweiterung des K-means darstellt und sich nur in der Initialisierung unterscheidet. Des weiteren werden diese Algorithmen mit einander verglichen.

2.3.1. K-means

Der K-means [Mac67] Algorithmus ist ein weit verbreiteter Algorithmus, der erstmals von Macqueen (1967) formuliert wurde und zu den partitionierenden Algorithmen gehört. Zunächst erstellt er eine Anfangspartition die dann iterativ verändert wird. In jeder Iteration wird versucht die Partition so zu ändern, dass eine Fehlerfunktion minimiert wird. Dabei ist die Fehlerfunktion davon abhängig welche Distanzmetrik benutzt wird.

Wird die euklidische Distanzmetrik benutzt, dann ist diese Fehlerfunktion die *quadratische Fehlerfunktion*. Sei $C = \{C_1, \ldots, C_k\}$ eine Menge von Clustern, dann lautet die quadratische Fehlerfunktion wie folgt:

$$E(C) = \sum_{i=1}^{k} \sum_{m_i \in C_i} d(m_i, cen_i)^2 \tag{2.5}$$

Dabei bezeichnet cen_i den Zentroiden des Clusters $C_i = \{m_1, \ldots, m_n\}$ und lässt sich mit der Formel 2.6 berechnen.

$$cen_i = \frac{1}{n} \sum_{i=1}^{n} m_i \tag{2.6}$$

Der Zentroid steht dabei als Zentrum des Clusters repräsentativ für das Cluster. Die Zentroide werden zu Beginn des Algorithmus zufällig gewählt. Anschließend folgt eine Iteration aus zwei Schritten. Zunächst kommt der Schritt der Datenzuweisung. Dabei wird jeder Datenpunkt dem Cluster des Zentroiden mit dem geringsten Abstand zugewiesen. Schritt zwei ist die Neuberechnung der Zentroide. Hier werden die Zentroide mit Hilfe der Formel aus Gleichung (2.6) neu berechnet. Zusammengefasst läuft der Algorithmus folgendermaßen ab:

1. Wähle zufällig k Datenpunkte als Zentroide.
2. Berechne für jeden Punkt den Zentroid, der den geringsten Abstand zu diesem Punkt hat und füge den Punkt zum Cluster dieses Zentroids hinzu.
3. Berechne die Cluster Zentren neu mit der Formel 2.6.
4. Falls sich in Schritt 3 mind. ein Zentroid geändert hat, dann gehe zu 2. Andernfalls: Stopp;

Der Pseudocode des Algorithmus wird in Algorithmus 2.1 gegeben.

Beispiel In Abbildung 2.3 wird ein Beispiel für einen Durchlauf des K-means Algorithmus gegeben mit $k = 2$. Im Schritt a) Sind zunächst nur die Datenpunkte zu sehen. In Schritt b)

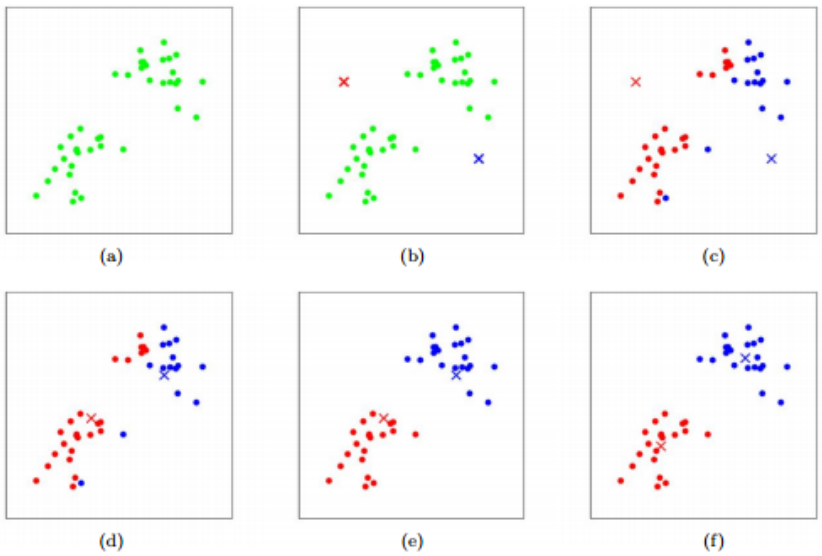


Abbildung 2.3.: Beispiel für die Ausführung des K-means Algorithmus [Pie13]

werden die Zentroide zufällig bestimmt. Anschließend werden in Schritt c) alle Punkte dem Zentroid zugewiesen, der den kleinsten Abstand zu diesem Punkt hat. Daraufhin werden in Schritt d) die Zentroide mit der Formel 2.6 neu berechnet. In Schritt e) werden dann wiederum die Punkte dem nächstgelegenen Zentroid zugewiesen, ehe in Schritt f) die Zentroide neu berechnet werden.

Algorithmus 2.1 K-means Algorithmus

```
1:  procedure K-MEANS(numberOfClusters, points)
2:      finish ← false;
3:      while (!finish) do
4:          clearClusters();
5:          oldCentroids ← getCurrentCentroids();
6:          ASSIGNCLUSTER();
7:          CALCULATECENTROIDS();
8:          newCentroids ← getCurrentCentroids();
9:
10:         totalCentroidsDistance ← getCentroidDistances(oldCentroids, newCentroids)
11:         finish ← (totalCentroidsDistance == 0) ;
12:     end while
13: end procedure
14:
15: procedure ASSIGNCLUSTER
16:     min ← ∞;
17:     nearestCluster ← empty cluster;
18:     for all point ∈ points do
19:         min ← ∞;
20:         for all cluster ∈ clusters do
21:             distance ← getDistance(point, cluster.getCentroid());
22:             if (distance < min) then
23:                 min ← distance;
24:                 nearestCluster ← cluster;
25:             end if
26:         end for
27:         nearestCluster.addPoint(point);
28:     end for
29: end procedure
30:
31: procedure CALCULATECENTROIDS
32:     for all cluster ∈ clusters do
33:         sum ← [0, . . . , 0];
34:         clusterPoints ← cluster.getPoints();
35:         for all (point ∈ clusterPoints) do
36:             sum ← (sum + point.getElements)/clusterPoints.size;        // Vektor Addition
37:         end for
38:         cluster.getCentroid().setElements(sum);
39:     end for
40: end procedure
```

Die Berechnung der initialen Zentroide erfolgt zufällig, wobei nicht eindeutig festgelegt ist, was zufällig heißt. Nach Lloyd [Llo82] werden zufällig beliebige Punkte als initiale Zentroide bestimmt. MacQueen [Mac67] hingegen schlägt vor, zufällige Punkte aus der Datenmenge als initiale Zentroide zu wählen. Es existieren mittlerweile mehrere Möglichkeiten die initialen Zentroide zu bestimmen [CKV13]. Die Komplexität des Algorithmus beträgt $O(k \cdot n \cdot t)$. Dabei ist n die Anzahl der Datenpunkte, k die Anzahl der Cluster und t die Anzahl der Iterationen. In den meisten Anwendungsfällen und besonders bei größeren Datenmengen gilt, dass $k < n$ und $t < n$, somit kann er in den meisten Fällen mit $O(n)$ abgeschätzt werden [HPK11]. Dies macht ihn für große Datenmengen sehr effizient.

Einschränkungen

Der Algorithmus konvergiert zwar gegen ein Minimum der quadratischen Fehlerfunktion, allerdings wird nur garantiert, dass er gegen ein lokales Minimum konvergiert und nicht gegen ein globales. Lokal bedeutet dabei, dass es zwar ein Minimum in der Funktion ist, es ist aber nicht das kleinste Minimum der Funktion. Global wiederum bedeutet, dass es das kleinste Minimum ist und es auch kein kleineres gibt. Des Weiteren muss die Anzahl der Cluster vor der Ausführung des Algorithmus bekannt sein. In der Praxis steht man allerdings oft vor dem Problem, dass nicht genug Informationen über die Daten vorhanden sind, um zu entscheiden, wie viele Cluster am Ende entstehen soll. Eine Möglichkeit dies zu umgehen ist, dass der Algorithmus mehrmals mit unterschiedlichen k's ausgeführt wird. Am Ende aller Durchläufe wird das k bzw. der Durchlauf genommen, bei dem der beste Wert für eine vorher ausgewählte Metrik vorhanden ist.

Ein weiterer Punkt ist, dass der Algorithmus Ausreißer nicht behandelt. Sollte einer der Punkte einen großen Abstand zu allen anderen Punkten haben, verschiebt sich somit auch der Zentroid in die Richtung des Ausreißers. Hinzu kommt, dass der Algorithmus keine sphärischen Cluster erkennen kann und von der Wahl der initialen Zentroide abhängig ist. Durch die initiale Wahl der Zentroide kann die Anzahl der Iterationen stark variieren und das Ergebnis geändert werden.

2.3.2. K-means++

Der K-means++ [AV07] Algorithmus ist ein partitionierender Algorithmus, der sich nur in der Wahl der Zentroide vom K-means unterscheidet. In der Initialisierung der Zentroide wird versucht, dass Punkte, die weiter entfernt sind von einem Zentroid auch mit einer höheren Wahrscheinlichkeit als neuer Zentroid gewählt wird. Dazu wird die minimale Distanz zum nächstgelegenen Zentroid bestimmt, die mit $D(m_i)$ für einen Punkt m_i bezeichnet wird. Dann wird für jeden Punkt die gewichtete Wahrscheinlichkeitsfunktion bestimmt, mit deren Hilfe

der nächste Zentroid ausgewählt wird. Die Wahrscheinlichkeitsfunktion für einen Punkt m_i sieht folgendermaßen aus:

$$P(m_i) = \frac{D(m_i)^2}{\sum_{m_j \in M} D(m_j)^2} \tag{2.7}$$

Diese Wahrscheinlichkeitsfunktion wird immer, nachdem ein Zentroid ausgewählt wurde, für jeden Punkt neu berechnet. Punkte, die bereits als Zentroid ausgewählt wurden erhalten somit die Wahrscheinlichkeit 0, da sie den Abstand 0 zu sich selbst haben. Der komplette Initialisierungsschritt läuft folgendermaßen ab:

1. Sei $M = \{m_1, \ldots, m_n\}$ die Menge der Datenpunkte. Bestimme zufällig ein $m_i \in M$ als Zentroid.
2. Berechne für jedes $m_i \in M$ die minimale Distanz $D(m_i)$.
3. Wähle nun zufällig den nächsten Zentroid $m_j \in M$ mit einer Wahrscheinlichkeit von $P(m_j)$ aus.
4. Wiederhole Schritt 2 und 3 bis k Zentroide ausgewählt wurden.

Die Einschränkungen und Vorteile des K-means++ sind die gleichen wie die des K-means Algorithmus. Die Komplexität des Algorithmus ist ebenfalls die Gleiche. Die Initialisierung des K-means++ liegt zwar in $O(k \cdot n)$ und nicht in $O(k)$ wie beim K-means, allerdings bleibt die Gesamtkomplexität damit gleich, da $O(k \cdot n) + O(k \cdot n \cdot t) = O(k \cdot n \cdot t)$. Obwohl man annehmen könnte, dass der K-means++ im Allgemeinen länger braucht als der K-means, konnte gezeigt werden, dass bessere Laufzeiten und auch bessere Ergebnisse erzielt werden konnten [AV07]. Zudem wurde gezeigt, dass Ergebnis nicht mehr so stark vom Zufall abhängt.

2.3.3. DBSCAN

Der Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [EKS+96] Algorithmus ist ein dichte-basierter Algorithmus. Dieser Algorithmus versucht Punkte, die in einer bestimmten Region eine hohe Dichte aufweisen, zu einem Cluster zusammenzufassen. Dabei muss ein Cluster eine bestimmte Anzahl von Punkten haben, die vorher festgelegt werden muss. Sollte hingegen eine Region ein niedrige Dichte haben, ist das ein Anzeichen für Ausreißer. Der Radius des Bereiches, in dem untersucht wird wie viele Punkte in diesem Bereich liegen muss ebenfalls vorher festgelegt werden. Im DBSCAN Kontext werden Ausreißer auch als *Noise* bezeichnet, weshalb diese Bezeichnung hier übernommen wird. Die Menge der Noise im DBSCAN Kontext wird in Definition 2.3.7 formal definiert. Da der DBSCAN dichte-basiert ist und nicht zentroid-basiert, kann er willkürliche geformte Cluster erkennen. Um den Algorithmus formulieren zu können, werden zunächst notwendige Definitionen eingeführt, die sich an [EKS+96] orientieren.

Definitionen

Definition 2.3.1 (Epsilon- Nachbarschaft)
Die Epsilon-Nachbarschaft eines Punktes p, bezeichnet mit $N_\varepsilon(p)$, ist definiert durch

$$N_\varepsilon(p) = \{q \in M \mid d(q,p) \leq \varepsilon\}, \tag{2.8}$$

wobei M die Menge der zu clusternden Datenpunkte ist.

Die Epsilon Nachbarschaft eines Punktes beinhaltet somit alle Punkte, die einen Abstand kleiner oder gleich ϵ zu diesem Punkt haben. Solch ein Punkt wird zudem als *Kernobjekt* bezeichnet, falls die ε-Nachbarschaft des Punktes mindestens $minPts$ Punkte enthält. Formal ausgedrückt bedeutet das:

Definition 2.3.2 (Kernobjekt)
Ein Punkt p heißt Kernobjekt, falls

$$|N_\varepsilon(p)| \geq MinPts. \tag{2.9}$$

Dabei bezeichnet $MinPts$ die minimale Anzahl an Punkten, die in einem Cluster liegen müssen.

Definition 2.3.3 (Direkt dichte-erreichbar)
Ein Punkt p heißt direkt dichte-erreichbar von einem Punkt q falls

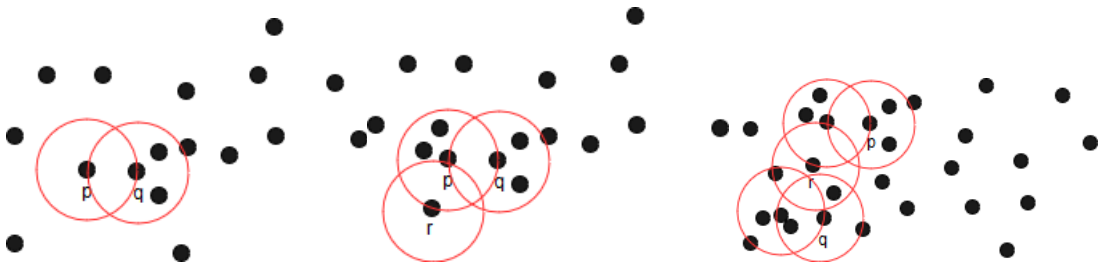
1. *$p \in N_\varepsilon(q)$*
2. *q ist ein Kernobjekt*

Dabei bezeichnet $MinPts$ die minimale Anzahl an Punkten, die in einem Cluster liegen müssen.

Definition 2.3.4 (dichte-erreichbar)
Ein Punkt p heißt dichte-erreichbar von einem Punkt q, falls es eine Sequenz von Punkten $p = p_1, p_2, \ldots, p_i = q$ gibt, so dass p_l direkt dichte-erreichbar ist von p_{l+1} $\forall l = 1, \ldots, i-1$.

Definition 2.3.5 (dichte-verbunden)
Zwei Punkte p und q heißen dichte-verbunden, falls es einen Punkt r gibt, so dass p dichte-erreichbar ist von r und q dichte-erreichbar ist von r.

Da die Definition von dichte-erreichbar auf der Definition von direkt dichte-erreichbar aufbaut, ist jeder Punkt der direkt-dichte erreichbar ist auch dichte-erreichbar, allerdings nicht umgekehrt. Es ist sogar so, dass dichte-erreichbar die *transitive Hülle* von direkt-dichte erreichbar ist. Das bedeutet, dass alle Punkte, die nur über die Transitivität in der direkt dichte-erreichbar Relation zueinander stehen, nicht aber direkt in Relation zueinander, stehen trotzdem direkt in Relation zueinander bezüglich der dichte-erreichbar Relation. Konkret bedeutet das, wenn p und q direkt dichte-erreichbar sind und q und r direkt dichte-erreichbar sind, dann müssen p und r nicht zwangsläufig direkt dichte-erreichbar sein. Allerdings sind p und r dann dichte-erreichbar. Des weiteren ist direkt-dichte erreichbar asymmetrisch. Dichte-Erreichbarkeit wiederum ist symmetrisch. Dies soll im Folgenden durch ein Beispiel verdeutlicht werden.



(a) direkt dichte-erreichbar **(b)** dichte-erreichbar **(c)** dichte-verbunden

Abbildung 2.4.: Beispiel zur Verdeutlichung von direkt dichte-erreichbar, dichte-erreichbar und dichte-verbunden

Beispiel In Abbildung 2.4 wird ein Beispiel zur Verdeutlichung der Definitionen gegeben. Der Parameter ε wird durch die roten Kreise gegeben. Diese stellen die ε-Umgebung eines Punktes dar. Für dieses Beispiel ist $MinPts = 3$. In (a) ist p direkt dichte-erreichbar von q, da $p \in N_\varepsilon(q)$ und $|N_\varepsilon(q)| = 4 > MinPts$. Allerdings ist q nicht direkt dichte-erreichbar von p. In (b) ist ein Beispiel für die Dichte-Erreichbarkeit gegeben. Dabei ist r dichte-erreichbar von q, weil p direkt dichte-erreichbar ist von q und r dichte-erreichbar ist von p. In der Abbildung (c) sind p und q dichte-verbunden, da sowohl p, als auch q dichte-erreichbar sind von r.

Ein dichtebasiertes Cluster kann dann wie in Definition 2.3.6 definiert werden.

Definition 2.3.6 (Cluster)
Sei M eine Menge von Datenpunkten. Ein Cluster $\emptyset \neq C \subseteq M$ erfüllt folgende Bedingungen:

1. $\forall p, q \in M : p \in C$ *und q dichte-erreichbar von p, dann ist* $q \in C$ *(Maximalität).*
2. $\forall p, q \in C :$ p *und* q *sind dichte-verbunden (Verbundenheit).*

Aus der Definition Definition 2.3.6 folgt, dass jedes Cluster C mindestens $MinPts$ Punkte enthalten muss. Denn sei $p \in C$, dann existiert ein $q \in C$, sodass p und q dichte-verbunden sind (Verbundenheit). Allerdings muss q dann mindestens die Kernobjekt Eigenschaft erfüllen, womit mindestens $MinPts$ in der Epsilon-Nachbarschaft von q liegen. Diese Punkte sind alle (direkt) dichte-erreichbar von q und somit auch in C (Maximalität).

Noise kann jetzt als Menge von Punkten definiert werden, die zu keinem Cluster gehören.

Definition 2.3.7 (Noise)
Sei M eine Menge von Datenpunkten und $C_i \subseteq M, i = 1, \ldots, k$ die zugehörigen Cluster. Dann ist

$$Noise = \{p \in M \mid \forall i = 1, \ldots, k : \ p \notin C_i \ \} \tag{2.10}$$

Da die Definitionen eingeführt sind, kann der Algorithmus beschrieben werden. Dieser wird in Algorithmus 2.2 in Pseudocode dargestellt. Zunächst sind die clusterId's aller Punkte auf

UNCLASSIFIED gesetzt. Dann wird mit einem zufälligen Punkt p begonnen. Für diesen Punkt p wird dann die Funktion *expandCluster* in Zeile 4 aufgerufen. In dieser Funktion werden zunächst alle Punkte gesucht, die in $N_\varepsilon(p)$ liegen und in der Menge *seeds* gespeichert. In Zeile 14 wird geprüft, ob es sich bei dem Punkt um ein Kernobjekt handelt. Falls nicht wird der Punkt als Noise identifiziert und es wird *false* zurückgegeben. Dadurch wird die clusterId nicht erhöht, was bedeutet, dass kein neues Cluster entsteht. Sollte der Punkt jedoch ein Kernpunkt sein, entsteht ein neues Cluster, welches mindestens die aktuellen Punkte aus der Menge seeds enthält. Aus diesem Grund wird in Zeile 19-21 die clusterId aller Punkte aus der Menge seeds auf die momentane clusterId gesetzt. Ab Zeile 24 wird dann für jeden Punkt aus der Menge seeds, außer den Punkt p, der an die Funktion übergeben wurde, geschaut, welcher der Punkte ein Kernobjekt ist. Sollte einer dieser Punkte ein Kernobjekt sein, wird seine clusterId auf die momentane clusterId gesetzt (Zeile 32). Hat Punkt zudem davor noch die clusterId UNCLASSIFIED gehabt haben, dann wird er zu der Menge seeds hinzugefügt (Zeile 29 - 31). Sobald die Menge seeds leer wird, wird die while-Schleife verlassen und es wird *true* zurückgegeben. Daraufhin wird die clusterId in der Funktion DBSCAN um eins erhöht (Zeile 5), da ein neues Cluster entstanden ist. Dies wird dann für jeden Punkt aus der Datenmenge *points* wiederholt.

Nach Definition 2.3.6 kann es passieren, dass zwei Cluster, die sich sehr nahe sind, zu einem Cluster verschmolzen werden. Daher wird der Abstand zwischen zwei Clustern C_1, C_2 als $d(C_1, C_2) = min\{d(p, q) \mid p \in C_1,\ q \in C_2\}$ definiert. Die Zwei Cluster C_1, C_2 sollen anschließend verschmolzen werden, falls $d(C_1, C_2) \leq \varepsilon$.

Da der Algorithmus dichte-basiert ist, ist es mit ihm möglich willkürlich geformte Cluster zu erkennen. Hinzu kommt, dass das Ergebnis des Algorithmus nicht durch Ausreißer verzerrt wird, da er Ausreißer behandelt. Anders als beim K-means muss vor der Ausführung des Algorithmus die Anzahl der Cluster nicht vorher bekannt sein.

Einschränkungen

Ein Nachteil den DBSCAN gegenüber dem K-means hat, ist, dass die Laufzeit von DBSCAN bei einer naiven Implementierung $O(n^2)$ beträgt. Wird eine räumliche Indexstruktur (R*-Baum) benutzt, dann kann die Komplexität auf $O(n\ logn)$ verbessert werden, was allerdings immer noch schlechter wäre als beim K-means. Dadurch wäre der Algorithmus bei sehr großen Mengen nicht die ideale Wahl. Hinzu kommt, dass der Parameter ϵ (sollte er einmal gesetzt worden sein) fest ist. Das bedeutet, dass Cluster, die eine unterschiedliche Dichte aufweisen nicht erkannt werden können. Ein weiterer Punkt sind die Parameter des Algorithmus. Zum einen sind zwei Parameter, die bestimmt werden müssen und zum anderen beeinflusst die Wahl der Parameter den Algorithmus sehr stark. Wird das Epsilon beispielsweise zu groß gewählt erhält man unter Umständen nur ein Cluster. Wird es zu klein gewählt, ist es möglich, dass kein Cluster entsteht. Das gleiche Problem erhält man auch mit den *Min Points*. Um dem entgegen zu wirken, kann eine Heuristik verwendet werden. Diese wird im Folgenden erläutert.

Algorithmus 2.2 DBSCAN Algorithmus

```
1:  procedure DBSCAN(eps, MinPts, points)
2:      clusterId ← 0
3:      for all p ∈ points do
4:          if expandCluster(eps, MinPts,clusterId, p, points) then
5:              clusterId ← clusterId + 1
6:          end if
7:      end for
8:  end procedure
9:
10: procedure expandCluster(eps, MinPts,clusterId currentPoint, points)
11:     UNCALSSIFIED ← -1
12:     NOISE ← -2
13:     seeds ← epsNeighbourhood(currentPoint, eps)
14:     if |seeds| < MinPts then
15:         currentPoint.setClusterId(NOISE)
16:         return false
17:     end if
18:
19:     for all seed ∈ seeds do
20:         seed.setClusterId(clusterId)
21:     end for
22:
23:     seeds ← seeds \ {currentPoint}
24:     while seeds ≠ ∅ do
25:         p ← seeds.getFirstPoint()
26:         result ← epsNeighbourhood(p, eps)
27:         if |result| ≥ MinPts then
28:             for all point ∈ result do
29:                 if point.clusterId == UNCLASSIFIED then
30:                     seeds ← seeds ∪ {point}
31:                 end if
32:                 point.setClusterId(clusterId)
33:             end for
34:         end if
35:         seeds ← seeds \ {p}
36:     end while
37:     return true
38: end procedure
```

Eigenschaft	K-means/++	DBSCAN
Komplexität	$O(n \cdot k \cdot t)$	$O(n^2)$
Parameter	k: Anzahl der Cluster	ϵ: Dichte Parameter MinPts: Mindestanzahl Punkte in einem Cluster
Erkennbare Formen von Clustern	Sphärische Cluster	Willkürliche Formen
Ausreißerbehandlung	Nein	Ja
Cluster Kriterium	Quadratische Fehlerfunktion	Verschmelze Punkte die dichte-erreichbar sind zu einem Cluser
Abhängig von Reihenfolge der Datenpunkte	Initialisierung der Zentroide von Reihenfolge der Daten abhängig	Nein
Typ der Attribute	Numerisch	Numerisch

Tabelle 2.1.: Vergleich der Algorithmen K-means, K-means++ und DBSCAN

Wahl der Parameter

Für die Wahl der Parameter des DBSCAN Algorithmus gibt es eine einfache Heuristik. Diese benutzt dazu den *k-Nearest-Neighbour*[1] Algorithmus. Dazu muss zunächst der Parameter k bestimmt werden. In der Literatur wird vorgeschlagen, diesen mit $k = 2 \cdot d - 1$ zu wählen, wobei d die Dimension der Datenpunkte ist. Der $MinPts$ Parameter wird auf k gesetzt. Für die Bestimmung von ε wird zunächst für jeden Punkt der Abstand zum k nächsten Nachbarn berechnet. Alle Distanzen werden dann absteigend sortiert in einem Graphen ausgegeben. Anschließend wird der erste Punkt bestimmt, bei dem ein „Ellenbogenpunkt“ zu erkennen ist. Sei dazu zunächst $F_k : M \to \mathbb{R}$ eine Funktion mit

$$F_k(p) = d(p, q_k) \text{ wobei } q_k \text{ der } k \text{ nächste Nachbar von } p \text{ ist.}$$

Anschließend wird $F_k(M)$, wobei M die Menge der Datenpunkte ist, in absteigender Ordnung sortiert und in einem zwei-zweidimensionalen Graph dargestellt. Danach wird $\varepsilon = F_k(z_0)$ gesetzt, wobei z_0 der erste Ellenbogenpunkt des Graphen von F_k. Der einfachste Weg diesen Ellenbogenpunkt zu bestimmen ist, indem der Graph von F_k einem Benutzer gezeigt wird und dieser den Punkt anschließend abliest [Est09].

Abschließend wird in Tabelle 2.1 ein Vergleich der Algorithmen gegeben. Dabei sind K-means und K-means++ in einer Spalte, da es zwischen den beiden keine Unterschiede, bis auf die Initialisierung gibt.

[1] http://scholarpedia.org/article/K-nearest_neighbor

2.4. Evaluationsmetriken

Für die Bewertung des Ergebnisses eines Clustering-Algorithmus wird normalerweise ein menschlicher Experte herangezogen. Dieser braucht geeignete Metriken, um das Ergebnis zu interpretieren. Dabei gibt es eine Vielzahl von Metriken, die benutzt werden können und es gibt auch nicht immer die ideale Metrik, die benutzt wird. Denn je nach Fokus des Analysten und Ziel der Analyse können die herangezogen Metriken von Fall zu Fall variieren. Daher ist auch trotz der Metriken ein menschlicher Experte zur Auswertung der Analyse notwendig.

2.4.1. Klassifikation der Metriken

Es gibt drei Arten von Metriken, die unterschieden werden. Die *internen Metriken*, die *externen Metriken* und die *relativen Metriken* [JD88].

Interne Metriken

Die internen Metriken haben zum Ziel, Aussagen über die interne Struktur der Cluster zu treffen. Dabei wird versucht die Kompaktheit, die Separation oder beides zu beurteilen. Die Kompaktheit beschreibt wie groß die intra-Cluster Ähnlichkeit ist, also wie groß die Abstände der Punkte innerhalb eines Clusters sind. Es gibt dabei mehrere Möglichkeiten diese zu berechnen. Im Folgenden wird die Kompaktheit als Abstand der Punkte eines Clusters zum zugehörigen Zentroid gewählt.

Definition 2.4.1 (Kompaktheit)
Sei $C = \{C_1, \ldots, C_k\}$ eine Menge von Clustern. Dann ist die Kompaktheit definiert als

$$Kompaktheit(C) = \sum_{i=1}^{k} \sum_{m_i \in C_i} d(cen_i, m_i) \tag{2.11}$$

Die Separation hingegen beschreibt die inter-Cluster Ähnlichkeit, also wie groß der Abstand zwischen den Clustern ist. Dabei gibt es auch hier mehrere Möglichkeiten diese zu berechnen. In dieser Arbeit werden die Abstände der Zentroide der Cluster als Maß genommen.

Definition 2.4.2 (Separation)
Sei $C = \{C_1, \ldots, C_k\}$ eine Menge von Clustern. Dann ist die Separation von C definiert als

$$Separation(C) = \sum_{i=1}^{k} \sum_{j=1}^{k} d(cen_i, cen_j) \tag{2.12}$$

Dabei beschreibt cen_i jeweils den Zentroid des Clusters C_i.

Index Name	Formel
Rand Index	$R = \frac{a+d}{M}$
Jaccard Koeffizient	$J = \frac{a}{a+b+c}$
Folkes und Mallows Index	$FM = \sqrt{\frac{a}{a+b} * \frac{a}{a+c}}$
Precision	$P = \frac{a}{a+c}$
Recall	$Rec = \frac{a}{a+d}$
F-Measure	$F = 2 * \frac{1}{(\frac{1}{P} + \frac{1}{Rec})}$

Tabelle 2.2.: Übersicht über externe Metriken (aus [GMW07])

Externe Metriken

Bei externen Metriken werden Daten zur Evaluation benutzt, die nicht für das Clustering benutzt wurden. Das heißt , dass externe Daten verwendet werden. Dafür werden vor-klassifizierte Ergebnisse gebraucht. Anschließend werden die vor-klassifizierten Ergebnisse mit den Ergebnissen des Algorithmus verglichen.

Zunächst bezeichnen wir mit $C = \{C_1, \ldots, C_n\}$ die Menge der Cluster, die nach Ausführung eines Clustering Algorithmus entstehen. Mit $P = \{P_1, \ldots, P_k\}$ wird die Partition bezeichnet, mit der C verglichen werden soll. Anschließend werden Variablen a, b, c und d eingeführt, die für einige externe Metriken benutzt werden.

- a: Bezeichnet die Anzahl der Paare von Punkten, die im selben Cluster in C, als auch im selben Cluster in P liegen (True Positiv).
- b: Bezeichnet die Anzahl der Paare von Punkten, die im selben Cluster in C liegen, aber in P in verschiedenen Clustern liegen (True negativ).
- c: Ist das Gegenstück zu b. Die Anzahl der Punkte von Paaren, die im selben Cluster in C sind, allerdings in P in verschiedenen Clustern (False Positiv).
- d: Anzahl der Paare von Punkten, wo die Punkte in verschieden Clustern aus C liegen, als auch in P (False Negativ).

Sei M die Anzahl der Paare, die aus den n Datenpunkten gebildet werden können. Dann ist

$$M = a + b + c + d = \frac{n(n-1)}{2} \tag{2.13}$$

Aufbauend darauf werden in Tabelle 2.2 einige bekannte externe Metriken dargestellt.

Algorithmus 2.3 Algorithmus zur Bestimmung der Parameter

```
1: for n_c = n_min to n_max do
2:     for i=1 to r do
3:         P ← Wähle Parameterkonfiguration aus
4:         C ← Führe Algorithmus mit den Parametern P aus
5:         q_i ← Berechne Metrik für C
6:     end for
7:     Wähle besten Wert aus {q_1, ..., q_r} aus.
8: end for
```

Relative Metriken

Das Ziel der relativen Metriken, ist, eine geeignete Parameterkonfigurationen für die Algorithmen zu finden. Es werden mehrere Durchläufe der Algorithmen mit unterschiedlichen Parametern durchgeführt. Am Ende werden dann die Ergebnisse anhand einer geeigneten Metrik bewertet und so das Ergebnis, welches den besten Wert der Metrik aufweist, ausgewählt. Dazu wird meist eine interne Metrik benutzt.

Ist die Anzahl der Cluster ein Parameter des Algorithmus, dann kann der in Algorithmus 2.3 gezeigte Pseudocode zur Bestimmung der Parameter genutzt werden. Dabei bezeichnet n_c den Parameter der Anzahl der Cluster. Des Weiteren muss der Algorithmus, der minimale und der maximale Wert der für die Anzahl der Cluster benutzt wird, so wie die Anzahl der Durchläufe r als Eingabe erhalten.

Als Beispiel könnte $r = 10$ Durchläufe gewählt werden und $n_{min} = 1$ und $n_{max} = 50$ gesetzt werden. Als Metrik könnte die Separation oder die Kompaktheit herangezogen werden.

2.5. Verwandte Arbeiten

Für dem K-means Algorithmus wurden zahlreiche Erweiterungen vorgestellt. Die meisten davon versuchen entweder, die Initialisierung der Zentroide zu optimieren [CKV13] oder durch zusätzliche Heuristiken, wie zum Beispiel durch eine minimale/ maximale Anzahl der Cluster [Jai10]. Des Weiteren wurden Optimierungen vorgenommen in dem die Anzahl der Vergleiche, die beim K-means gemacht werden zu reduzieren [Phi02] oder einen *kd*-Baum als Datenstruktur zu verwenden, um so eine effizientere Berechnung zu ermöglichen [PM99]. Steinbach et al. [STK+03] schlagen den *bisecting K-means* vor, der eine hierarchische Version des K-means darstellt. Jain [Jai10] geben eine Übersicht über weitere bekannte Erweiterungen des K-means.

Mexicano et al. [MRC+16] schlagen einen Ansatz vor, der ebenfalls versucht mit einer Heuristik die Ausführungszeit des K-means zu reduzieren. Die *early stop heuristic* geht davon aus, dass die erste Verschiebung der Zentroiden, die nach der Initialisierung stattfindet am größten

ist. Ausgehend davon wird dieser Wert als Vergleichswert genommen und wird mit D_{max} bezeichnet. Terminiert wird anschließend, sobald die Verschiebung der Zentroide kleiner oder gleich 5% von D_{max} beträgt. Für diesen Ansatz konnte eine Reduktion der Ausführungszeit von bis zu 87% mit einem Qualitätsverlust von nur 2,46% erreicht werden.

In dieser Arbeit wird ebenfalls versucht ein Konvergenzkriterium zu finden, das auf geeigneten Metriken basiert. Dazu werden geeignete Metriken untersucht und ausgewählt. Allerdings wird ein Konzept entwickelt, bei dem die Metriken auf die Zwischenberechnungen des K-means zu greifen, um so eine effizientere Berechnung zu ermöglichen. Es sollen zudem stabilere Ergebnisse werden, als dies bei Mexicano et al. [MRC+16] der Fall war. Hinzu kommt, dass diese Metriken zudem dazu genutzt werden, um Zeitpunkte für eine Visualisierung von Zwischenergebnissen zu finden.

3. Konzept zur Berechnung der Metriken in Teilschritten

In diesem Kapitel wird ein Konzept für eine effiziente Berechnung der Metriken vorgestellt. Dazu werden in Abschnitt 3.1 Anforderungen an die Klassen von Metriken gestellt und die Klasse, die die Anforderungen am ehesten erfüllt, ausgewählt. Anschließend werden konkrete Metriken dieser Klasse vorgestellt. In Abschnitt 3.3 werden dann Optimierungsansätze vorgestellt, um so eine effizientere Berechnung zu ermöglichen.

3.1. Anforderungen

In diesem Abschnitt wird eine Klasse von Metriken ausgewählt, die weiter untersucht wird. Um eine geeignete Klasse auszuwählen, werden erst Kriterien benötigt, um die Klassen von Metriken miteinander vergleichen zu können. Dazu werden Anforderungen an die Metriken gestellt. Anschließend wird verglichen welche Klasse der Metriken die Anforderungen am ehesten erfüllt und diese wird dann weitergehend untersucht. Die Anforderungen, die aufgrund der Ziele an die Metriken gestellt werden, sind:

Eignung für große Datenmengen (A1) Die Berechnung der Metriken sollte für große Datenmengen mit möglichst geringer Komplexität möglich sein.

Möglichkeit Zwischenergebnisse auszuwerten (A2) Um geeignete Interaktionspunkte für eine Visualisierung zu finden, sollten die Metriken leicht interpretierbare Werte für Zwischenergebnisse der Algorithmen liefern.

Auswertung nur anhand der Cluster (A3) Das Ergebnis des Clusterings sollte ohne zusätzliche Daten ausgewertet werden können.

Entdeckung von neuen Clustern (A4) Sollten durch den Algorithmus Cluster entdeckt werden, die in vorherigen Durchläufen nicht erkannt wurden, sollten die Metriken das Ergebnis deswegen nicht schlechter bewerten, nur weil es nicht einem bereits bekannten Ergebnis entspricht.

Anwendbarkeit auf ausgewählte Algorithmen (A5) Die Metriken sollten auf die Algorithmen, die im Fokus dieser Arbeit stehen, anwendbar sein.

Anforderung	Interne Metriken	Externe Metriken
A1	wird nicht erfüllt	wird nicht erfüllt
A2	wird erfüllt	wird erfüllt
A3	wird erfüllt	wird nicht erfüllt
A4	wird erfüllt	wird nicht erfüllt
A5	wird erfüllt	wird erfüllt

Tabelle 3.1.: Darstellung der Anforderungen und welche Klasse von Metriken die jeweilige Anforderung erfüllt

Da die relativen Metriken andere Metriken nutzen, um geeignete Parameter heraus zu finden, werden nur die internen und die externen Metriken untersucht.

Die Anforderung A1 wird sowohl für die externen, als auch für die internen Metriken nur unzureichend erfüllt. Da die internen Metriken die Kompaktheit und/oder die Separation betrachten, müssen dazu die Zentroide berechnet werden. Da für die Berechnung der Zentroide alle Punkte einmal betrachtet werden müssen ergibt sich daraus mindestens eine Laufzeit von $O(n)$. Für die externen Metriken muss das Ergebnis der Clustering mit dem des vorklassifizierten Datensatzes verglichen werden. Dadurch müssen auch hier alle Punkte einmal betrachtet werden. Die internen Metriken haben jedoch den Vorteil, dass der K-means beispielsweise die Zentroide berechnet. Wenn die Metriken auf diese Zentroide zugreifen können, beträgt die Berechnung der Zentroide für die Metriken $O(1)$ statt $O(n)$ ohne, dass dadurch die Berechnungszeit für den K-means erhöht wird. Die Anforderung A2 wird für beide Metriken erfüllt. Allerdings ist diese Anforderung eher vom Algorithmus abhängig, als von der Metrik. So ist es für den DBSCAN eher weniger sinnvoll Zwischenergebnisse auszuwerten, da eventuell noch nicht jeder Punkt in ein Cluster eingeteilt wurde. Somit wären die Metriken für die Zwischenergebnisse nicht sehr sinnvoll. Die Anforderung A3 wird nur für die internen Metriken erfüllt, da die externen Metriken zusätzlich zu dem Ergebnis der Clustering noch einen vor-klassifizierten Datensatz brauchen. Da dieser mit dem Ergebnis der Clustering verglichen wird, werden neue Cluster, die bei der Clustering entstehen, aber nicht im vor-klassifizierten Datensatz enthalten sind, schlechter bewertet. Aus diesem Grund wird auch die Anforderung A4 für nicht für die externen Metriken erfüllt. Für die internen Metriken hingegen wird die Anforderung erfüllt. Die Anforderung A5 wiederum wird für beide Klassen von Metriken erfüllt, da sich beide grundsätzlich auf die Algorithmen K-means, K-means++ und DBSCAN anwenden lassen.

In Tabelle 3.1 wird zusammengefasst welche Anforderungen die internen und welche die externen Metriken erfüllen. Die internen Metriken erfüllen dabei 4 der 5 Anforderungen. Für die Anforderung A1 können zudem noch Optimierungen in Anlehnung an den Algorithmus durchgeführt werden. Die externen Metriken hingegen erfüllen 2 der 5 Anforderungen. Aus diesem Grund werden die internen Metriken für die weitere Untersuchung ausgewählt.

3.2. Interne Metriken

In diesem Abschnitt werden einige bekannte interne Metriken vorgestellt, die für die Experimente genutzt werden. Zu jeder Metrik wird auch die Komplexität für die Berechnung der Metrik mit angegeben, unter der Annahme, dass keine Optimierungen für die Berechnung vorgenommen werden. Wie die Metriken und die Algorithmen angepasst werden können, um eine effizientere Berechnung zu ermöglichen, wird in Abschnitt 3.3 eingehender untersucht. Anschließend soll experimentell bestimmt werden, welche der hier vorgestellten Metriken sich dazu eignen einen Zeitpunkt für eine frühzeitige Terminierung der Algorithmen zu finden und welche sich dafür eignen, geeignete Zeitpunkte für eine Visualisierung zu bestimmen.

Für jede Metrik wird in diesem Abschnitt auch vorgestellt wie die Komplexität zur Berechnung dieser Metrik aussieht. Dafür wird davon ausgegangen, dass die Berechnung aller Zentroiden in $O(r \cdot n)$ möglich ist, wobei r die Dimension der Datenpunkte beschreibt. Ebenso wird angenommen, dass die Berechnung des Abstands zweier Punkt in $O(r)$ möglich ist. Dadurch wäre die Berechnung der Zentroide in $O(r \cdot n)$ möglich.

In diesem Abschnitt wird mit $C = \{C_1, \ldots, C_k\}$ eine Menge von Clustern und mit cen_i der Zentroid des Clusters C_i bezeichnet. Zudem sei $n = \sum_{i=1}^{k} |C_i| = |M|$ die Anzahl der Punkte in der Datenmenge M.

3.2.1. Sum of Squared Errors

Die Sum of Squared Errors (SSE) wurde bereits in Abschnitt 2.3.1 als Fehlerfunktion eingeführt, die der K-means Algorithmus versucht zu minimieren. Sie eignet sich aber auch als interne Metrik, die die Kompaktheit der Cluster beschreibt. Sie berechnet für jedes Cluster den quadrierten Abstand der Punkte in diesem Cluster zum Zentroiden des Clusters und summiert diese Werte dann auf. Berechnen lässt sie sich mit der Formel 3.1.

$$E(C) = \sum_{i=1}^{k} \sum_{m_i \in C_i} d(m_i, cen_i)^2 \tag{3.1}$$

Komplexität Die Berechnung der Zentroide liegt in $O(r \cdot n)$ und anschließend wird für jedes Cluster und jeden Punkt der quadratische Abstand bestimmt. Dies beträgt ebenfalls $O(r \cdot n)$. Somit ergibt sich insgesamt für die gesamte Komplexität $O(2 \cdot r \cdot n) = O(r \cdot n)$.

3.2.2. Dunn Index

Der Dunn Index (DI) [Dun74] kombiniert die Kompaktheit und die Separation. Dabei wird der minimale Separationswert ins Verhältnis zum maximalen Kompaktheitswert gesetzt. Die Berechnung des Dunn Index wird in Formel 3.2 gegeben.

$$DI(C) = \frac{min\{Separation(C_i, C_j) \mid C_i, C_j \in C\}}{max\{Kompaktheit(C_i) \mid C_i \in C\}} \tag{3.2}$$

Je höher der Dunn Index ist, desto besser ist die Qualität der Cluster. Es konnte festgestellt werden, dass Rauschen das Ergebnis verzerrt, da er den minimalen Separations- und den maximalen Kompaktheiswert betrachtet [HBV02]. Bezdek und Pal [BP98] schlagen daher 3 generalisierte Dunn Indizes vor, die robuster gegen Ausreißer sind.

Komplexität Für die Kompaktheit und die Separation muss zunächst der Zentroid berechnet werden. Die Berechnung aller Zentroiden liegt in $O(r \cdot n)$. Für die Separation müssen zusätzlich noch die Distanzen zwischen allen Paaren von Zentroiden berechnet werden, dies sind $\frac{k \cdot (k-1)}{2}$ Paare. Also wären das zusätzlich noch $O(k^2)$ Berechnungen die hinzukommen, wobei jede dieser Berechnung in $O(r)$ liegt. Insgesamt ergibt sich somit eine Komplexität von $O(r \cdot n + r \cdot k^2) = O(r \cdot (n + k^2)$ [VCH10].

3.2.3. Silhouetten Koeffizient

Der Silhouetten Koeffizient (SK) [Rou87] versucht eine Aussage über die Kompaktheit und die Separation zu treffen, sowohl für einzelne Punkte, als auch für Cluster. Dazu sei p ein beliebiger Punkt aus einem Cluster C. Mit a wird die durchschnittliche Distanz von p zu den anderen Punkten innerhalb des Clusters C bezeichnet. Somit berechnet sich a mit der Formel

$$a(p) = avg\{d(p, p') \mid p \neq p' \in C\} \tag{3.3}$$

Des weiteren wird b als das Minimum der durchschnittlichen Distanzen von p zu Punkten in anderen Clustern definiert. Betrachten wir also alle Cluster, dann ist

$$b(p) = \min_{C \neq C'}\{avg\{d(p, p') \mid p' \in C'\}\} \tag{3.4}$$

Die Definition der Silhouette eines Punktes p wird in der Formel 3.5 gegeben.

$$SK(p) = \frac{b(p) - a(p)}{max\{a(p), b(p)\}} \tag{3.5}$$

Der Silhouetten-Koeffizient für ein Cluster wäre dann der Durchschnitt der Summe von $s(p)$ für alle Punkte p aus der Datenmenge. Der Wert des Koeffizienten liegt immer zwischen -1 und 1, wobei ein höherer Wert für eine bessere Qualität der Cluster steht.

Komplexität Für die Berechnung des Silhouetten-Koeffizienten muss zunächst die Distanz für alle Paare von Punkten berechnet werden. Sind die Punkte im selben Cluster, dann wird die Distanz für die Berechnung von a benutzt. Andernfalls wird sie für die Berechnung von b benutzt. Die Anzahl aller Paare von Punkten ist $\frac{n*(n-1)}{2}$. Die Berechnung hierfür liegt in $O(n^2)$ und jede dieser Abstandsberechnung liegt in $O(r)$, womit sich dafür eine Komplexität von $O(n^2 \cdot r)$ ergibt. Sobald alle Distanzen verfügbar sind, liegt die Berechnung von b in $O(k \cdot r)$ für einen Punkt. Für alle Punkte wäre das somit $O(n \cdot k \cdot r)$. Da a und b bekannt sind, liegt die Berechnung von $s(p)$ in $O(1)$. Da dies aber für jeden Punkt berechnet werden muss, kommt eine Laufzeit von $O(n)$ hinzu. Insgesamt ergibt sich somit eine Komplexität von $O(n^2 \cdot r + n \cdot k \cdot r + n) = O(n^2 \cdot r)$, da $k \leq n$ gilt [VCH10].

3.2.4. Davies-Bouldin Index

Der Davies-Bouldin Index (DBI) [DB79] misst die Ähnlichkeit zwischen jedem Cluster und dem Cluster, der ihm am ähnlichsten ist. Dazu wird mit R_{ij} der Grad der Ähnlichkeit zwischen zwei Clustern C_i und C_j bezeichnet. Dieser basiert auf dem Streuungsgrad und der Unähnlichkeit. Der Streuungsgrad repräsentiert dabei die Kompaktheit in einem Cluster und wird in Definition 3.2.1 definiert.

Definition 3.2.1 (Streuungsgrad)
Sei C_i ein Cluster, dann ist der Streuungsgrad s_i definiert als

$$s_i = \frac{1}{k_i} \sum_{x \in C_i} d(x, cen_i) \tag{3.6}$$

Dabei ist k_i die Anzahl der Punkte im Cluster C_i.

Die Unähnlichkeit hingegen beschreibt die Separation zweier Cluster und wird in Definition 3.2.2 definiert.

Definition 3.2.2 (Unähnlichkeitsmaß)
Seien C_i und C_j zwei Cluster. Dann ist die Unähnlichkeit zwischen C_i und C_j definiert als

$$d_{ij} = d(cen_i, cen_j) \tag{3.7}$$

Die Berechnung des Davies-Bouldin Index wird in der Formel 3.8 dargestellt.

$$DBI = \frac{1}{k} \sum_{i=1}^{k} max_{j \neq i} R_{ij} \tag{3.8}$$

R_{ij} kann dabei frei gewählt werden, muss allerdings folgende Bedingungen erfüllen:

i) $R_{ij} \geq 0$

ii) $R_{ij} = R_{ji}$

iii) Wenn $s_i = 0$ und $s_j = 0$ dann $R_{ij} = 0$

iv) Wenn $s_j > s_k$ und $d_{ij} = d_{ik}$ dann $R_{ij} > R_{ik}$

v) Wenn $s_j = s_k$ und $d_{ij} < d_{ik}$ dann $R_{ij} > R_{ik}$

Ein R_{ij} das die Bedingungen erfüllt, wäre $R_{ij} = \frac{s_i + s_j}{d_{ij}}$.

Desto geringer der Davies-Bouldin Index ist, desto besser ist die Qualität der Cluster.

Komplexität Zur Berechnung des Davies-Bouldin Index müssen zunächst die Zentroide für den Streuungsgrad berechnet werden. Diese Berechnung liegt in $O(r \cdot n)$. Sind die Zentroide bekannt, ist zur Berechnung aller Streuungsgrade ein Aufwand von $O(r \cdot n)$ nötig. Für das Unähnlichkeitsmaß müssen alle Paare von Zentroiden berechnet werden, was einer Komplexität von $O(r \cdot k^2)$ entspricht. Anschließend muss der Term $max_{j \neq i} R_{ij}$ berechnet werden, was einem Aufwand von $O(k)$ entspricht, da die Werte zur Berechnung eines R_{ij} bekannt sind und es insgesamt k R_{ij}'s gibt, von denen das Maximum bestimmt werden muss. Da diese Berechnung k-mal ausgeführt werden muss ist die Komplexität hierfür $O(k^2)$. Insgesamt ergibt sich somit für die Berechnung des Index eine Komplexität von $O(r \cdot n + r \cdot k^2 + k^2) = O(r \cdot (n + k^2))$ [VCH10].

3.2.5. Coggins-Jain Index

Der Coggins-Jain Index (CJ) [CJ85] berechnet für jeden Cluster den Quotienten aus Separation und Kompaktheit und wählt dann den minimalen Wert. Der Index berechnet sich wie in Formel 3.9 beschrieben, wobei $Separation(C_i) = min\{Separation(C_i, C_j) \mid C_j \in C\}$.

$$CJ(C) = min \left\{ \frac{Separation(C_i)}{Kompaktheit(C_i)} | C_i \in C \right\} \tag{3.9}$$

Ein höherer Wert des Index bedeutet auch eine bessere Qualität der Cluster.

Komplexität Für die Berechnung des Coggins-Jain Index müssen sowohl für die Kompaktheit, als auch für die Separation alle Zentroide berechnet werden, was in $O(r\dot{n})$ liegt. Angenommen es wird ein Cluster betrachtet, dann muss für dieses Cluster die Separation zu allen anderen Clustern berechnet werden. Das sind $k-1$ Cluster und da die Zentroide bekannt sind, wäre diese Berechnung in $O(k)$ möglich. Für die Kompaktheit, müsste der Abstand von allen Punkten zum Zentroid bestimmt werden, was einem Aufwand von $O(r \cdot n)$ entspricht. Da die Berechnung für die Separation und die Kompaktheit für alle Cluster durchgeführt werden müssen, kommt

für beide Berechnung der Faktor k hinzu. Damit ergibt sich insgesamt eine Komplexität von $O(k^2 + k \cdot r \cdot n)$.

3.2.6. Anzahl geänderter Punkte

Diese Metrik beschreibt die Anzahl der Punkte die ihre Clusterzugehörigkeit seit der letzten Iteration verändert haben. Damit ist gemeint, dass es eine Änderung gibt, wenn ein Punkt seine Clusterzugehörigkeit ändert. Das heißt man berechnet die Änderungsrate in dem man die Anzahl der Punkte nimmt, die ihre Clusterzugehörigkeit seit der letzten Iteration geändert haben und setzt diese ins Verhältnis zu der Anzahl aller Punkte.

Sei $M = \{m_1, \ldots, m_n\}$ eine Menge an Datenpunkten und sei $c_i(p)$ für $i > 0$ eine Funktion, die einem Punkt p sein zugehöriges Cluster in der Iteration i zuordnet. Dann lässt die Anzahl der geänderten Punkte in der Iteration i mit der Formel Gleichung (3.10) berechnen.

$$\#GP_i = |\{p \in M \mid c_i(p) \neq c_{i-1}(p) \}| \tag{3.10}$$

Für den K-means und den K-means++ Algorithmus wird für die erste Iteration $\#GP_1 = |M|$ gesetzt, da in der ersten Iteration alle Punkte zum ersten Mal einem Cluster zugewiesen werden.

Komplexität Für die Berechnung der Komplexität der Anzahl der geänderten Punkte, wird vorausgesetzt, dass es möglich ist, auf die Funktion $c_i(p)$ und $c_{i-1}(p)$ in $O(1)$ zuzugreifen. Generell ist es nicht möglich auf die Berechnung der letzten Iteration zuzugreifen ohne den Algorithmus leicht abzuändern. Für diesen Fall wird allerdings davon ausgegangen, dass dies möglich ist, da der Algorithmus in Abschnitt 3.3 entsprechend angepasst wird. Anschließend müssen die Funktionen für alle Punkte ausgewertet werden. Damit ergibt sich für die Berechnung eine Komplexität von $O(n)$.

3.3. Optimierungsansätze für die Berechnung

In diesem Abschnitt werden Optimierungsansätze zur Anpassung der Metriken und Algorithmen untersucht, um so eine effizientere Berechnung der Metriken zu ermöglichen. Da die Metriken in jeder Iteration berechnet werden sollen, ist eine Komplexität von $O(1)$ oder maximal $O(k)$ wünschenswert. Dies wird versucht zu erreichen, in dem die Berechnungen, die im K-means durchgeführt werden gespeichert und für die Metriken zur Verfügung gestellt werden. Dies soll allerdings keine großen Auswirkungen auf die Laufzeit des Algorithmus haben. Das bedeutet, dass nur zusätzliche Werte gespeichert oder Berechnungen in konstanter Zeit hinzugefügt werden. Im Folgenden werden einige Ansätze vorgestellt, um die Berechnung der Metriken zu optimieren. In Algorithmus 3.1 wird die Umsetzung der Ansätze dargestellt, wobei

nur die *assignCluster()* Methode des K-means abgeändert wurde. Der Rest des Algorithmus ist wie in Algorithmus 2.1 beschrieben.

Berechnung der Zentroide (OA1) Der erste Ansatz ist, dass die Berechnung der Zentroide im K-means auch den Metriken zur Verfügung gestellt wird. Denn die meisten internen Metriken verwenden auch diese Zentroide zur Berechnung. Dadurch ist es für die Metriken möglich einen Zentroiden in $O(1)$ zu berechnen, wodurch die Berechnung aller Zentroiden für die Metriken in $O(k)$ möglich ist. Dieser Ansatz wurde in Algorithmus 3.1 nicht dargestellt, da dem Cluster ein Feld *centroid* gegeben wurde, welches dann bei der Berechnung der Metriken durch eine Methode *getCentroid()* in einem Cluster abgerufen werden konnte. Ein Unterschied zur ursprünglichen Berechnung der Metriken ist nicht zu erwarten, da sich die Berechnung nicht ändert, sondern lediglich die Art und Weise wie den Metriken der Zentroid zur Verfügung gestellt wird.

Berechnung der Kompaktheit (OA2) Der zweite Ansatz zielt darauf ab, die Komplexität für die Berechnung der Kompaktheit zu reduzieren. Selbst wenn die Berechnung der Zentroide in $O(k)$ möglich ist, muss für die Kompaktheit trotzdem noch der Abstand jedes Punktes innerhalb eines Clusters mit dem Zentroiden berechnet werden. Dies ergibt dann eine Komplexität von $O(k\dot{n} \cdot r)$. Aus diesem Grund wird versucht die Kompaktheit nicht durch die Distanz vom Zentroiden zu allen Punkten innerhalb eines Clusters zu bestimmen, sondern nur die Distanz vom Zentroiden zu dem Punkt, der innerhalb des Clusters liegt, aber am weitesten vom Zentroiden entfernt ist. Dazu muss beim K-means Algorithmus, wenn die Punkte zu einem Zentroiden zugewiesen werden immer die Distanz gespeichert werden, wo die Distanz zum Zentroiden am größten ist. Dazu wäre eine zusätzliche Variable im K-means Algorithmus zu speichern und eine zusätzliche *if*-Abfrage wäre notwendig. Dies wird in Algorithmus 3.1 in den Zeilen 20 - 22 deutlich. In Zeile 20 wird abgefragt ob der Wert *min*, der den Abstand vom Punkt zum nächsten Zentroiden beschreibt, größer ist, als die momentan gespeicherte *maxPointDistance* für das Cluster. Falls ja wird der *maxPointDistance*-Wert des Clusters auf den Wert *min* gesetzt.

Zusätzlich muss in der Zeile 8 noch der Wert der *maxPointDistance* eines Clusters auf 0 gesetzt werden, da sonst fälschlicherweise der *maxPointDistance*-Wert der letzten Iteration größer ist, als der der jetzigen Iteration, obwohl der Punkt nicht mehr im Cluster enthalten ist. Der Vorteil dieses Ansatz ist, dass er sich für jede Metrik umsetzen lässt, die die Kompaktheit berechnet. Allerdings ist es dadurch möglich, dass die Metriken durch diesen Ansatz andere Werte annehmen als durch ihre ursprüngliche Berechnung. Wie stark die Unterschiede sind wird in Abschnitt 5.1 evaluiert.

Berechnung der Anzahl der geänderten Punkte (OA3) Der dritte Ansatz betrifft die Anzahl der geänderten Punkte. Dafür wird im K-means Algorithmus eine zusätzliche Variable eingeführt, die immer um eins hochgezählt wird, falls ein Punkt einem neuen Cluster zugewiesen wird. Auch hier bleibt die Komplexität des Algorithmus unverändert, da eine zusätzliche Zählervariable eingeführt wird und eine *if*-Abfrage, ob ein Punkt einem neuen Cluster zugewiesen wurde. Wenn ja wird der Zähler um eins erhöht. In

Ansatz	Betroffene Metriken	Komplexität	Optimierte Komplexität
OA1	DI, CJ, DBI, SSE	$O(n \cdot r)$	$O(k)$
OA2	DI, CJ, DBI, S	$O(n \cdot r)$	$O(k)$
OA3	$\#GP_i$	$O(n)$	$O(1)$
OA4	SSE	$O(n \cdot r)$	$O(1)$

Tabelle 3.2.: Übersicht der Optimierungsansätze mit den Metriken, die von dem Ansatz betroffen sind, so wie die Komplexität für den optimierten Ansatz und die ursprüngliche Komplexität

Algorithmus 3.1 wird dies in den Zeilen 16 - 18 verdeutlicht. In Zeile 16 wird abgefragt, ob das Cluster des Punktes aus der letzten Iteration mit dem neu zugewiesenen Cluster übereinstimmt. Falls dies nicht der Fall ist, wird die Variable *changedPoints* um eins erhöht (Zeile 17). Dadurch muss die Metrik nur diese Zählervariable abfragen und durch die Anzahl aller Punkte teilen, womit die Berechnung dann in $O(1)$ liegt.

Berechnung der SSE (OA4) Der vierte Ansatz betrifft die Sum of Squared Errors. Dazu wird ausgenutzt, dass während die Punkte einem Zentroid zugeordnet werden, die Distanz zwischen dem Punkt und seinem zugehörigen Zentroid bestimmt wird. Dieser Wert muss anschließend nur noch quadriert werden. Die notwendigen Schritte werden in Algorithmus 3.1 in der Zeile 15 verdeutlicht. Dabei wird die *sseDistance* gespeichert und der quadrierte Wert des Punktes zu seinem Zentroiden wird addiert. Zusätzlich muss die *sseDistance* vor jeder Iteration noch auf 0 gesetzt werden. Durch diesen Ansatz sollte die SSE nicht von der SSE mit der ursprünglichen Berechnung abweichen, da die Berechnung dieselbe bleibt. Jedoch wird durch diesen Ansatz die SSE der letzten Iteration berechnet und nicht der momentanen Iteration. Denn im K-means Algorithmus werden zuerst die Punkte einem Zentroiden zugeordnet und anschließend die Zentroide neu berechnet, womit sich nach der Neuberechnung der Zentroide auch eine neue SSE ergeben kann.

In Tabelle 3.2 wird eine Übersicht über die Optimierungsansätze gegeben, so wie die jeweilige Metrik, die von diesem Ansatz betroffen ist. Mit Komplexität ist die ursprüngliche Komplexität gemeint, die ohne den Optimierungsansatz notwendig ist. Die optimierte Komplexität beschreibt die Komplexität, falls der Optimierungsansatz für die Berechnung der Metriken genutzt wird.

Algorithmus 3.1 Optimierte AssignCluster Methode

```
1:  procedure ASSIGNCLUSTER
2:      min ← ∞;
3:      nearestCluster ← empty cluster;
4:      changedPoints← 0
5:      for all point ∈ points do
6:          min ← ∞;
7:          for all cluster ∈ clusters do
8:              cluster.setMaxPointDistance(0);
9:              distance ← getDistance(point, cluster.getCentroid());
10:             if (distance < min) then
11:                 min ← distance;
12:                 nearestCluster ← cluster;
13:             end if
14:         end for
15:         sseDistance ← sseDistance + min^2;                  // Umsetzung von OA4
16:         if (point.getCluster() != cluster) then
17:             changedPoints ← changedPoints + 1;              // Umsetzung von OA3
18:         end if
19:         nearestCluster.addPoint(point);
20:         if (min > cluster.maxPointDistance() then
21:             cluster.setmaxPointDistance(min);               // Umsetzung von OA2
22:         end if
23:         point.setCluster(cluster);
24:     end for
25: end procedure
```

4. Experimentelle Umsetzung

In diesem Kapitel wird auf die Umsetzung der Experimente eingegangen. Das Ziel der Experimente ist, herauszufinden, welche der in Abschnitt 3.2 vorgestellten Metriken sich für einen Zeitpunkt für eine frühzeitige Terminierung von Clustering-Algorithmen ohne großen Qualitätsverlust eignen. Des Weiteren soll erforscht werden, welche Metriken sich für die Bestimmung von Zeitpunkten zur Visualisierung von Zwischenergebnissen während der Ausführung von Clustering-Algorithmen eignen. In Abschnitt 4.1 wird auf den Aufbau und die Struktur des Prototypen eingegangen. In Abschnitt 4.2 wird erläutert welche Schritte in den Experimenten durchgeführt wurden und es wird auf die technischen Details eingegangen, die zur Umsetzung der Experimente genutzt wurden. Anschließend werden die Datensätze, die für die Experimente genutzt wurden in Abschnitt 4.3 beschrieben.

Hier kurz die Ziele der Experimente erklären. Hier auch iwo erwähnen, dass DBSCAN nicht gut geeignet für die Konvergenz.

4.1. Prototypische Implementierung

In diesem Abschnitt wird auf den Aufbau des Prototypen eingegangen, der im Rahmen dieser Arbeit entstanden ist. Als Programmiersprache wurde *Java* genutzt. In Abbildung 4.1 wird eine Übersicht über die Struktur der Pakete als UML Paketdiagramm gegeben. Im *Model* Paket sind die Implementierungen für einen Datenpunkt und ein Cluster enthalten. Der Ablauf der Experimente befindet sich im Paket *Evaluation*. Dieser wird in Abschnitt 4.2 genauer untersucht. Das Paket *Metrics* enthält die Implementierungen der Metriken, die in Abschnitt 3.2 vorgestellt wurden. Jede dieser Metrik erbt von der abstrakten Klasse *Metric* und überschreibt die Methode *calculate*, die als Eingabe eine Liste von Clustern bekommt und gibt den Wert der Metrik als *double* zurück. Für jede dieser Metrik wird auch immer die Zeit, die die zur Berechnung gebraucht wird, festgehalten. Zur Verwaltung der Metriken ist die Klasse *MetricHelper* vorhanden. Diese ist als *Singleton*[1] implementiert und enthält eine Map namens *metricsMap*, die als Schlüssel den Namen einer Metrik enthält und als Wert eine Instanz der jeweiligen Metrik-Klasse. Zudem enthält sie eine Methode *calculateAllMetricsWithMeasure*, die den Wert jeder Metrik berechnet und die benötigte Zeit für die Berechnung erfasst. Soll eine Metrik hinzugefügt werden, dann muss eine Klasse für diese Metrik erstellt werden, die von der

[1] http://campus.murraystate.edu/academic/faculty/wlyle/430/rc008-designpatterns_online.pdf

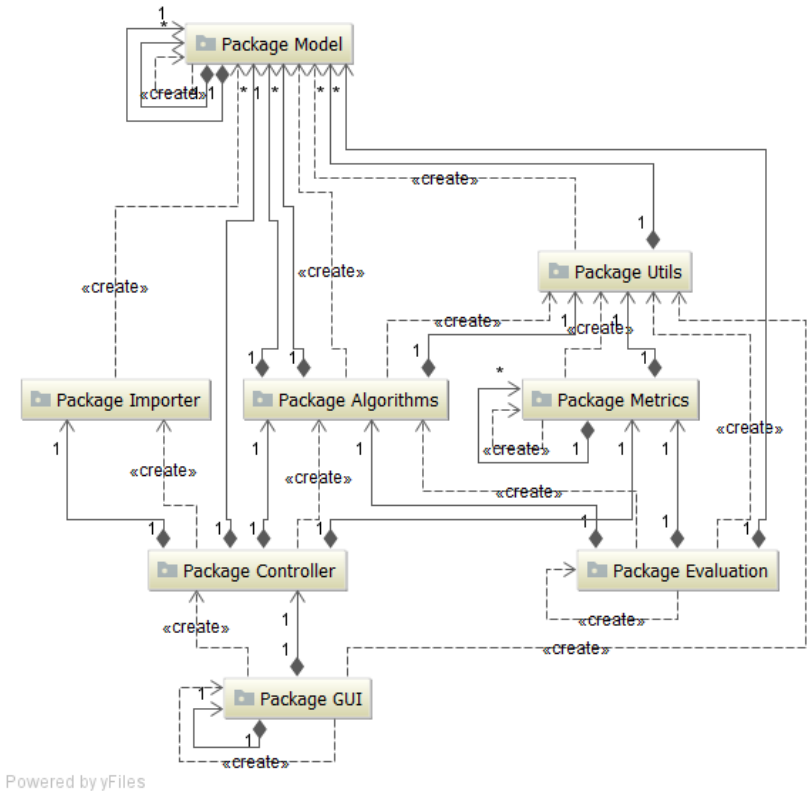


Abbildung 4.1.: UML Paketdiagramm der Pakete des Prototypen

Klasse *Metric* erbt und die Methode *calculate* überschreibt. Anschließend muss eine Instanz dieser Metrik-Klasse der *metricsMap* zusammen mit einem passenden Namen hinzugefügt werden. Diese Metrik wird dann ohne weitere Schritte in den Experimenten berechnet und in der graphischen Benutzeroberfläche angezeigt. Allerdings werden die Metriken nur für die Algorithmen K-means und K-means++ berechnet.

Im *Algorithms* Paket wurden die Clustering-Algorithmen K-means, K-means++ und DBSCAN implementiert. Diese Algorithmen erben alle von der abstrakten Klasse *Algorithm* und implementieren die Methoden *nextIteration* und *run*. Die Methode *nextIteration* gibt das Zwischenergebnis der nächsten Iteration des Algorithmus zurück. Die Methode *run* führt den Algorithmus solange aus, bis er terminiert und gibt das Endergebnis des Algorithmus zurück. Um einen Algorithmus hinzuzufügen, muss eine Klasse erstellt werden, die von der Klasse *Algorithm* erbt und genau diese beiden Methoden implementiert. Soll dieser Algorithmus auch in der graphischen Oberfläche ausgewählt werden können, muss zunächst ein Button für diesen Algorithmus in der Klasse *StartFrame* hinzugefügt werden. Anschließend muss ein Controller für diesen Algorithmus erstellt werden, der von der Klasse *Controller* erbt. Dieser Controller muss dann der Klasse *StartFrame* hinzugefügt werden. Einen Algorithmus für die Experimente

hinzuzufügen ist so ohne weiteres nicht möglich, da zunächst die Wahl der Parameter und welche Metriken genutzt werden sollen, geklärt werden muss.

Im Paket *GUI* ist die Implementierung der graphischen Benutzeroberfläche enthalten. Diese wurde als *Swing*-Anwendung umgesetzt. Für die Darstellung der Ergebnisse einer Clustering in Diagrammen wurde die Open-Sourcce Bibliothek *JFreeChart*[2] genutzt. In Abbildung 4.2 ist ein Beispiel für den Startbildschirm nach dem Generieren des Datensatzes „Gaussian mixture“. Der Datensatz wurde generiert in dem in der Menüleiste auf „Generieren“ und anschließend

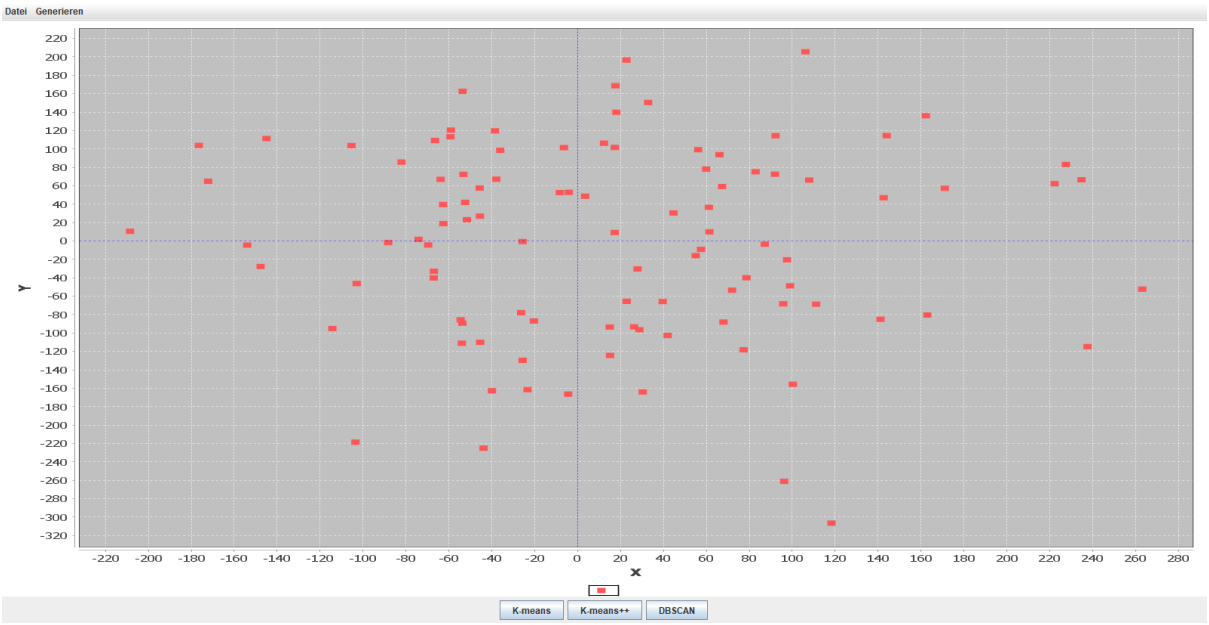


Abbildung 4.2.: Startbildschirm des Prototypen nach dem Generieren des Datensatzes „Gaussian mixture“

auf „Gaussian mixture“ geklickt wurde. Anschließend kann mittels einer der drei Buttons ein Algorithmus ausgewählt werden, der durchgeführt werden soll. Wird auf den Button *next iteration* geklickt, wird die nächste Iteration des Algorithmus ausgeführt und das Ergebnis dieser Iteration angezeigt. Dazu werden auch die Metriken angezeigt, die für das Zwischenergebnis berechnet wurden. Sollte der Algorithmus terminieren erscheint eine Meldung, dass der Algorithmus terminiert ist. In Abbildung 4.3 ist ein Beispiel für die Visualisierung der Zwischenergebnisse zusammen mit den ausgewählten Metriken gegeben. Diese Benutzeroberfläche ist nur indirekt Bestandteil der Arbeit, da sie nicht für die Experimente genutzt wurde, sondern nur zur Veranschaulichung der Algorithmen und dessen Zwischenergebnisse dienen sollte. Die Experimente befinden sich im Paket *Evaluation*. Dieses wird im Abschnitt 4.2 zusammen mit dem Ablauf der Experimente eingehender erläutert.

[2] http://www.jfree.org/jfreechart/

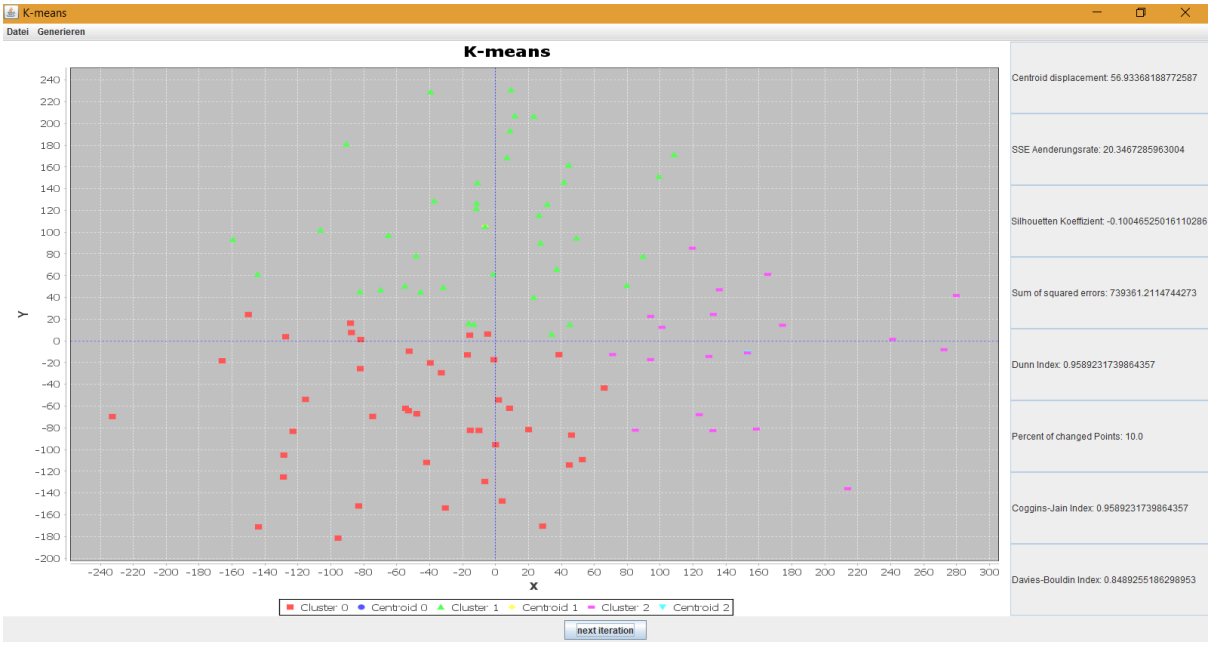


Abbildung 4.3.: Beispiel für das Anzeigen eines Zwischenergebnisses für den Algorithmus K-means

4.2. Versuchsaufbau

In diesem Abschnitt wird der Aufbau und der Ablauf der Experimente beschrieben. Die Experimente befinden sich im Prototypen im Paket *Evaluation*. Das Ziel der Experimente ist es, die implementierten Algorithmen auf vorher ausgewählten Datensätzen auszuführen. Welche Datensätze dafür genutzt wurden, wird in Abschnitt 4.3 beschrieben. Für jeden Algorithmus wurde jede Iteration mit der momentanen Ausführungszeit des Algorithmus, der jeweiligen Iteration, so wie den Metriken und deren Berechnungszeit festgehalten. Für eine besser Übersicht wurden diese Daten in eine CSV-Datei exportiert. Wie so eine Datei nach dem Exportieren aussieht wird in Abbildung 4.4 dargestellt. Dabei sind die Werte für alle Metriken, die in Abschnitt 3.2 vorgestellt wurden, für jede Iteration enthalten zusammen mit der Zeit die die Berechnung der Metriken mit Hilfe der Optimierungsansätze aus Abschnitt 3.3 benötigt haben. Da die Algorithmen K-means und K-means++ die Zentroide zufällig initialisieren, werden mehrere Durchläufe für die Algorithmen durchgeführt. Für jeden der beiden Algorithmen werden 10 Durchläufe durchgeführt. Für jeden Durchlauf wird eine CSV-Datei wie in Abbildung 4.4 erstellt. Dabei wurde in der ersten Spalte die Zeit des Algorithmus zur Zeit der Iteration, die in der zweiten Spalte steht, festgehalten. Die Spalten daneben enthalten die Werte der Metriken zur jeweiligen Iteration. Zusätzlich wurde noch für jede Metrik die Zeit festgehalten, die für ihre Berechnung benötigt wurde. Für den K-means und den K-means++ wurden zudem auch noch unterschiedliche k's festgelegt, da in der Praxis oft nicht bekannt ist, welches k das Optimale ist. Als untere Grenze wurde $k = 5$ gewählt und als obere Grenze $k = 50$. Zudem wurden nicht alle k's zwischen 5 und 50 ausprobiert, sondern immer nur in 5er Schritten.

algorithm time	iter	%SSE_i	Silhouett	SSE	DI	%GP_i	CJ	DBI	%SSE_i time	Silhouett	DI time	%GP_i tin	CJ time	DBI time
12 ms	1	#####	0,348	2,05E+11	0,582	0,862	0,761	1,03	0 ms	0 ms	0 ms	0 ms	0 ms	3 ms
22 ms	2	0,512	0,341	1,92E+11	0,557	0,094	0,760	0,98	0 ms	0 ms	0 ms	1 ms	0 ms	2 ms
29 ms	3	0,030	0,302	1,88E+11	0,511	0,048	0,642	0,96	0 ms	0 ms	0 ms	2 ms	0 ms	2 ms
37 ms	4	0,015	0,309	1,85E+11	0,483	0,034	0,620	0,96	0 ms	0 ms	0 ms	0 ms	0 ms	2 ms
43 ms	5	0,020	0,309	1,79E+11	0,522	0,057	0,638	0,86	0 ms	0 ms	0 ms	1 ms	0 ms	3 ms
48 ms	6	0,040	0,303	1,74E+11	0,568	0,044	0,671	0,79	0 ms	0 ms	0 ms	0 ms	0 ms	2 ms
56 ms	7	0,016	0,304	1,73E+11	0,575	0,017	0,684	0,79	0 ms	0 ms	0 ms	0 ms	0 ms	4 ms
62 ms	8	0,004	0,287	1,72E+11	0,574	0,011	0,695	0,79	0 ms	0 ms	0 ms	0 ms	0 ms	2 ms
68 ms	9	0,004	0,271	1,71E+11	0,584	0,018	0,721	0,79	0 ms	0 ms	0 ms	0 ms	0 ms	2 ms
73 ms	10	0,010	0,250	1,69E+11	0,595	0,023	0,762	0,77	0 ms	0 ms	0 ms	1 ms	0 ms	2 ms
78 ms	11	0,010	0,227	1,68E+11	0,595	0,020	0,751	0,77	0 ms	0 ms	0 ms	0 ms	0 ms	2 ms
83 ms	12	0,007	0,226	1,67E+11	0,599	0,020	0,771	0,77	0 ms	0 ms	0 ms	0 ms	0 ms	2 ms
88 ms	13	0,005	0,214	1,66E+11	0,609	0,017	0,751	0,78	0 ms	0 ms	0 ms	0 ms	0 ms	2 ms
92 ms	14	0,004	0,203	1,65E+11	0,622	0,017	0,740	0,80	0 ms	0 ms	0 ms	0 ms	0 ms	2 ms
99 ms	15	0,007	0,199	1,63E+11	0,661	0,024	0,786	0,83	0 ms	0 ms	0 ms	0 ms	0 ms	2 ms
104 ms	16	0,027	0,214	1,53E+11	0,732	0,052	0,733	0,86	0 ms	0 ms	0 ms	0 ms	0 ms	2 ms
109 ms	17	0,084	0,233	1,42E+11	0,790	0,056	0,678	0,87	0 ms	0 ms	0 ms	0 ms	0 ms	2 ms
115 ms	18	0,050	0,247	1,38E+11	0,826	0,031	0,649	0,89	0 ms	0 ms	0 ms	0 ms	0 ms	2 ms
121 ms	19	0,013	0,247	1,37E+11	0,839	0,020	0,629	0,90	0 ms	0 ms	0 ms	0 ms	0 ms	2 ms
127 ms	20	0,005	0,246	1,36E+11	0,824	0,013	0,616	0,90	0 ms	0 ms	0 ms	0 ms	0 ms	2 ms

Abbildung 4.4.: Beispiel für die Darstellung der exportierten Ergebnisse nach einem Durchlauf des K-means Algorithmus

Der Grund dafür ist, das wenn die Algorithmen für alle k's von 1 bis 50 ausgeführt werden würden, ein Algorithmus $10 \cdot 50 = 500$ mal ausgeführt werden müsste. Dies würde jedoch die Kapazitäten dieser Arbeit sprengen und wäre auch nicht praxistauglich.

Ausgeführt wurden die Experimente auf einer virtuellen Maschine auf einer *OpenStack*[3] Instanz. Als Betriebssystem wurde Ubuntu 14.04 genutzt mit 16GB Arbeitsspeicher, 8 Kernen und 80GB Festplattenspeicher.

4.3. Datensätze

In diesem Abschnitt werden die Datensätze beschrieben, die für die Evaluation genutzt wurden. Die Datensätze teilen sich dabei auf in reale und synthetische Datensätze.

4.3.1. Reale Datensätze

Die realen Datensätze stammen aus dem UCI Machine Learning Repository [FA12]. Davon wurden 5 Datensätze genommen, die unter dem Task *Clustering* oder *Classification* zu finden waren. Datensätze, die unter dem *Classification* Task zu finden sind, sind dabei ebenfalls für die Experimente geeignet, denn der Unterschied zu den Datensätzen des Task *Clustering* ist,

[3] https://www.openstack.org/

Datensatz	#Instanzen	#Attribute	Optimales k	Abkürzung
3D Road Network	434874	4	-	3DRN
Individual Household Electric Power Consumption	2075259	9	-	IHEPC
Shuttle (Statlog)	58000	9	7	Shuttle
Online Reatil	541909	8	5	OR
Skin Segmentation	245057	4	2	SKin

Tabelle 4.1.: Reale Datensätze mit der Anzahl der Instanzen, der Anzahl der Attribute und dem optimalem k (falls vorhanden)

dass zusätzlich noch für eine Teilmenge der Daten eine Klassifizierung vorliegt. Dadurch ist für diese Datensätze die Anzahl der Cluster bekannt und es kann beispielsweise im späteren Verlauf evaluiert werden, wie gut die Ansätze für das optimale k sind und wie gut für ein eher ungeeignetes k. Zusätzlich wurde darauf geachtet, dass es sich bei den Attributen nur um numerische Attribute handelt. Attribute die nicht optimal geeignet sind für die behandelten Clustering-Algorithmen, wie z. B. eine ID oder nicht numerische Attribute wurden entfernt. Die verwendeten Datensätze mit der zugehörigen Anzahl der Instanzen, der Attribute und dem optimalem k werden in Tabelle 4.1 dargestellt. Bei zwei dieser Datensätze ist das optimale k unbekannt. Dies wurde mit einem „-“ gekennzeichnet. Der Datensatz mit den meisten Instanzen ist der *Individual household electric power consumption* Datensatz. Der Datensatz mit den wenigsten Instanzen ist der *Shuttle* Datensatz.

4.3.2. Synthetische Datensätze

Für die synthetischen Datensätze wurden die unterschiedlichsten Datensätze ausgewählt. Es wurde darauf geachtet, dass Datensätze mit einer unterschiedlichen Anzahl an Clustern, mit spiral-förmigen Clustern und Datensätze, die für die ausgewählten Algorithmen geeignet und ungeeignet sind, vertreten sind. In Tabelle 4.2 werden die ausgewählten synthetischen Datensätze mit der Anzahl der Instanzen, der Attribute, dem optimalem k und der zugehörigen Referenz dargestellt.

Datensatz	#Instanzen	#Attribute	Optimales k	Referenz
A1	3000	2	20	[KF02]
A2	5250	2	35	[KF02]
A3	7500	2	50	[KF02]
Aggregation	788	2	7	[GMT07]
Chainlink	1000	3	2	[Ult05]
Cluto-t4	8000	2	6	[Kar02]
Cluto-t5	8000	2	6	[Kar02]
Cluto-t7	10000	2	9	[KHK99]
Cluto-t8	8000	2	8	[KHK99]
D31	3100	2	31	[VRB02]
Diamond9	3000	2	9	[SC04]
Engytime	4096	2	2	[Ult05]
R15	600	2	15	[VRB02]
S1	5000	2	15	[FV06]
S2	5000	2	15	[FV06]
S3	5000	2	15	[FV06]
S4	5000	2	15	[FV06]
Spiral	312	2	3	[CY08]
Unbalance	6500	2	8	[RF16]

Tabelle 4.2.: Synthetische Datensätze mit der Anzahl der Instanzen, der Anzahl der Attribute, dem optimalen k und der dazu gehörigen Referenz

5. Evaluation

In diesem Kapitel werden die Ansätze aus Abschnitt 3.3 anhand von mehreren Datensätzen getestet. In Abschnitt 5.1 werden die untersuchten Metriken evaluiert. Dazu wird geschaut welche der Metriken sich für ein frühzeitiges Terminieren vom K-means Algorithmus eignen und es wird ein entsprechendes Konvergenzkriterium festgelegt. Des Weiteren wird untersucht, welche Metriken sich dafür eignen, geeignete Zeitpunkte für die Visualisierung von Zwischenergebnissen zu finden. In Abschnitt 5.2 wird das definierte Konvergenzkriterium evaluiert. Dafür wird geschaut wie viel Zeit und Iterationen im Vergleich zum ursprünglichen Algorithmus eingespart werden können, wenn der Algorithmus terminieren würde, sobald das Konvergenzkriterium erreicht wurde, und wie hoch der Qualitätsverlust ist, wenn der Algorithmus konvergiert. Die Evaluation der Zeitpunkte für die Visualisierung wird in Abschnitt 5.3 vorgenommen. Anschließend wird in Abschnitt 5.4 evaluiert, wie viele Visualisierungen sich ergeben, wenn die Algorithmen durch das Konvergenzkriterium früher terminieren.

5.1. Metriken

In diesem Abschnitt werden die ausgewählten Metriken evaluiert. Dazu wird zum Einen untersucht, welche Metriken sich für ein Konvergenzkriterium eignen und welche sich dafür eignen, um geeignete Zeitpunkte zur Visualisierung von Zwischenergebnissen zu finden. Dazu werden die in Abschnitt 4.2 beschriebenen Experimente mit den in Abschnitt 4.3 vorgestellten Datensätzen ausgeführt.

Da bis auf die SSE und die $\#GP_i$ alle Metriken im Bereich zwischen 0 und 1 liegen, werden diese dahingehend abgeändert, dass für diese ebenfalls ein Wert zwischen 0 und 1 erreicht wird, um sie besser mit den anderen Metriken vergleichen zu können. Für die SSE wird die Änderung seit der letzten Iteration in Prozent betrachtet. Die Berechnung wird in Gleichung (5.1) dargestellt.

$$\%SSE_i = \frac{(SSE_{i-1} - SSE_i)}{SSE_{i-1}} \tag{5.1}$$

Dabei bezeichnet SSE_i die SSE am Ende der Iteration i. Da die SSE für den K-means und den K-means++ in jeder Iteration versucht wird zu minimieren, ist die SSE in der nächsten Iteration immer kleiner als in der vorherigen. Dadurch ist immer ein Wert zwischen 0 und 1 gewährleistet. Allerdings ist die $\%SSE_i$ dadurch nur für $i > 1$ definiert, da die SSE_0 undefiniert ist.

Für die Anzahl der geänderten Punkte wird die Änderung in Prozent seit der ersten Iteration betrachtet, da sich in der ersten Iteration alle Punkte ändern und es somit zumindest keine größeren Änderungen geben kann. Dadurch errechnet sich die Prozent der geänderten Punkte wie in Gleichung (5.2) beschrieben.

$$\%GP_i = \frac{(\#GP_0 - \#GP_i)}{\#GP_0} \tag{5.2}$$

Zunächst konnte festgestellt werden, dass die Berechnung der Metriken durch die in Abschnitt 3.3 beschrieben Optimierungsansätze sehr gering ist. In Abbildung 5.1 wird der Berechnungsaufwand der Metriken im Verhältnis zur Ausführungszeit des K-means Algorithmus für den Realdatensatz *Individual Household Electric Power Consumption* mit $k = 50$ dargestellt. Dazu wurden die Berechnungszeiten für jede Iteration der jeweiligen Metrik aufsummiert

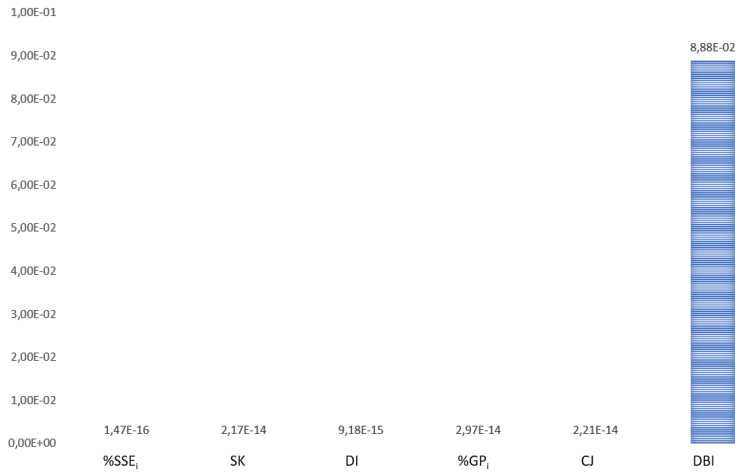
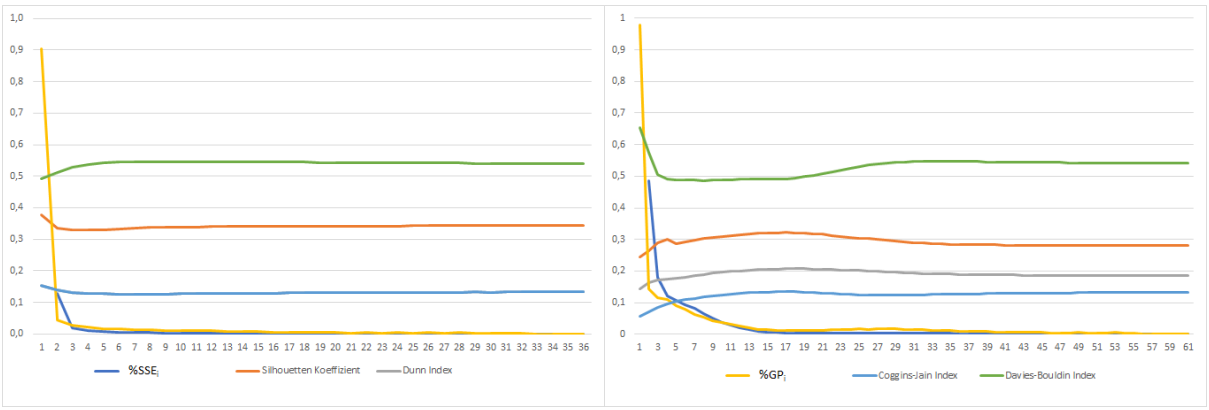


Abbildung 5.1.: Berechnungszeit der Metriken im Verhältnis zur Ausführungszeit des K-means Algorithmus in Prozent für den *Individual Household Electric Power Consumption* Datensatz mit $k = 50$

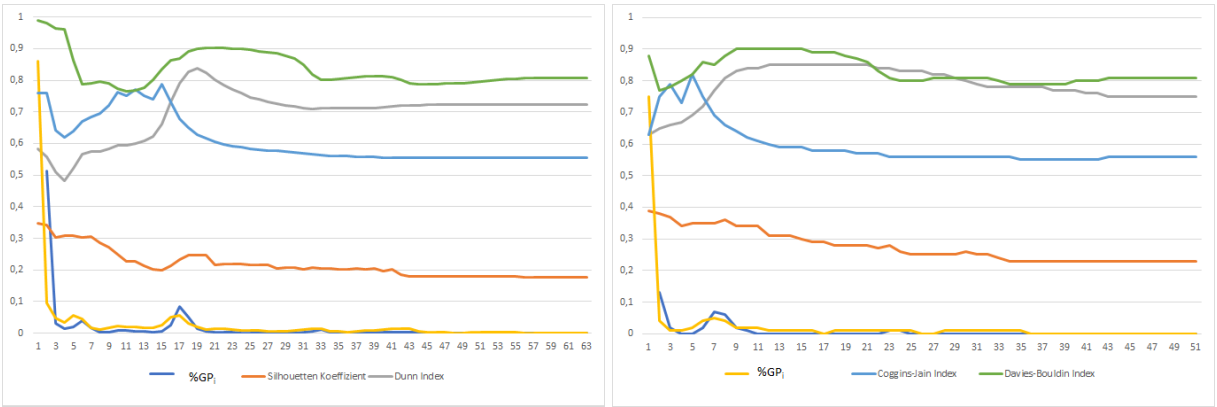
und durch die Ausführungszeit des Algorithmus geteilt. Zusätzlich wurde dies mit dem Faktor 100 multipliziert um so eine Prozentangabe zu erhalten. Für alle Metriken ist zu sehen, dass die Berechnungszeit deutlich unter 1% im Vergleich zu der Ausführungszeit des K-means liegt. Den größten Berechnungsaufwand hat dabei der Davies-Bouldin Index mit $0,0888\%$. Die anderen Metriken weisen sogar einen deutlich geringeren Berechnungsaufwand. Als Nächstes werden die Verläufe der Metriken während der Ausfürhung der Algorithmen betrachtet um so geeignete Metriken heraus zu finden. In Abbildung 5.2 wird der Verlauf der Metriken für den K-means (a) und den K-means++ (b) Durchlauf dargestellt. Als Datensatz wurde der Realdatensatz *3D Road Network* mit $k = 5$ verwendet. Auf der x-Achse ist die jeweilige Iteration und auf der y-Achse der Wert der jeweiligen Metrik zu sehen. Für diesen Datensatz ist zu erkennen, dass der Verlauf der Metriken kaum Schwankungen aufweist und die Metriken sich relativ schnell einem bestimmten Grenzwert annähern. Für den K-means sind beim Davies-Bouldin Index



(a) K-means **(b)** K-means++

Abbildung 5.2.: Verlauf der Metriken während der Ausführung des K-means und des K-means++ Algorithmus auf dem Datensatz *3D Road Network* mit $k = 5$

leichte Schwankungen im Verlauf zu erkennen. Dies ist jedoch beim K-means++ nicht zu sehen. Solch ein Verlauf ohne große Schwankungen wäre für ein Konvergenzkriterium wünschenswert, da relativ schnell abgeschätzt werden könnte, gegen welchen Wert die Metrik strebt, wenn der Algorithmus terminiert. Dies ist besonders bei den Realdatensätzen aufgetreten. Allerdings war insbesondere für die synthetische Datensätze ein anderer Verlauf der Metriken zu beobachten, wie Abbildung 5.3 zeigt. In dieser Abbildung ist der Verlauf der Metriken für



(a) K-means **(b)** K-means++

Abbildung 5.3.: Verlauf der Metriken während der Ausführung des K-means und des K-means++ Algorithmus auf dem Datensatz A1 mit $k = 5$

die Algorithmen auf dem Datensatz A1 mit $k = 5$ zu sehen. Dabei ist zu erkennen, dass der Verlauf der Metriken, im Vergleich zu Abbildung 5.2, größere Schwankungen aufweist. Zu Beginn der Algorithmen sind diese Schwankungen am deutlichsten zu erkennen. Dabei sind sowohl Schwankungen nach oben als auch nach unten zu erkennen, was die Einschätzung

deutlich erschwert, da nicht einmal ersichtlich ist, in welche Richtung sich diese Metriken bewegen. Dies ist insbesondere beim Davis-Bouldin Index, beim Coggins-Jain Index und beim Dunn Index der Fall. Der Silhouetten-Koeffizient weist kleinere Schwankungen auf als die vorher genannten Metriken. Für den K-means++ hat der Silhouetten-Koeffizient sogar einen monoton fallenden Verlauf. Allerdings ist dies nicht der gewünschte Verlauf der Metrik, da dies bedeuten würde, dass die Qualität der Cluster im Laufe des Algorithmus schlechter wird. Hinzu kommt, dass es auch Durchläufe gab, bei denen der Silhouetten-Koeffizient angestiegen ist. Ein Grund hierfür könnte der in Abschnitt 3.3 beschriebene Optimierungsansatz OA2 sein, welcher die Kompaktheit durch den Abstand des Punktes mit der größten Distanz zum Zentroiden beschreibt. Denn die erwähnten Metriken sind alle von diesem Ansatz betroffen. Allerdings ist eine Berechnung der Metriken ohne diesen Optimierungsansatz nicht erstrebenswert, da die Berechnung der Metriken sonst in mindestens $O(n)$ liegt, womit die Berechnung der Metriken in einer Iteration fast so hoch ist, wie die Berechnung für einen kompletten Durchlauf des Algorithmus. Für die $\%SSE_i$ und die $\%GP_i$ ist zu erkennen, dass diese am Anfang relativ stark abfallen und sich dann im Bereich um die 0 herum bewegen. Dabei sind zwar auch Ausreißer nach oben zu erkennen, allerdings sind diese nicht so gravierend. Zudem waren die Verläufe für die $\%SSE_i$ und die $\%GP_i$ für alle Datensätze ähnlich wie in Abbildung 5.2 und Abbildung 5.3 dargestellt.

Aus diesem Grund, weil sich der Verlauf am ehesten abschätzen lässt, haben sich diese beiden Metriken als am Geeignetsten für ein Konvergenzkriterium herausgestellt. Würde die $\%GP_i$ alleine als Konvergenzkriterium herangezogen werden, dann würde nur eine Aussage über die Änderung der Punkte getroffen werden, nicht aber über die Qualität. Denn es ist möglich, dass zwar wenige Punkte ihre Clusterzugehörigkeit ändern, dies jedoch einen großen Einfluss auf die Qualität hat. In Abbildung 5.4 wird dies verdeutlicht. Dabei ist der Wert der $\%SSE_i$ und der $\%GP_i$ in Prozent gegeben. Die $\%GP_i$ fällt in der Abbildung zwar monoton, die $\%SSE_i$ jedoch weist besonders zu Beginn noch größere Schwankungen auf. Auf der anderen Seite wiederum ist ein Verlauf wie in Abbildung 5.5 möglich. Dabei fällt die $\%SSE_i$ stark zu Beginn sehr stark und die $\%GP_i$ schwankt sehr stark. Zudem ist die $\%SSE_i$ bereits nach Iteration 6 deutlich unter 1%. Allerdings steigt sie danach auch wieder (Iteration 9 - 13). Die $\%GP_i$ hingegen weist zu diesem Zeitpunkt noch starke Schwankungen auf und fällt erst nach der Iteration 12 unter 10%. Würde nur die $\%SSE_i$ als Konvergenzkriterium gewählt werden, würde der Algorithmus bereits vor dem Wiederanstieg der $\%SSE_i$ terminieren, wodurch eine größere Qualitätsänderung verloren geht. Der erstgenannte Fall tritt zwar öfters auf, sollte jedoch der andere Fall auftreten, wäre auch dies durch eine Kombination der beiden Metriken abgedeckt. Als Konvergenzkriterium wurde somit eine Kombination aus diesen beiden Metriken festgelegt. Sobald diese Metriken einen bestimmten Schwellenwert unterschreiten, sollen die Algorithmen terminieren. Dies bedeutet, dass die Algorithmen terminieren, wenn die erste Iteration i erreicht ist, für die die Gleichung (5.3) erfüllt ist.

$$(\%SSE_i \leq \varepsilon_1) \wedge (\%GP_i \leq \varepsilon_2) \tag{5.3}$$

Für die Bestimmung von geeigneten Zeitpunkten für die Visualisierung von Zwischenergebnissen sind diese Metriken ebenfalls gut geeignet, da sie auch mit angeben wie viel sich

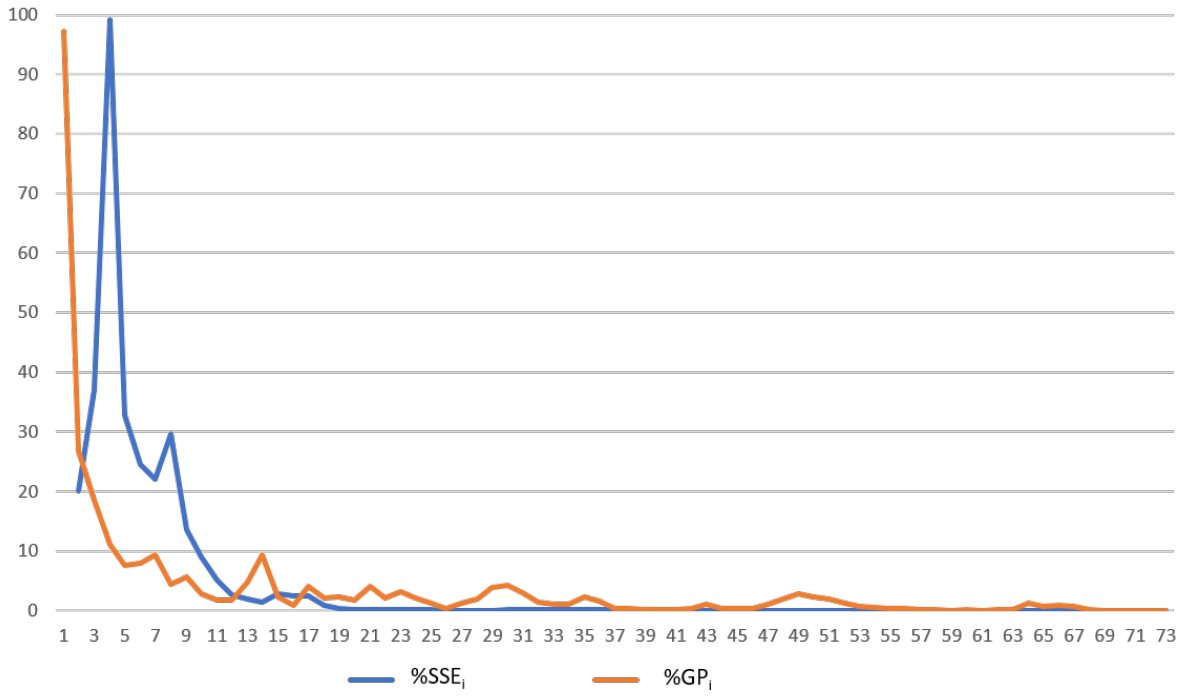


Abbildung 5.4.: Verlauf der $\%SSE_i$ und der $\%GP_i$ in Prozent während der Ausführung des K-means Algorithmus auf dem Datensatz *Shuttle* mit $k = 50$

geändert hat. Ähnlich wie beim Konvergenzkriterium wird ein Schwellenwert gesetzt und sobald dieser überschritten wird, stellt dies ein Zeitpunkt für eine Visualisierung dar. Anders als beim Konvergenzkriterium werden beide Metriken separat behandelt und es wird nicht geschaut ob in jeder Iteration der Schwellenwert überschritten wird, sondern es wird die Summe der jeweiligen Metrik berechnet, bis dieser Schwellenwert überschritten wird. Wird dieser Schwellenwert überschritten, dann wird die Summe auf 0 gesetzt und es wird wieder auf die gleiche Weise geschaut wann die Summe der jeweiligen Metrik den Schwellenwert überschreitet. Zur Verdeutlichung wird dies in Algorithmus 5.1 für die $\%GP_i$ in Pseudocode dargestellt. Dabei ist α der Schwellenwert der erreicht werden soll und sum die beschriebene Summe. Für die $\%SSE_i$ ist die Vorgehensweise analog dazu.

Die Schwellenwerte ε_1, ε_2 und α werden im späteren Verlauf experimentell bestimmt und evaluiert.

5.2. Konvergenzkriterien

In diesem Abschnitt soll evaluiert werden, wie gut das Konvergenzkriterium (vgl. (5.3)) geeignet ist. Dazu werden die Ergebnisse, die anhand der Datensätze gemacht wurden, bewertet. Zuvor

Algorithmus 5.1 Pseudocode für die Bestimmung von geeigneten Zeitpunkten für eine Visualisierung für einen Schwellenwert α

```
1:  sum ← 0
2:  numberOfVisualisations ← 0
3:  while !algorithm.isFinished do
4:      ...                                  // Führe nächste Iteration des Algorithmus aus
5:      sum ← sum + algorithm.getPercentOfChangedPoints()
6:      if sum ≥ α then
7:          visualize()
8:          sum ← 0
9:          numberOfVisualisations ← numberOfVisualisations + 1
10:     end if
11:     ...
12: end while
```

werden allerdings erst die Schwellenwerte für das Konvergenzkriterium in Abschnitt 5.2.1 bestimmt. Anschließend wird das Konvergenzkriterium anhand der Ergebnisse in Abschnitt 5.2.2 evaluiert. In Abschnitt 5.2.3 werden kritische Fälle, die sich während der Bewertung ergeben haben, genauer untersucht und es wird nach den Ursachen geforscht.

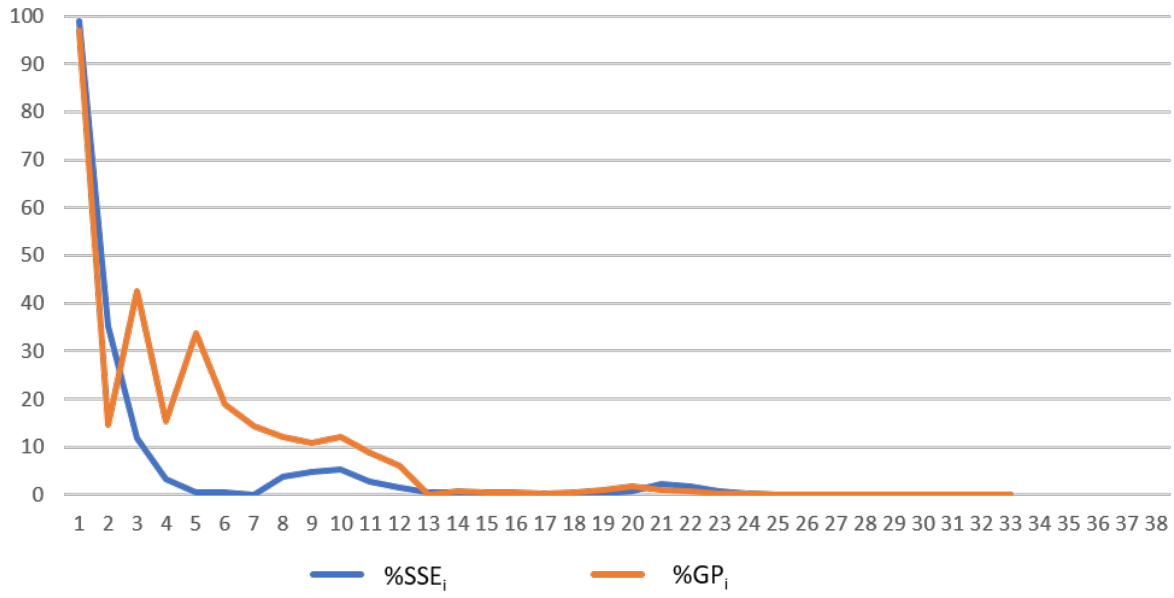


Abbildung 5.5.: Verlauf der $\%SSE_i$ und der $\%GP_i$ in Prozent während der Ausführung des K-means Algorithmus auf dem Datensatz *S4* mit $k = 50$

5.2.1. Bestimmung der Schwellenwerte

Für die Evaluierung des Konvergenzkriteriums (vgl. Gleichung (5.3)) müssen zunächst die Parameter ε_1 und ε_2 bestimmt werden. Dazu wurde der Verlauf der Metriken $\%GP_i$ und $\%SSE_i$ eingehender analysiert und geschaut unter welche Werte die Metriken fallen müssen, damit kein Anstieg im Verlauf mehr zu sehen ist. Wie die Abbildungen 5.4 und 5.5 zeigen, hat sich für die $\%GP_i$ dabei ein Wert von 10% als gut geeignet herausgestellt, womit $\varepsilon_1 = 0,1$ gesetzt wurde. Dies wird aber noch einmal in Abbildung 5.6 verdeutlicht. In dieser Abbildung ist die $\%SSE_i$ zu

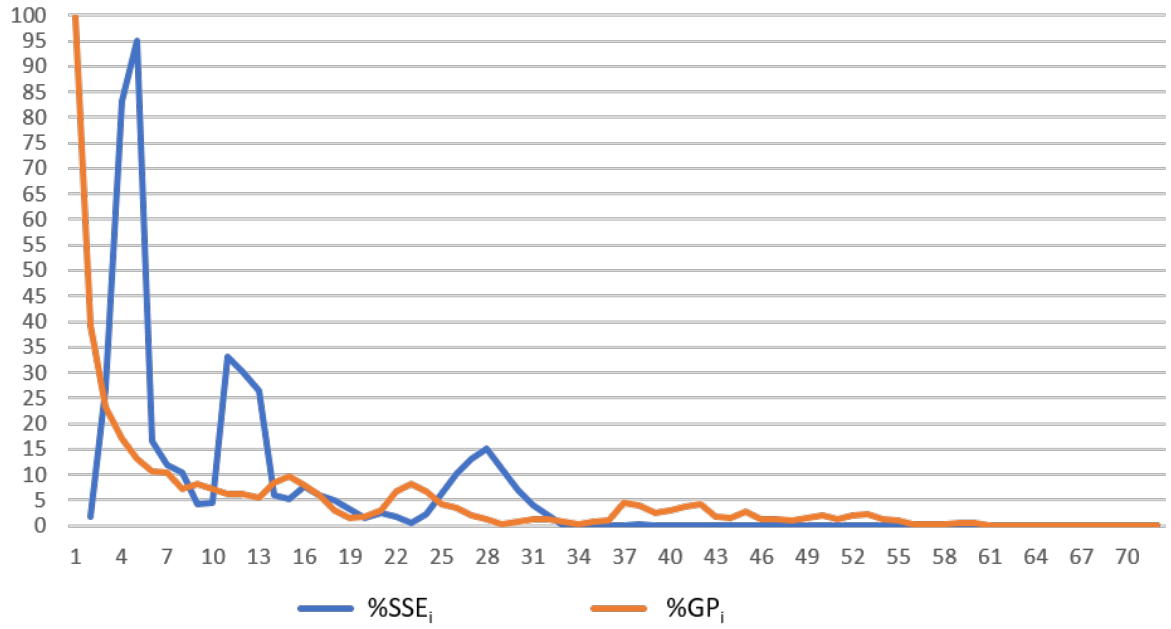


Abbildung 5.6.: Verlauf der $\%SSE_i$ und der $\%GP_i$ in Prozent während der Ausführung des K-means++ Algorithmus auf dem Datensatz *Online Retail* mit $k = 50$

Beginn deutlich unter 5%. Ab Iteration 7 jedoch fängt sie wieder an zu steigen und weist in der Folge größere Schwankungen auf. In der Iteration 25 steigt sie sogar auf einen Wert über 50% an. Die $\%GP_i$ hat zwar auch einige Schwankungen im Verlauf bis Iteration 11, jedoch fällt sie auch erst nach Iteration 11 unter den 10%-Wert. In den folgenden Iterationen hat die $\%GP_i$ einen monotonen fallenden Verlauf, allerdings müsste für die Konvergenz dann die $\%SSE_i$ wieder unter ihren Schwellenwert fallen. Dadurch kann ein größerer Qualitätsverlust verhindert werden, da die Terminierung des Algorithmus erst erfolgt, wenn sich die $\%SSE_i$ wieder eingependelt hat. In der Abbildung 5.6 wir allerdings auch deutlich, dass der Schwellenwert von $\%SSE_i$ deutlich unter 5% liegen sollte. Denn die $\%SSE_i$ fällt in Iteration 19 unter 5%, steigt danach aber trotzdem auch nochmal an, während die $\%GP_i$ unter ihrem Schwellenwert bleibt. Um auch so einen Fall abzudecken wird daher der Schwellenwert von $\%SSE_i$ auf 1% gesetzt. Also wird $\varepsilon_2 = 0,01$ gesetzt. Daher lautet das festgelegte Konvergenzkriterium

mit festgelegten Schwellenwerten wie in (C, 5.4) beschrieben und wird im Folgenden mit C bezeichnet.

$$(\%SSE_i \leq 0.01) \wedge (\%GP_i \leq 0.1) \tag{C, 5.4}$$

5.2.2. Bewertung anhand der Ergebnisse

Im Folgenden soll evaluiert werden wie gut dieses Kriterium geeignet ist, indem die Qualität der Terminierung des Algorithmus mit der Qualität der früheren Terminierung des Algorithmus aufgrund des Konvergenzkriteriums verglichen wird. Dazu wird der Algorithmus ausgeführt und wenn das Konvergenzkriterium C erreicht ist, wird die Qualität, die Anzahl der Iterationen und die Ausführungszeit des Algorithmus bis zu diesem Zeitpunkt festgehalten. Wenn der Algorithmus terminiert, weil sich die Zentroide nicht mehr verschieben wird dafür ebenfalls die Qualität, die Anzahl der Iterationen und die Ausführungszeit festgehalten. Für die Qualität wird die SSE herangezogen, da diese das Konvergenzkriterium des Standard K-means ist und gegen ein lokales Minimum konvergiert. Somit ist die SSE zum Zeitpunkt der Terminierung der bestmögliche Wert der bei einer früheren Terminierung erreicht werden kann. Für die Beschreibung des Qualitätsverlusts wird der relative Qualitätsverlust herangezogen. Die Formel zur Berechnung des relativen Qualitätsverlustes ist in (5.5) beschrieben.

$$\%E = \frac{SSE_C - SSE_S}{SSE_C} \tag{5.5}$$

Dabei bezeichnet SSE_C die SSE zum Zeitpunkt der Terminierung durch das Konvergenzkriterium C und SSE_S bezeichnet die SSE am Ende der Terminierung durch das Konvergenzkriterium des Standard K-means, welches im Folgenden mit S bezeichnet wird. Die Versuche wurden wie in Abschnitt 4.2 beschrieben durchgeführt. Es wurde zudem geschaut wo die kleinste, größte und der Median der Iterationen liegt bei denen der Algorithmus durch das Konvergenzkriterium C terminiert. Falls nicht anders angegeben werden im Folgenden Aussagen über den Median getroffen, da der Median im Gegensatz zum Mittelwert beispielsweise robuster gegen Ausreißer ist [Dix53].

Tabelle 5.1.: Einsparung von Iteration und Ausführungszeit, so wie Qualitätsverlust und SSE_C für K-means und K-means++ nach der Ausführung auf den Realdatensätzen für die Median Durchläufe der Konvergenz

		K-means				K-means++			
Datensatz	k	%I	%A	%E	SSE_C	%I	%A	%E	SSE_C
IHEPC	5	81,82%	67,41%	0,15%	6,00E+07	89,47%	64,93%	52,02%	6,46E+07
IHEPC	10	95,08%	85,46%	0,80%	2,29E+07	87,80%	70,66%	0,02%	2,25E+07
IHEPC	15	96,59%	89,58%	69,19%	5,12E+07	90,91%	76,76%	1,17%	1,32E+07
IHEPC	20	93,14%	86,95%	1,13%	1,79E+07	94,64%	83,73%	6,62%	1,53E+07
IHEPC	25	94,41%	89,50%	0,83%	1,38E+07	96,59%	86,98%	3,13%	8,00E+06
IHEPC	30	96,00%	91,81%	2,93%	1,59E+07	95,39%	87,24%	3,18%	7,87E+06
IHEPC	35	95,97%	92,05%	10,04%	1,20E+07	93,66%	86,49%	24,56%	7,95E+06
IHEPC	40	95,71%	92,37%	6,39%	1,11E+07	96,09%	89,75%	6,37%	5,91E+06
IHEPC	45	97,77%	94,73%	20,23%	1,29E+07	97,97%	91,79%	13,88%	6,21E+06
IHEPC	50	96,58%	93,62%	10,09%	8,48E+06	95,18%	89,45%	1,99%	4,68E+06
OR	5	51,72%	36,78%	1,36%	3,45E+09	59,09%	58,51%	0,55%	3,40E+09
OR	10	34,38%	24,63%	0,45%	2,01E+09	52,78%	53,54%	3,00%	1,95E+09
OR	15	56,86%	50,84%	5,66%	1,44E+09	66,67%	65,25%	12,14%	1,68E+09
OR	20	52,27%	47,29%	3,94%	1,37E+09	77,50%	77,50%	6,56%	1,34E+09
OR	25	69,23%	63,24%	4,36%	1,32E+09	80,85%	84,75%	3,15%	1,29E+09
OR	30	67,74%	62,80%	6,09%	1,32E+09	84,62%	85,07%	35,01%	1,20E+09
OR	35	81,31%	76,97%	5,08%	1,32E+09	80,49%	80,95%	33,40%	1,13E+09
OR	40	80,65%	76,63%	7,95%	1,34E+09	87,27%	88,42%	2,04%	1,13E+09
OR	45	77,27%	73,37%	4,25%	1,29E+09	88,24%	88,30%	6,88%	2,42E+08
OR	50	75,34%	71,26%	2,14%	1,26E+09	72,73%	73,34%	3,09%	1,32E+08
Shuttle	5	95,90%	85,02%	0,21%	1,51E+08	71,43%	55,50%	0,21%	1,24E+08
Shuttle	10	96,64%	88,71%	22,88%	7,32E+07	96,61%	86,14%	0,11%	5,31E+07
Shuttle	15	93,22%	85,25%	31,44%	8,89E+07	96,25%	87,35%	10,74%	1,64E+07
Shuttle	20	94,35%	89,02%	11,41%	6,76E+07	98,53%	91,43%	22,23%	1,08E+07
Shuttle	25	93,85%	89,03%	0,68%	9,94E+07	88,14%	82,49%	42,13%	9,50E+06
Shuttle	30	92,24%	88,21%	8,81%	1,16E+08	88,14%	80,45%	1,55%	5,71E+06
Shuttle	35	91,51%	88,09%	0,31%	7,18E+07	93,98%	88,09%	4,13%	2,81E+06
Shuttle	40	89,90%	86,47%	0,22%	8,35E+07	87,50%	83,01%	44,91%	4,36E+06
Shuttle	45	91,67%	87,03%	0,32%	8,44E+07	86,67%	81,73%	3,02%	2,20E+06
Shuttle	50	91,35%	88,15%	1,71%	8,39E+07	82,05%	78,20%	4,14%	2,20E+06

		K-means				K-means++			
Datensatz	k	%I	%A	%E	SSE_C	%I	%A	%E	SSE_C
Skin	5	57,14%	32,84%	0,01%	5,10E+08	72,22%	54,62%	0,15%	5,03E+08
Skin	10	81,08%	72,67%	4,07%	3,58E+08	63,16%	47,25%	0,07%	2,63E+08
Skin	15	71,43%	64,51%	9,17%	2,03E+08	81,40%	70,10%	11,52%	2,39E+08
Skin	20	76,92%	70,24%	7,49%	1,63E+08	75,76%	66,75%	0,17%	1,45E+08
Skin	25	69,70%	64,43%	2,05%	1,23E+08	76,67%	72,53%	1,56%	1,22E+08
Skin	30	86,96%	83,89%	4,28%	1,12E+08	80,00%	73,89%	4,26%	1,00E+08
Skin	35	72,22%	67,63%	7,07%	8,64E+07	79,49%	74,65%	6,29%	8,50E+07
Skin	40	88,64%	85,73%	11,65%	8,33E+07	77,14%	72,88%	5,51%	7,93E+07
Skin	45	80,33%	78,47%	11,19%	8,27E+07	92,45%	88,06%	7,00%	6,96E+07
Skin	50	82,54%	79,28%	10,71%	6,91E+07	80,00%	77,05%	6,99%	6,06E+07
3DRN	5	78,26%	60,33%	4,43%	1,41E+33	86,36%	70,00%	5,77%	1,43E+33
3DRN	10	71,88%	60,96%	8,90%	3,63E+32	86,15%	50,00%	12,05%	3,76E+32
3DRN	15	77,23%	75,00%	14,04%	2,04E+32	80,85%	50,00%	3,57%	1,82E+32
3DRN	20	88,02%	80,51%	34,22%	1,83E+32	90,32%	66,67%	5,62%	1,25E+32
3DRN	25	90,12%	70,07%	54,91%	1,47E+32	79,49%	50,00%	5,04%	9,35E+31
3DRN	30	92,27%	87,12%	40,89%	9,35E+31	78,13%	50,00%	1,98%	4,25E+31
3DRN	35	83,82%	78,68%	11,76%	4,68E+31	88,33%	80,00%	55,03%	7,42E+31
3DRN	40	91,88%	87,40%	43,51%	6,31E+31	79,55%	75,00%	2,47%	2,64E+31
3DRN	45	91,93%	87,69%	66,23%	1,11E+32	75,00%	66,67%	2,97%	2,41E+31
3DRN	50	88,12%	84,00%	22,11%	3,78E+31	87,10%	75,00%	3,07%	1,94E+31
Min		34,38%	24,63%	0,01%	8,48E+06	52,78%	47,25%	0,02%	2,20E+06
Max		97,77%	94,73%	69,19%	1,41E+33	98,53%	91,79%	55,03%	1,43E+33
Median		88,07%	82,20%	6,24%	1,37E+08	86,26%	76,91%	4,20%	1,23E+08
Mittelwert		82,82%	76,27%	12,19%	5,32E+31	83,74%	74,58%	9,86%	4,79E+31

Realdatensätze

In Tabelle 5.1 werden die Ergebnisse für die Durchläufe, wo die Algorithmen K-means und K-means++ im Median durch das Konvergenzkriterium C terminiert haben, für die Realdatensätze dargestellt. Das bedeutet, dass von 10 Durchläufen untersucht wurde, bei welchem dieser Durchläufe der Median der Iteration liegt, in der konvergiert wurde. Dabei stellt %I die Iterationen in Prozent dar, die durch C im Vergleich zu S eingespart wurden. Diese lässt sich mit der Gleichung

$$\%I = \frac{I_C - I_S}{I_S} \tag{5.6}$$

berechnen, wobei I_C die Anzahl der Iterationen für die Terminierung mittels des Konvergenzkriteriums C beschreibt und I_S entsprechend für das Konvergenzkriterium S. Die Zeit, die eingespart wurde, wird in Prozent in den Spalten %A angegeben. Die Berechnung verläuft analog zu $\%I$. Zur besseren Übersicht wurden zudem Zellen grün markiert, falls $\%I \geq 80\%$ $\%A \geq 80\%$ und rot , falls $\%E \geq 5\%$ war. Des weiteren wurde für jede Spalte separat der Min-, Max-, Median- und Mittelwert erfasst. Zunächst ist in der Tabelle zu sehen, dass durch das Konvergenzkriterium relativ viel Zeit und Iterationen eingespart werden können. Insbesondere für den Datensatz mit den meisten Instanzen, dem *IHEPC* Datensatz. Dies ist sowohl für den K-means als auch den K-means++ der Fall. So wurden immer mehr als 80% an Iterationen eingespart und immer mindestens 64% der Ausführungszeit. Insgesamt waren die Qualitätsverluste Tabelle 5.1 beim K-means 22-mal unter 5% und 32-mal unter 10%. Für den K-means waren die Qualitätsverluste sogar 26-mal unter 5% und 37-mal unter 10%. In der Einsparung der Ausführungszeit und der Iterationen sind sich zudem der K-means und der K-means++ sehr ähnlich. Dies wird in Abbildung 5.7 verdeutlicht. In dieser ist die Ausführungszeit für alle realen Datensätze mit allen k's für den K-means und den K-means++ dargestellt. Für den *IHCEP* Datensatz wird dabei festgestellt, dass der K-means in der Einsparung der Ausführungszeit für alle k's oberhalb des K-means++ ist.

Im Qualitätsverlust gibt es allerdings deutliche Unterschiede bei den Algorithmen. So ist beim *IHEPC* Datensatz für $k = 5$ der Qualitätsverlust für K-means mit 0,15% sehr gering und beim K-means++ mit 52,02% sehr hoch, was zunächst überraschend wirkt, da für den K-means++ auf Grund seiner besseren Initialisierungsstrategie auch bessere Ergebnisse erwartet wurden. Wird für diesen Fall allerdings die absolute Qualität betrachtet, kann festgestellt werden, dass die Qualität, die mit dem K-means++ erreicht wurde deutlich besser als die des K-means ist. Denn für den K-means wäre die $SSE_S = 59905497,3$ und für den K-means++ wäre die $SSE_S = 31015774,91$[1]. Hinzu kommt, dass es für die realen Datensätze nur 5-mal der Fall war, dass die SSE_C des K-means kleiner war als die des K-means++. Die 5-mal wo dies auftrat wurden in der Tabelle 5.1 mit blau gekennzeichnet.

In Abbildung 5.8 sind besonders gute Ergebnisse beim K-means für den *Shuttle* Datensatz für $k = 5, 25, 35, 40, 45, 50$ zu erkennen. Dabei lag der Qualitätsverlust außer für $k = 50$ unter 1%. Für $k = 50$ lag der Qualitätsverlust bei 1,71%. In Abbildung 5.7 ist zudem zusehen, dass für diese Fälle mindestens 85% der Ausführungszeit eingespart werden. Würde man nur nach dem Qualitätsverlust gehen, müsste der *OR* Datensatz hervorgehoben werden, d K-means für diesen Datensatz keine großen Schwankungen aufweist und zudem immer knapp unter oder über dem 5%-Wert liegt. Betrachtet man allerdings für diesen Datensatz die eingesparte Ausführungszeit ist zu sehen, dass diese immer unter 80% liegt (vgl. Abbildung 5.7). Für den K-means++ ist der *Skin* Datensatz hervorzuheben. Für $k = 5, 10, 30, 35, 45, 50$ war der Qualitätsverlust unter 5% und für $k = 10, 30, 35, 45$ war zudem die Einsparung über 80%. Dafür waren für $k = 25$ und $k = 40$ die Qualitätsverluste über 40%. Wird zudem die Abbildung 5.8 betrachtet, ist zu erkennen, dass der Parameter k einen großen Einfluss auf den Qualitätsverlust hat. Sowohl

[1]Die SSE_S berechnet sich durch Umformung der Gleichung (5.5) mit $SSE_S = SSE_C - \%E * SSES_C$

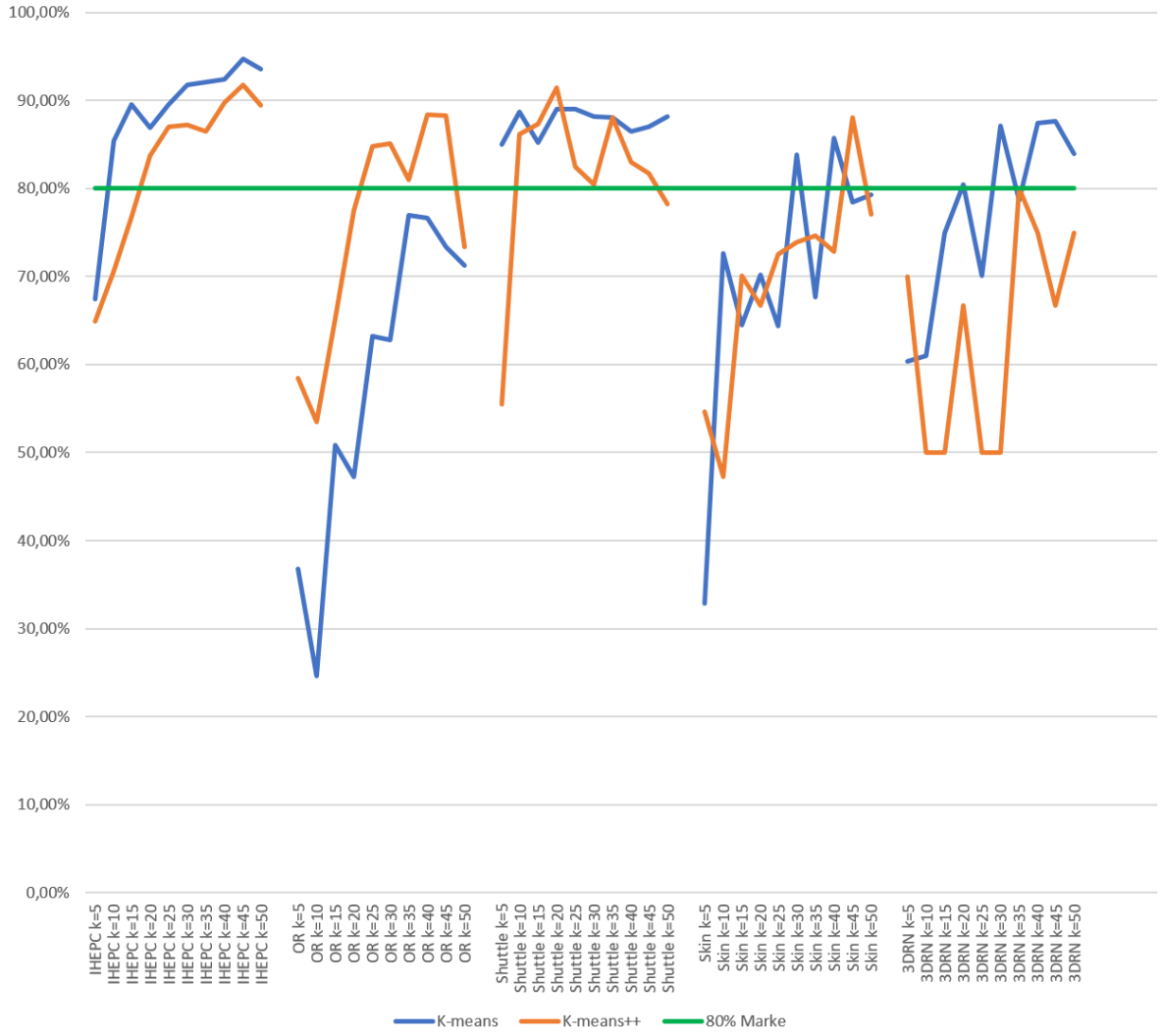


Abbildung 5.7.: Einsparung der Ausführungszeit in Prozent für den K-means und K-means++ über alle reale Datensätze und die verschiedenen k's

für den K-means als auch den K-means++ sind immer wieder größere Schwankungen für den Qualitätsverlust zu erkennen.

In Tabelle 5.2 wird der Qualitätsverlust und die eingesparte Iteration in Abhängigkeit von der eingesparten Ausführungszeit für den K-means dargestellt. Dabei wurde untersucht wie sich bei einer Einsparung die größer oder gleich 50%, 70%,80% und 90% der Qualitätsverlust und die eingesparten Iterationen verhalten. Zudem wird die Anzahl der Durchläufe aufgefasst für die die Einsparung der Ausführungszeit unter 50%, …, 90% war und ins Verhältnis zu allen gesetzt. In dieser Tabelle ist zu sehen, dass der der Qualitätsverlust im Median steigt, wenn man einen Bereich mit einer höheren Einsparung der Ausführungszeit betrachtet. So beträgt

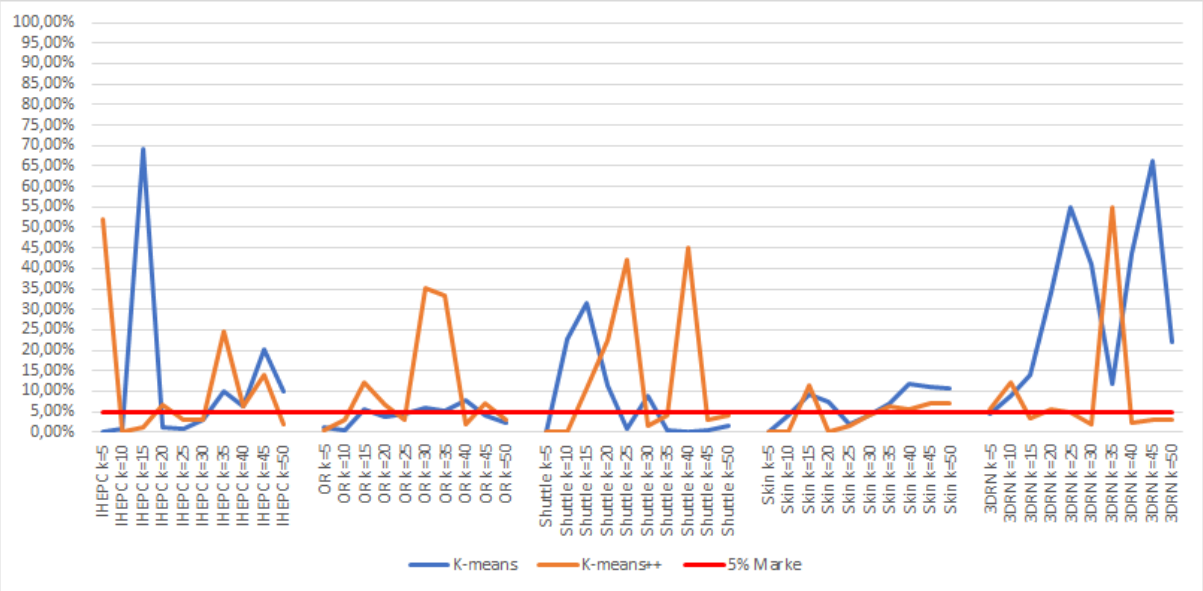


Abbildung 5.8.: Qualitätsverlust in Prozent für den K-means und K-means++ über alle reale Datensätze und die verschiedenen k's

	%A größer als	50%	70%	80%	90%
	Anzahl	46/50	37/50	26/50	5/50
Min	%E	0,15%	0,21%	0,21%	2,93%
	%I	56,86%	75,34%	86,96%	95,71%
Median	%E	7,28%	8,81%	9,43%	10,04%
	%I	89,27%	91,67%	93,18%	96,00%
Max	%E	69,19%	69,19%	69,19%	20,23%
	%I	97,77%	97,77%	97,77%	97,77%

Tabelle 5.2.: Qualitätsverlust und Einsparung an Iterationen in Abhängigkeit von der eingesparten Ausführungszeit für den K-means auf den realen Datensätzen

der Qualitätsverlust für eine Einsparung der Ausführungszeit von 50% im Median 7,28% und für eine Einsparung der Ausführungszeit von 90% beträgt der der Qualitätsverlust im Median 10,04%. Das bedeutet, dass der Qualitätsverlust steigt, wenn es eine höhere Einsparung der Ausführungszeit gab. In Tabelle 5.3 wird die selbe Tabelle für den K-means++ dargestellt. Für den K-means++ ist dabei ebenfalls zu sehen, dass im Median für höhere Einsparungen der Ausführungszeit auch höhere Qualitätsverluste zu erkennen sind. Herauszuheben ist dabei, dass der Qualitätsverlust beim K-means++ im Median immer geringer ist als beim K-means, außer für eine Einsparung der Ausführungszeit von 90% oder mehr. Für diese liegt der Qualitätsverlust im Median bei 18,06% für den K-means++ und bei 10,04% für den K-means.

	%A größer als	50%	70%	80%	90%
Min	Anzahl	45/50	35/50	21/50	2/50
	%E	0,02%	0,02%	0,11%	13,88%
	%I	52,78%	72,73%	80,49%	97,97%
Median	%E	4,26%	5,51%	6,62%	18,06%
	%I	87,10%	87,80%	93,66%	98,25%
Max	%E	55,03%	55,03%	44,91%	22,23%
	%I	98,53%	98,53%	98,53%	98,53%

Tabelle 5.3.: Qualitätsverlust und Einsparung an Iterationen in Abhängigkeit von der eingesparten Ausführungszeit für den K-means++ auf den realen Datensätzen

Synthetische Datensätze

In Tabelle A.1 auf Seite 88 werden alle Ergebnisse der Durchläufe, in denen die Algorithmen K-means und K-means++ im Median durch C terminiert haben, für die synthetischen Datensätze dargestellt.

Allgemein sind für die synthetischen Datensätze bessere Ergebnisse zu erkennen als für die realen Datensätze, was den Qualitätsverlust anbelangt. So lag der Qualitätsverlust bei den synthetischen Datensätzen für den K-means im Median bei 1,79% und für den K-means++ bei 1,19%. Bei den realen Datensätzen sind es für den K-means hingegen 6,24% und für den K-means++ 4,2%. In der eingesparten Ausführungszeit sind die realen Datensätze dafür im Median besser. Für den K-means sind das 82,20% und für den K-means++ 76,91%. Bei den synthetischen Datensätzen sind es für den K-means 79,45% und für den K-means++ 75%.

Zudem ist zu sehen, dass bei den synthetischen Datensätzen auch gute Ergebnisse erzielt werden konnten für Datensätze bei denen der K-means eher ungeeignet ist. So war für den *Spiral* Datensatz der Qualitätsverlust für den K-means++ für $k = 5$ bei 7,97%, für die restlichen k's allerdings immer deutlich unter 1%. Für den K-means lag der Qualitätsverlust, außer für $k = 10$, für alle k's unter 1%. Für $k = 10$ lag er bei 1,18%. Die Einsparung der Ausführungszeit lag zudem für beide Algorithmen für alle k's über 50% und es konnten sogar bis zu 92,31% für den K-means und bis zu 93,14% für den K-means++ eingespart werden. Das ist dadurch zu begründen, dass sich die SSE für Datensätze, die für den K-means ungeeignet sind, kaum Änderungen in der SSE ergeben. In Abbildung 5.9 wird dies für den *Spiral* Datensatz mit $k = 50$ für den K-means++ Algorithmus verdeutlicht. Sowohl die $\%SSE_i$ als auch $\%GP_i$ gehen relativ schnell gegen 0 und zeigen im Verlauf auch keinerlei Schwankungen.

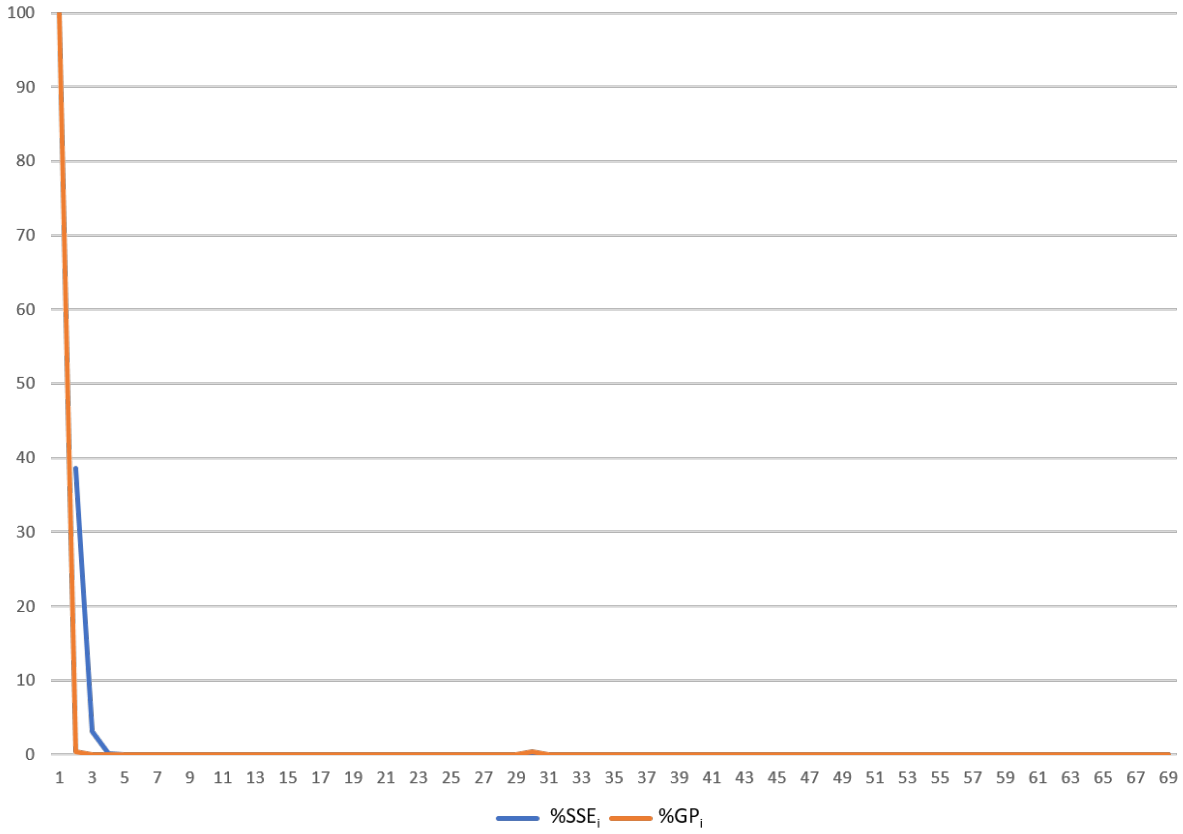


Abbildung 5.9.: Verlauf der $\%SSE_i$ und der $\%GP_i$ in Prozent während der Ausführung des K-means++ Algorithmus auf dem Datensatz *Spiral* mit $k = 50$

Tabelle 5.4.: Darstellung des Qualitätsverlusts und der Einsparung an Iterationen in Abhängigkeit von der eingesparten Ausführungszeit für den K-means auf den synthetischen Datensätzen

	%A größer als	>50%	>70%	>80%	>90%
	Anzahl	178	129	61	3
Min	%E	0,00%	0,00%	0,00%	0,12%
	%I	50,00%	56,25%	68,00%	86,67%
Median	%E	2,45%	2,82%	3,01%	0,20%
	%I	2,45%	2,82%	84,92%	92,08%
Max	%E	33,15%	33,15%	27,41%	4,39%
	%I	92,31%	92,31%	92,31%	4,39%

Tabelle 5.5.: Darstellung des Qualitätsverlusts und der Einsparung an Iterationen in Abhängigkeit von der eingesparten Ausführungszeit für den K-means++ auf den synthetischen Datensätzen

	%A größer als	>50%	>70%	>80%	>90%
	Anzahl	182/190	126/190	55/190	5/190
Min	%E	0,00%	0,00%	0,00%	0,04%
	%I	53,85%	67,86%	76,19%	83,87%
Median	%E	1,93%	2,30%	2,83%	4,69%
	%I	74,41%	78,37%	84,78%	90,61%
Max	%E	52,74%	52,74%	52,74%	21,99%
	%E	93,14%	93,14%	93,14%	93,14%

5.2.3. Mögliche Problematik der Qualitätsverluste

In diesem Unterabschnitt wird untersucht, wie es zu den teilweise großen Qualitätsverlusten kommen kann, insbesondere bei den realen Datensätzen.

Dazu wurde zunächst der Datensatz *3DRN* betrachtet. Für diesen gab es für den K-means einen Qualitätsverlust von 66,23% für k=45 (vgl. Tabelle 5.1). Es ist aber auch festzustellen, dass es eine Einsparung in der Ausführungszeit von 87,69% gab. In Abbildung 5.10 ist der Verlauf der $\%SSE_i$ und der $\%GP_i$ in Prozent zu sehen und zusätzlich sind die beiden Schwellenwerte $\varepsilon_1 = 1\%$ und $\varepsilon_2 = 10\%$ mit angegeben. Dabei ist der Schwellenwert für die $\%GP_i$ (ε_2) bereits nach Iteration 3 unterschritten und bleibt in der Folge auch unter diesem Schwellenwert. Die $\%SSE_i$ fällt in Iteration 9 unter ihren Schwellenwert (ε_1). Allerdings gibt es im späteren Verlauf noch einmal einen Anstieg bis zu fast 40% (Iteration 19-21). Durch diese große Änderung die sich noch ergibt entsteht dieser große Qualitätsverlust.

Eine Möglichkeit um dies zu verhindern wäre die Schwellenwerte ε_1 und ε_2 geringer zu setzen. Allerdings ist nicht klar wie gering diese dann gesetzt werden sollten, da dies sehr stark vom Datensatz abhängig ist. Es ist auch nicht garantiert, dass dadurch nicht auch große Qualitätsverluste entstehen, denn theoretisch könnte die $\%SSE_i$ beliebig nahe gegen 0 gehen und trotzdem können noch einmal Schwankungen nach oben auftreten. Zudem sind dann auch keine so großen Einsparungen in der Ausführungszeit möglich.

5.3. Interaktionspunkte

In diesem Abschnitt wird evaluiert, wie oft für die beiden Metriken $\%SSE_i$ und $\%GP_i$ Zeitpunkte für eine Visualisierung von Zwischenergebnissen bestimmt werden. Dazu sollen zunächst die Schwellenwerte für beide Metriken in Abschnitt 5.3.1 bestimmt werden. Anschließend

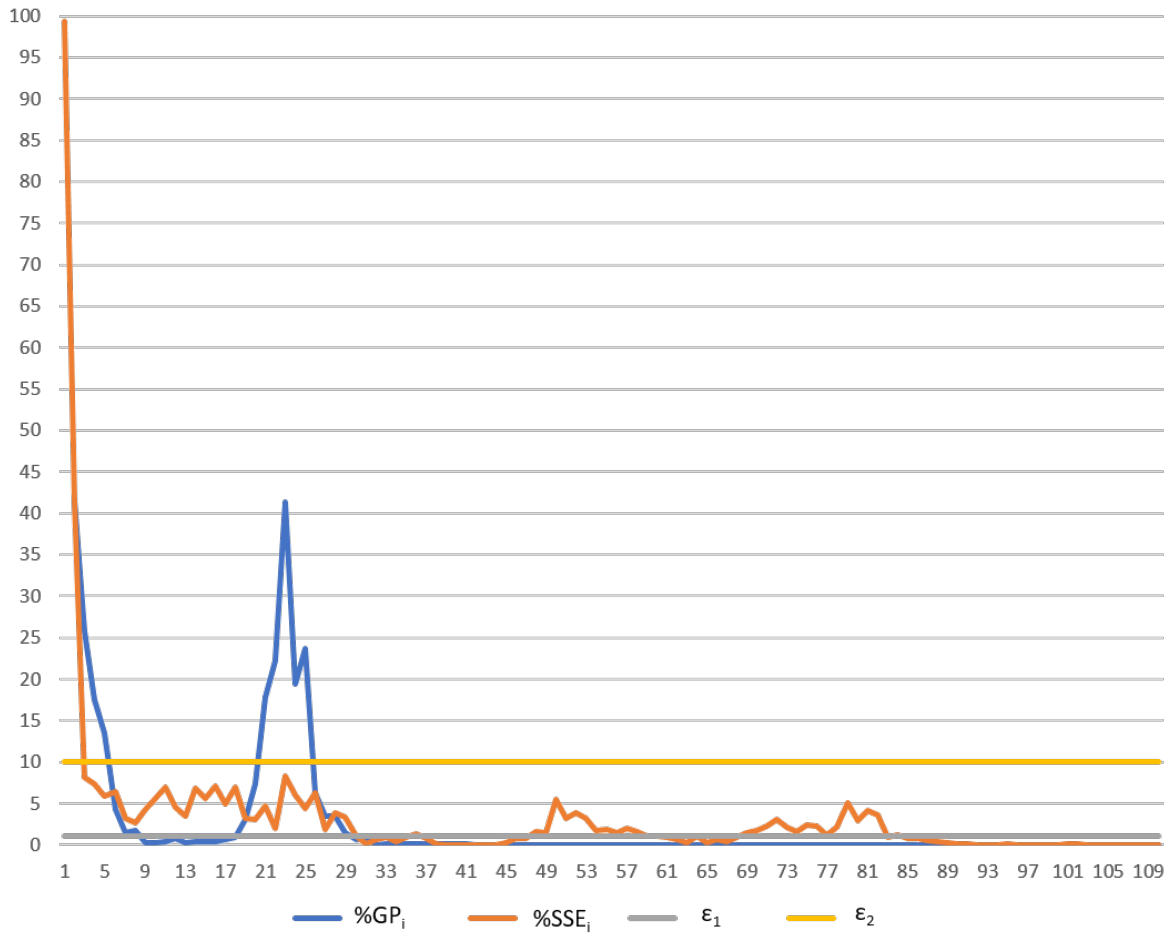


Abbildung 5.10.: Verlauf der $\%SSE_i$ und der $\%GP_i$ in Prozent während der Ausführung des K-means Algorithmus auf dem Datensatz *Online Retail* mit $k = 45$

werden die Ergebnisse für den K-means und den K-means++ in Abschnitt 5.3.2 bewertet. Die Bewertung für den DBSCAN erfolgt in Abschnitt 5.3.4.

5.3.1. Bestimmung der Schwellenwerte

Die Bestimmung eines Schwellenwertes gestaltet sich insofern schwierig, als das zunächst nicht klar ist, was überhaupt eine geeignete Anzahl an Visualisierungen darstellt. Allerdings ist festzustellen, dass beispielsweise ein Schwellenwert von 50% dafür sorgen würde, dass es nur sehr wenige Visualisierungen geben würde. Hinzu kommt, dass in Abschnitt 5.2 bereits diskutiert wurde, dass die SSE_i und die GP_i zwar ähnliche Verläufe haben, die Kurve SSE_i jedoch tendenziell unterhalb der GP_i liegt. Aus diesem Grund wird für beide Metriken eine unterschiedliche Anzahl an Visualisierungen für den gleichen Schwellenwert vermutet, wodurch zwei Ergebnisse entstehen, die anschließend miteinander verglichen werden können.

Als Schwellenwert wurde für beide Metriken der Wert 10% angenommen, wobei durchaus auch andere Werte wie zum Beispiel 5% denkbar wären. Größere Werte als 10% erscheinen allerdings nicht sinnvoll, da die Experimente gezeigt haben, dass es sonst relativ wenige Visualisierungen gibt.

5.3.2. Bewertung für K-means und K-means++

Für die Visualisierung ist eine Bewertung wie für die Konvergenzkriterien nicht möglich. Denn für die Visualisierung müsste für eine geeignete Bewertung die Interaktion mit einem Nutzer berücksichtigt werden. Anschließend könnte evaluiert werden, ob und wie viel Zeitaufwand auf Grund der Visualisierungen entsteht oder ob dadurch sogar Zeit eingespart wird, weil für einen Nutzer die Zwischenergebnisse ausreichend sind. Dies könnte beispielsweise durch eine Nutzerstudie evaluiert werden. Dies würde allerdings den Rahmen dieser Arbeit sprengen, weshalb dies zukünftigen Arbeiten überlassen bleibt.

In diesem Abschnitt wird daher nicht evaluiert wie viel zusätzlicher Aufwand durch die Visualisierung entsteht bzw. eingespart wird, sondern wie viele Visualisierungen im Verhältnis zu der Anzahl der Iterationen entstanden sind. Denn es ist wichtig, dass zum Einen eine Mindestanzahl an Visualisierungen vorhanden ist, um so eine bessere Nachvollziehbarkeit der Ergebnisse zu ermöglichen, zum anderen sollten nicht zu viele Visualisierungen entstehen, da dies sonst den Aufwand der Analyse deutlich erhöhen würde. Welche Werte dabei als untere und welche als obere Schranke sinnvoll sind, müsste ebenfalls in einer Nutzerstudie herausgefunden werden. Für einen besseren Vergleich der Ergebnisse wurden aber im Folgenden für eine untere Schranke 5% an Visualisierungen im Verhältnis zur Anzahl der Iterationen und 20% für eine obere Schranke angenommen. Im Folgenden wird %Vis verwendet als Abkürzung für die Anzahl der Visualisierungen im Verhältnis zur Anzahl der Iterationen in Prozent.

In Tabelle 5.6 werden für die realen Datensätze die Anzahl der Iterationen, so wie die %Vis für die SSE_i und die GP_i in Prozent angegeben und das sowohl für den K-means als auch für den K-means++. Zusätzlich wurden noch für jede Spalte die Min-, Max-, Median- und Mittelwerte angegeben. In der Tabelle ist zu erkennen, dass die %Vis für die SSE_i in den meisten Fällen niedriger ist, als für die %Vis für die GP_i.

Tabelle 5.6.: Visualisierungen, die in Abhängigkeit der Anzahl der Itertaionen, für den K-means und den K-means++ auf den realen Datensätzen entstanden sind

		K-means			K-means++		
Datensatz	#Cluster	#It	%Vis SSE_i	%Vis GP_i	#It	%Vis SSE_i	%Vis GP_i
Skin	5	22	23,08%	18,18%	29	10,34%	27,59%
Skin	10	35	13,79%	20,00%	171	2,92%	5,26%
Skin	15	44	12,50%	18,18%	142	1,41%	3,52%

Tabelle 5.6 fortgesetzt von vorheriger Seite

		K-means			K-means++		
Datensatz	#Cluster	#It	%Vis SSE_i	%Vis GP_i	#It	%Vis SSE_i	%Vis GP_i
Skin	20	44	14,71%	11,36%	190	2,11%	3,68%
Skin	25	47	12,12%	12,77%	129	3,10%	3,88%
Skin	30	68	6,67%	7,35%	62	6,45%	9,68%
Skin	35	53	12,82%	13,21%	59	8,47%	10,17%
Skin	40	84	6,67%	10,71%	57	8,77%	14,04%
Skin	45	61	10,20%	13,11%	67	2,99%	8,96%
Skin	50	77	7,81%	9,09%	69	5,80%	8,70%
Shuttle	5	166	2,41%	4,22%	19	5,26%	5,26%
Shuttle	10	127	3,15%	4,72%	37	5,41%	5,41%
Shuttle	15	116	5,17%	8,62%	39	7,69%	10,26%
Shuttle	20	163	2,45%	4,91%	45	4,44%	8,89%
Shuttle	25	129	3,10%	6,20%	39	5,13%	10,26%
Shuttle	30	115	6,96%	8,70%	42	7,14%	7,14%
Shuttle	35	105	6,67%	10,48%	49	4,08%	8,16%
Shuttle	40	99	5,05%	9,09%	44	6,82%	9,09%
Shuttle	45	95	6,32%	10,53%	63	4,76%	9,52%
Shuttle	50	98	6,12%	12,24%	71	5,63%	7,04%
OR	5	29	17,24%	17,24%	29	17,24%	13,79%
OR	10	35	25,71%	37,14%	33	24,24%	18,18%
OR	15	50	22,00%	28,00%	38	23,68%	15,79%
OR	20	42	28,57%	35,71%	38	13,16%	23,68%
OR	25	66	15,15%	31,82%	36	8,33%	16,67%
OR	30	56	19,64%	30,36%	44	15,91%	25,00%
OR	35	89	11,24%	20,22%	52	7,69%	9,62%
OR	40	84	13,10%	22,62%	53	13,21%	16,98%
OR	45	79	12,66%	17,72%	60	3,33%	11,67%
OR	50	73	12,33%	19,18%	44	11,36%	11,36%
IHEPC	5	34	5,88%	8,82%	34	8,82%	8,82%
IHEPC	10	59	6,78%	11,86%	45	2,22%	6,67%
IHEPC	15	124	1,61%	4,84%	56	3,57%	5,36%
IHEPC	20	116	2,59%	6,03%	112	1,79%	5,36%
IHEPC	25	161	2,48%	5,59%	124	0,81%	6,45%
IHEPC	30	179	2,23%	5,59%	131	2,29%	6,11%

Tabelle 5.6 fortgesetzt von vorheriger Seite

		K-means			K-means++		
Datensatz	#Cluster	#It	%Vis SSE_i	%Vis GP_i	#It	%Vis SSE_i	%Vis GP_i
IHEPC	35	201	1,99%	4,98%	150	2,00%	5,33%
IHEPC	40	203	1,97%	5,42%	213	0,94%	4,69%
IHEPC	45	277	1,81%	4,33%	165	1,82%	4,24%
IHEPC	50	275	1,09%	4,00%	225	1,33%	4,00%
3DRN	5	43	11,63%	13,95%	37	5,41%	8,11%
3DRN	10	74	14,86%	17,57%	56	5,36%	12,50%
3DRN	15	121	4,13%	9,92%	64	3,13%	6,25%
3DRN	20	149	6,71%	11,41%	56	7,14%	8,93%
3DRN	25	162	4,32%	11,73%	35	5,71%	8,57%
3DRN	30	177	8,47%	10,73%	39	10,26%	10,26%
3DRN	35	99	10,10%	9,09%	44	6,82%	6,82%
3DRN	40	116	5,17%	8,62%	42	4,76%	9,52%
3DRN	45	98	7,14%	7,14%	42	9,52%	11,90%
3DRN	50	100	5,00%	7,00%	41	4,88%	7,32%
	MIN	2	4,00%	19,00	0,81%	3,52%	
	MAX	15	37,14%	225,00	24,24%	27,59%	
	Median	5	10,62%	51	5,41%	8,86%	
	Mittelwert	6,24	12,85%	71	6,71%	9,73%	

In Tabelle 5.7 wird untersucht, wie das Verhalten für unterschiedliche Wertebereiche der %Vis ist. Dazu wurde der Bereich zwischen 5 und 20% genommen und geschaut für wie viele Durchläufe das zutrifft, wenn man diese Bereich verkleinert. Zusätzlich wurde geschaut bei wie vielen Durchläufen die %Vis unter 5% und bei wie vielen sie über 20% lag. Zu diesen wurde auch noch immer die Gesamtanzahl der Durchläufe dazu geschrieben. Alle Zeilen bei denen die Anzahl der Durchläufe größer als 25 war wurden mit grün markiert.

Auffällig dabei ist, dass sowohl für den K-means als auch für den K-means++ die %Vis für beide Metriken immer bei mehr als der Hälfte der Durchläufe im Bereich zwischen 5 und 20% liegt. Zudem ist zu erkennen, dass sich die meisten Durchläufe für die %Vis im Bereich zwischen 5 und 15% liegen und das jeweils für beide Metriken bei beiden Algorithmen. Für die GP_i liegt die Anzahl der Durchläufe für den K-means++, für die die %Vis im Bereich zwischen 5 und 10% liegt mit 27 auch über der Hälfte. Somit sind die Visualisierungen für die meisten Durchläufe im Bereich zwischen 5 und 20%. Zudem sind für mehr als die Hälfte der Durchläufe die Visualisierungen im Bereich zwischen 5 und 15%.

Tabelle 5.7.: Anzahl der Durchläufe für verschiedene Wertebereiche der %Vis für die realen Datensätze

		K-means		K-means++	
		%Vis SSE_i	%Vis GP_i	%Vis SSE_i	%Vis GP_i
Anzahl der Durchläufe für die die %Vis im Bereich x ist	$5\% \leq x \leq 20\%$	33/50	36/50	27/50	41/50
	$10\% \leq x \leq 20\%$	13/50	19/50	7/50	14/50
	$15\% \leq x \leq 20\%$	3/50	6/50	2/50	4/50
	$5\% \leq x \leq 15\%$	27/50	26/50	25/50	37/50
	$5\% \leq x \leq 10\%$	10/50	13/50	20/50	27/50
	$x < 5\%$	14/50	6/50	21/50	6/50
	$x > 20\%$	3/50	8/50	2/50	3/50

Wie gut geeignet diese Werte wiederum sind, muss eine Nutzerstudie ergeben. Sollte sich dabei ergeben, dass es zu viele Visualisierungen sind, können die Schwellenwerte für die Metriken entsprechend reduziert werden. Umgekehrt, wenn es zu wenig Visualisierungen sind, können die Schwellenwerte erhöht werden. Allerdings ist es selbst mit einer Nutzerstudie schwer zu evaluieren, was eine geeignete Zahl an Visualisierungen ist, da es vom Datensatz und vor allem vom Expertenwissen des Nutzers abhängt, was eine gute Anzahl an Visualisierungen ist.

5.3.3. Synthetische Datensätze

In Tabelle A.2 auf Seite 90 werden die Ergebnisse der Visualisierung für die synthetischen Datensätze dargestellt. Auch hier ist zu erkennen, dass die Visualisierungen für die GP_i tendenziell höher sind, als für die SSE_i. Des Weiteren sind die Iterationen für die synthetischen Datensätzen im Median bei 33, wohingegen es bei den realen Datensätzen 97 im Median waren. Dies könnte einer der Gründe sein, weshalb die % Vis im Median für beide Metriken bei beiden Algorithmen größer ist. Dennoch ist es auch hier der Fall, dass die meisten Durchläufe eine %Vis im Bereich von 5 und 20% auf. In Tabelle A.2 werden die Anzahl der Durchläufe dargestellt für unterschiedliche Werte Bereiche der %Vis. Die Zeilen, bei denen dies auf mehr als die Hälfte der Durchläufe zutraf, wurden grün markiert. Dabei ist zu sehen, dass die %Vis für einen sehr großen Teil der Durchläufe zwischen 5 und 20 % liegt.

5.3.4. DBSCAN

Für den DBSCAN hat sich gezeigt, dass die Metriken, die für den K-means und den K-means++ genutzt wurden, ungeeignet sind, da diese meist die Zentroide betrachten. Da der DBSCAN die Zentroide nicht während seiner Ausführung berechnet, wäre eine zusätzliche Berechnung der Zentroide ein zu großer zeitlicher Aufwand. Daher wurden die Kriterien für eine Visualisierung

Tabelle 5.8.: Anzahl der Durchläufe für unterschiedliche %Vis Bereiche für die synthetischen Datensätze

		K-means		K-means++	
		%Vis SSE_i	%Vis Gp_i	%Vis SSE_i	%Vis Gp_i
Anzahl der Durchläufe für die die %Vis im Bereich x ist	$5\% \leq x \leq 20\%$	165/190	177/190	156/190	166/190
	$10\% \leq x \leq 20\%$	87/190	77/190	45/190	109/190
	$15\% \leq x \leq 20\%$	63/190	44/190	9/190	18/190
	$5\% \leq x \leq 15\%$	141/190	134/190	148/190	149/190
	$5\% \leq x \leq 10\%$	83/190	34/190	111/190	17/190
	$x < 5\%$	18/190	8/190	34/190	19/190
	$x > 20\%$	7/190	5/190	2/190	5/190

für den DBSCAN anders festgelegt. Der erste Ansatz ist analog zu dem Ansatz, wie er für den K-means/++ verwendet wurde. Denn es wird visualisiert, wenn die Summe der Punkte, die ihre Clusterzugehörigkeit ändern, 10% übersteigt. Eine Änderung der Clusterzugehörigkeit meint in diesem Sinne, wenn ein Punkt als Ausreißer identifiziert wird oder wenn er einem Cluster zugeordnet wird. Das zweite Kriterium ist, dass immer visualisiert wird, wenn ein neues Cluster entsteht.

Bevor dies an den Datensätzen getestet werden konnte, mussten allerdings zuerst die Parameter für den Algorithmus bestimmt werden. Dazu wurde die Heuristik implementiert und angewendet, die in Abschnitt 2.3.3 auf Seite 31 vorgestellt wurde. In Tabelle 5.9 werden die Ergebnisse der Visualisierung für diese zwei Kriterien dargestellt. #Vis CE steht dabei für die Anzahl der Visualisierungen, wenn ein neues Cluster entsteht und #Vis für die Anzahl der Visualisierungen, die für die GP_i gemacht wurden. In der Tabelle ist zu erkennen, dass die #Vis CE immer größer oder gleich der #Vis GP_i ist. Dabei werden teilweise auch sehr große Werte angenommen. So sind für den synthetischen Datensatz *cluto-t8-8k* 222 Visualisierungen für die #Vis CE gemessen worden. Bei den realen Datensätzen waren es für den *3DRN* Datensatz sogar 1810 Visualisierungen. Zwar ist dadurch eine bessere Nachvollziehbarkeit möglich, allerdings steigt der Aufwand der Analyse stark an und es wäre ein nicht vertretbarer Zeitaufwand erforderlich. Für die #Vis GP_i hingegen, zumindest für die synthetischen Datensätze, scheinen die Anzahl der Visualisierungen vergleichbar mit denen, wie sie für den K-means und den K-means++ gemacht wurden. Für die realen Datensätze sind für den *Shuttle* Datensatz 1, für den *Shuttle* Datensatz 1 uns den *3DRN* Datensatz 0 Visualisierungen. Der Grund hierfür könnte in der Wahl der Parameter liegen, da die Parameter manuell aus dem k-Distanzgraphen abgelesen wurden.

Tabelle 5.9.: Visualisierungen für den DBSCAN für die realen und synthetischen Datensätze

	Datensatz	#Vis GP_i	#Vis CE
Real	Skin	2	368
	3DRN	0	1810
	Shuttle	1	1
	OR	10	578
	IHEPC	12	1672
Synthetisch	a1	19	28
	a2	26	34
	a3	40	49
	Aggregation	8	8
	D31	31	31
	R15	15	15
	s1	15	46
	s2	13	39
	s3	8	40
	s4	7	53
	spiral	3	3
	unbalance	3	5
	chainlink	16	16
	cluto-t4-8k	18	85
	cluto-t5-8k	22	256
	cluto-t7-10k	29	208
	cluto-t8-8k	14	222
	diamond9	26	89
	engytime	5	101

5.4. Visualisierungen bis zur Konvergenz

In diesem Abschnitt werden die Ansätze für die Konvergenz und die Anzahl der Zeitpunkte für eine Visualisierung miteinander verknüpft. In Tabelle 5.10 wird dafür für den K-means und den K-means++ die Anzahl der Iterationen bis zur frühzeitigen Terminierung durch das Konvergenzkriterium dargestellt. Dazu wird Anzahl der Visualisierungen für beide Metriken ins Verhältnis zur Anzahl der Iterationen bis zur frühzeitigen Terminierung gesetzt. Die Anzahl der Visualisierungen im Verhältnis zur Anzahl der Iterationen bis zur Konvergenz wird mit %Vis$_{konv}$ bezeichnet. Dabei ist auffällig, dass die %Vis$_{konv}$ für beide Metriken deutlich höher ist, als dies ohne die Konvergenz der Fall war. So sind teilweise Werte bis zu 100% zu erkennen und der Median liegt beim K-means für die SE_i bei 50%, was bedeutet, dass für diese bis zur

Konvergenz jede zweite Iteration visualisiert wird. Ein Beispiel, bei dem alle Iterationen bis zur Konvergenz mit der GP_i visualisiert werden, ist für den Datensatz *OR* mit $k = 25$ gegeben. In Tabelle 5.6 ist zu erkennen, dass für diesen Durchlauf 66 Iterationen ohne frühzeitige Terminierung erreicht wurden. Das bedeutet, dass sogar alle Visualisierungen für diesen Durchlauf bis zur frühzeitigen Terminierung durch das Konvergenzkriterium stattfinden. Für die GP_i ist der Median wert sogar bei 80%. Für den K-means++ ist die %Vis$_konv$ für die SSE_i mit 46,68% ähnlich dem K-means. Für die GP_i hingegen ist ein deutlicher Unterschied zu erkennen, da diese im Median bei 48,82% liegt.

Tabelle 5.10.: %Vis$_{konv}$ für den K-means und den K-means++ auf den realen Datensätzen

		K-meas			K-means++		
Datensatz	#Cluster	#I$_{konv}$	%Vis SSE_i	%Vis GP_i	#I$_{konv}$	%Vis SSE_i	%Vis GP_i
Skin	5	10	30,00%	40,00%	6	32,73%	49,09%
Skin	10	7	57,14%	100,00%	13	23,27%	46,53%
Skin	15	13	30,77%	61,54%	8	48,86%	61,08%
Skin	20	11	45,45%	45,45%	11	37,50%	37,50%
Skin	25	15	26,67%	40,00%	11	36,47%	36,47%
Skin	30	9	44,44%	55,56%	14	29,41%	29,41%
Skin	35	15	33,33%	46,67%	11	45,99%	45,99%
Skin	40	10	50,00%	90,00%	19	20,83%	26,04%
Skin	45	12	41,67%	66,67%	5	86,89%	86,89%
Skin	50	14	35,71%	50,00%	15	25,97%	25,97%
Shuttle	5	7	57,14%	100,00%	47	8,43%	12,65%
Shuttle	10	5	80,00%	100,00%	4	92,91%	92,91%
Shuttle	15	8	75,00%	75,00%	4	45,98%	68,97%
Shuttle	20	10	40,00%	80,00%	8	50,00%	87,50%
Shuttle	25	8	50,00%	100,00%	15	26,14%	39,20%
Shuttle	30	9	88,89%	77,78%	14	58,63%	51,30%
Shuttle	35	9	77,78%	88,89%	7	100,00%	100,00%
Shuttle	40	10	50,00%	90,00%	12	40,40%	48,48%
Shuttle	45	8	75,00%	75,00%	13	47,37%	47,37%
Shuttle	50	9	66,67%	88,89%	18	34,11%	34,11%
OR	5	14	35,71%	35,71%	12	42,15%	42,15%
OR	10	23	39,13%	56,52%	17	54,45%	78,66%
OR	15	22	50,00%	63,64%	17	60,00%	78,00%
OR	20	21	57,14%	71,43%	14	78,57%	92,86%
OR	25	21	47,62%	100,00%	13	79,12%	100,00%

		K-meas			K-means++		
Datensatz	#Cluster	#I$_{konv}$	%Vis SSE_i	%Vis GP_i	#I$_{konv}$	%Vis SSE_i	%Vis GP_i
OR	30	19	57,89%	89,47%	9	69,64%	81,25%
OR	35	17	58,82%	94,12%	17	51,83%	51,83%
OR	40	17	64,71%	100,00%	11	65,48%	84,18%
OR	45	18	55,56%	77,78%	9	86,08%	86,08%
OR	50	18	50,00%	77,78%	20	45,21%	40,18%
IHEPC	5	7	28,57%	42,86%	4	55,88%	55,88%
IHEPC	10	3	66,67%	100,00%	7	55,59%	69,49%
IHEPC	15	5	40,00%	80,00%	11	17,74%	26,61%
IHEPC	20	8	37,50%	87,50%	6	48,28%	48,28%
IHEPC	25	9	44,44%	100,00%	5	72,88%	91,10%
IHEPC	30	8	50,00%	100,00%	8	48,52%	72,79%
IHEPC	35	9	44,44%	88,89%	13	31,40%	47,10%
IHEPC	40	9	44,44%	100,00%	8	50,39%	62,98%
IHEPC	45	7	71,43%	85,71%	6	71,17%	100,00%
IHEPC	50	10	30,00%	70,00%	13	22,64%	45,27%
3DRN	5	10	50,00%	60,00%	6	85,27%	85,27%
3DRN	10	21	52,38%	61,90%	10	87,84%	97,60%
3DRN	15	28	17,86%	42,86%	23	12,95%	17,26%
3DRN	20	18	55,56%	94,44%	14	41,61%	48,55%
3DRN	25	17	41,18%	88,24%	33	12,04%	21,06%
3DRN	30	14	92,86%	92,86%	39	28,41%	20,66%
3DRN	35	17	58,82%	52,94%	12	69,26%	51,95%
3DRN	40	10	60,00%	100,00%	24	12,64%	12,64%
3DRN	45	8	87,50%	87,50%	25	20,41%	16,33%
3DRN	50	12	41,67%	58,33%	13	23,25%	31,00%
Min		3	17,86%	35,71%	4	8,43%	12,64%
Max		28	92,86%	100,00%	47	100,00%	100,00%
Median		10	50,00%	80,00%	12	46,68%	48,82%
Mittelwert		12,38	51,75%	76,64%	13	47,85%	55,69%

6. Zusammenfassung und Ausblick

In dieser Arbeit wurde ein Konvergenzkriterium anhand von 2 Metriken aufgestellt und es wurden Zeitpunkte für eine Visualisierung von Zwischenergebnissen bestimmt. Dazu wurde in Abschnitt 3.3 ein Konzept und eine prototypische Implementierung entwickelt, bei der für den K-means und den K-means++ eine effiziente Berechnung der Metriken ermöglicht wurde. In Abschnitt 5.1 wurde veranschaulicht, dass Berechnungsaufwand für die gesamte Berechnung der Metriken in jeder Iteration insgesamt deutlich unter 1% der Ausführungszeit der Algorithmen lag. Zudem wurde ein Konvergenzkriterium aufgestellt, dass auf der Änderung der quadratischen Fehlerfunktion im Vergleich zur vorherigen Iteration (mit $\%SSEi$ bezeichnet), so wie auf der Änderung der Anzahl der geänderten Punkte seit der letzten Iteration im Verhältnis zu allen Punkten (mit $\%GPi$ bezeichnet) basiert. Das Konvergenzkriterium sieht dabei vor, dass sobald diese Metriken einen bestimmten Schwellenwert überschreiten, dass der Algorithmus dann terminiert wird. In den Experimenten konnte gezeigt werden, dass sowohl für synthetische, als auch für reale Datensätze Einsparungen in den Iterationen und der Ausführungszeit von über 90% mit geringem Qualitätsverlust möglich waren. So konnten für den *Shuttle* Datensatz 85,02% der Ausführungszeit eingespart werden bei einem Qualitätsverlust von 0,21%. Für den Datensatz mit den meisten Instanzen, den *IHEPC* Datensatz konnte ähnliches erreicht werden. Für diesen war eine Einsparung der Ausführungszeit von 89,5% bei einem Qualitätsverlust von 0,83% möglich. Zudem wurde verdeutlicht, dass für eine geringere Einsparung der Ausführungszeit ein geringerer Qualitätsverlust vorhanden war. So war für den K-means++ bei einer Einsparung von 50% der Qualitätsverlust im Median bei 4,26%. Für eine Einsparung von 90% hingegen, lag der Qualitätsverlust im Median bei 18,06%. Ein Problem das sich ergab war, dass sowohl die $\%SSE_i$, als auch die $\%GP_i$ keinen monotonen Verlauf aufwiesen. Das bedeutet, dass Fälle aufgetreten sind, bei denen die $\%SSE_i$ und die $\%GP_i$ nach dem sie unter einen bestimmten Schwellenwert gefallen sind, noch einmal anstiegen. Dieses Verhalten wurde allerdings nur bei 12% der Durchläufe, die im Median konvergiert haben, beobachtet. Zudem lassen sich diese Fälle auch nicht vermeiden, da der K-means nur gegen ein lokales und nicht gegen ein globales Minimum der SSE konvergiert.

Des Weiteren wurden diese beiden Metriken genutzt, um geeignete Zeitpunkte für eine Visualisierung auszumachen. Dabei ergaben sich für beide Metriken, die für jeweils beide Algorithmen geprüft wurden, unterschiedliche Werte für die Anzahl der Visualisierungen. Diese erstreckten sich von 0% bis 30% der Anzahl der Iterationen des jeweiligen Durchlaufs. Es wurde zudem aufgezeigt, dass sich für beide Metriken die meisten Durchläufe Visualisierungen im Bereich von 5% bis 20% der gesamten Anzahl an Iterationen aufwiesen. Mehr als die Hälfte der Durchläufe wies zudem Visualisierungen im eingeschränkteren Bereich von 5% bis 15% auf. Anschließend

wurde untersucht wie viele Visualisierungen sich bis zum festgelegten Konvergenzkriterium ergeben. Dabei hat sich herausgestellt, dass sich dafür deutlich mehr Visualisierungen im Verhältnis von Visualisierungen und Iterationen bis Konvergenz ergab. Dabei waren Visualisierungen von bis zu 100% vorhanden und im Median lagen die Visualisierungen für beide Algorithmen knapp unter 50 %. Nur bei der GP_i für den K-means lag der Median-Wert bei 80%.

Ausblick

In zukünftigen Arbeiten kann versucht werden das in dieser erstellte Konzept auf bereits existierende System anzuwenden. So könnte untersucht werden, ob sich das Konvergenzkriterium in ähnlicher Weise auch für bereits bekannte verteilte Systeme, wie z. B. Apache Hadoop[1] oder Apache Spark[2] anwenden lässt und ob dabei ähnliche Ergebnisse in Hinblick zur Zeiteinsparung und Qualitätsverlust erzielt werden können. Zudem könnte erforscht werden, ob das Konzept auch skalierbar ist. Das heißt zum einen, dass es für deutlich größere Datenmengen noch effizient einsetzbar ist und ob es möglich ist die Berechnungen auf mehrere Rechner zu verteilen, also beispielsweise in einem Cluster. Dabei könnte auch untersucht werden, ob die Ergebnisse durch Vorverarbeitungsschritte, wie z.B. das *Data Sampling*, also wenn die Analyse nur auf einer Teilmenge der Daten durchgeführt wird, verbessert werden können.

Des Weiteren könnte für eine Evaluation, wie geeignet die Zeitpunkte der Visualisierungen sind eine Nutzerstudie durchgeführt werden. Dazu müsste untersucht werden, ob zum einen die Ergebnisse besser nachvollziehbar und besser interpretiert werde können im Vergleich ohne Visualisierung von Zwischenergebnissen und zum Anderen, wie sich der Mehraufwand, der durch die Visualisierungen entsteht, verhält. Für ersteres müsste geschaut werden, wie gut die Ergebnisse von einem Nutzer mit Visualisierung ausgewertet wird und wie die Auswertung ohne Visualisierung erfolgt. Für letzteres müsste zusätzlich die Zeit erfasst werden, die insgesamt für die Analyse anfällt. Zusätzlich könnte in dieser Studie untersucht werden, wie die Interpretation der Ergebnisse und der Zeitaufwand für eine Kombination des Konvergenzkriteriums und der Visualisierungen ausfällt.

[1] http://hadoop.apache.org/
[2] https://spark.apache.org/

A. Anhang

A.1. Tabelle der Ergebnisse der Konvergenz für die synthetischen Datensätze

		K-means				K-means++			
Datensatz	k	%I	%A	%E	SSE_C	%I2	%A3	%E4	SSE_C
a1	5	86,36%	81,77%	2,13%	1,31E+11	78,26%	78,95%	2,62%	1,31E+11
a1	10	64,29%	61,74%	0,70%	5,16E+10	77,14%	79,10%	3,63%	5,69E+10
a1	15	70,97%	69,68%	6,29%	3,22E+10	75,86%	74,60%	1,32%	3,14E+10
a1	20	64,00%	61,14%	13,08%	1,62E+10	46,67%	46,15%	0,00%	1,22E+10
a1	25	77,42%	78,52%	26,74%	1,37E+10	82,14%	81,82%	1,94%	1,39E+10
a1	30	83,78%	82,42%	21,83%	1,14E+10	78,57%	68,84%	1,08%	8,61E+09
a1	35	73,08%	70,70%	1,04%	7,61E+09	81,25%	78,23%	2,24%	7,38E+09
a1	40	74,07%	73,90%	2,30%	7,07E+09	77,78%	71,43%	1,62%	6,10E+09
a1	45	74,07%	73,49%	1,74%	6,15E+09	66,67%	49,64%	0,45%	4,84E+09
a1	50	69,57%	67,80%	2,67%	4,91E+09	70,83%	65,43%	0,25%	4,07E+09
a2	5	55,00%	42,70%	0,66%	4,15E+11	76,47%	71,05%	2,41%	4,19E+11
a2	10	78,57%	79,37%	0,37%	1,99E+11	85,37%	87,60%	14,39%	2,27E+11
a2	15	65,52%	66,90%	5,10%	1,12E+11	66,67%	67,63%	1,55%	1,30E+11
a2	20	73,33%	73,74%	5,03%	8,03E+10	87,50%	89,70%	13,22%	8,15E+10
a2	25	70,97%	71,71%	4,76%	5,68E+10	75,76%	75,71%	1,20%	5,32E+10
a2	30	86,96%	86,67%	7,76%	4,76E+10	75,00%	75,33%	3,22%	4,20E+10
a2	35	64,29%	65,54%	1,07%	3,12E+10	71,43%	67,03%	0,43%	3,33E+10
a2	40	68,00%	69,92%	18,57%	3,46E+10	78,13%	72,84%	2,12%	2,45E+10
a2	45	70,37%	69,28%	0,98%	2,24E+10	74,07%	73,19%	1,13%	1,83E+10
a2	50	74,19%	75,59%	15,21%	1,86E+10	76,00%	71,91%	2,23%	1,48E+10
a3	5	75,00%	47,50%	0,34%	8,88E+11	81,82%	82,11%	2,83%	9,52E+11
a3	10	86,57%	84,19%	10,52%	4,84E+11	90,48%	90,55%	21,99%	5,76E+11
a3	15	79,59%	81,25%	0,92%	2,76E+11	87,88%	90,61%	4,69%	2,70E+11
a3	20	65,52%	63,96%	6,16%	1,93E+11	74,19%	76,50%	4,56%	1,89E+11
a3	25	81,82%	80,54%	4,65%	1,43E+11	80,95%	81,27%	4,24%	1,33E+11

Tabelle A.1 Fortsetzung von vorheriger Seite

		K-means				K-means++			
Datensatz	k	%I	%A	%E	SSE_C	%I2	%A3	%E4	SSE_C
a3	30	56,00%	58,49%	1,72%	1,02E+11	76,47%	74,35%	5,02%	1,08E+11
a3	35	62,96%	64,81%	1,50%	7,91E+10	72,00%	70,78%	0,69%	8,05E+10
a3	40	70,00%	70,53%	7,35%	7,21E+10	76,67%	76,63%	2,27%	7,04E+10
a3	45	74,19%	70,53%	5,28%	6,42E+10	73,33%	73,90%	2,72%	5,97E+10
a3	50	62,50%	57,71%	4,76%	5,42E+10	72,00%	71,00%	2,19%	4,38E+10
Aggregation	5	53,85%	66,67%	0,02%	1,70E+04	79,17%	62,96%	35,57%	2,63E+04
Aggregation	10	70,37%	74,07%	1,20%	6,71E+03	63,16%	63,16%	0,28%	7,21E+03
Aggregation	15	63,16%	60,00%	1,06%	5,22E+03	73,33%	64,52%	1,46%	3,97E+03
Aggregation	20	63,16%	63,64%	0,37%	3,76E+03	72,41%	53,85%	5,53%	2,48E+03
Aggregation	25	69,23%	72,00%	2,21%	2,65E+03	64,71%	59,26%	0,22%	1,59E+03
Aggregation	30	76,67%	76,92%	1,36%	1,36E+03	80,65%	77,55%	3,31%	1,03E+03
Aggregation	35	65,22%	60,71%	0,05%	7,66E+02	75,86%	53,13%	1,71%	7,59E+02
Aggregation	40	77,50%	77,94%	6,11%	8,92E+02	87,50%	81,88%	0,10%	4,27E+02
Aggregation	45	83,33%	78,87%	0,46%	6,64E+02	83,33%	62,07%	0,00%	2,44E+02
Aggregation	50	69,23%	73,49%	0,00%	6,04E+02	81,48%	63,64%	0,00%	1,74E+02
chainlink	5	74,07%	76,00%	0,65%	3,42E+02	83,87%	90,32%	5,05%	3,59E+02
chainlink	10	47,37%	34,62%	0,21%	1,24E+02	71,88%	61,90%	4,90%	1,25E+02
chainlink	15	69,23%	67,14%	0,58%	7,85E+01	57,14%	55,00%	0,31%	5,75E+01
chainlink	20	67,65%	61,40%	16,57%	4,29E+01	63,16%	42,42%	2,46%	3,20E+01
chainlink	25	56,52%	59,46%	1,32%	3,60E+01	73,08%	68,89%	5,96%	1,98E+01
chainlink	30	70,73%	55,91%	14,04%	1,71E+01	70,83%	60,00%	8,75%	1,56E+01
chainlink	35	50,00%	50,00%	0,05%	1,13E+01	68,18%	61,29%	0,31%	9,15E+00
chainlink	40	65,22%	76,77%	1,56%	1,01E+01	80,77%	67,95%	1,17%	7,04E+00
chainlink	45	75,00%	76,19%	0,00%	1,07E+01	70,00%	62,90%	7,72%	5,41E+00
chainlink	50	80,49%	60,76%	6,94%	8,24E+00	73,91%	58,88%	18,40%	2,93E+00
cluto-t4-8k	5	52,94%	41,25%	0,09%	2,87E+07	42,11%	42,00%	0,03%	2,85E+07
cluto-t4-8k	10	81,82%	81,16%	2,02%	1,43E+07	84,44%	84,85%	0,45%	1,56E+07
cluto-t4-8k	15	72,73%	58,38%	0,92%	8,50E+06	69,57%	69,34%	3,20%	9,22E+06
cluto-t4-8k	20	75,00%	63,23%	3,10%	5,96E+06	67,86%	70,05%	1,18%	6,03E+06
cluto-t4-8k	25	66,67%	66,78%	0,45%	4,42E+06	73,53%	71,22%	1,78%	4,55E+06
cluto-t4-8k	30	72,22%	72,30%	1,76%	3,87E+06	77,27%	74,25%	2,01%	3,36E+06
cluto-t4-8k	35	67,74%	52,53%	5,24%	2,86E+06	78,13%	61,87%	10,03%	3,08E+06
cluto-t4-8k	40	74,29%	73,98%	1,98%	2,32E+06	81,25%	80,25%	2,92%	2,25E+06
cluto-t4-8k	45	71,88%	66,80%	2,32%	2,19E+06	84,91%	84,41%	3,37%	2,01E+06
cluto-t4-8k	50	65,52%	65,40%	1,97%	1,63E+06	77,78%	78,14%	1,23%	1,69E+06
cluto-t5-8k	5	85,29%	85,86%	2,65%	2,79E+07	73,33%	73,68%	2,73%	1,86E+07

Tabelle A.1 Fortsetzung von vorheriger Seite

		K-means				K-means++			
Datensatz	k	%I	%A	%E	SSE_C	%I2	%A3	%E4	SSE_C
cluto-t5-8k	10	76,92%	71,96%	0,74%	7,41E+06	85,71%	84,78%	0,40%	7,08E+06
cluto-t5-8k	15	85,00%	84,07%	1,87%	5,67E+06	84,38%	84,97%	0,27%	5,66E+06
cluto-t5-8k	20	81,82%	72,93%	4,08%	3,67E+06	87,23%	86,10%	2,50%	3,12E+06
cluto-t5-8k	25	80,00%	63,84%	1,83%	2,61E+06	76,92%	72,73%	1,91%	2,23E+06
cluto-t5-8k	30	80,00%	78,12%	1,13%	1,72E+06	76,00%	75,73%	1,33%	1,61E+06
cluto-t5-8k	35	75,76%	73,05%	3,46%	1,65E+06	77,42%	77,46%	0,74%	1,19E+06
cluto-t5-8k	40	74,29%	74,89%	1,48%	1,38E+06	69,23%	67,65%	1,18%	1,06E+06
cluto-t5-8k	45	72,97%	72,41%	3,66%	8,80E+05	86,21%	84,60%	5,99%	9,01E+05
cluto-t5-8k	50	76,47%	75,09%	3,53%	8,28E+05	77,50%	75,50%	5,59%	6,76E+05
cluto-t7-10k	5	45,00%	42,03%	0,08%	7,14E+07	50,00%	49,15%	0,10%	7,12E+07
cluto-t7-10k	10	86,67%	77,70%	4,79%	3,70E+07	70,00%	68,71%	0,36%	3,75E+07
cluto-t7-10k	15	93,89%	88,78%	3,46%	2,28E+07	87,27%	85,71%	11,65%	2,31E+07
cluto-t7-10k	20	79,07%	78,04%	1,73%	1,57E+07	84,44%	63,88%	1,12%	1,55E+07
cluto-t7-10k	25	85,00%	80,13%	6,34%	1,28E+07	82,93%	80,77%	2,88%	1,24E+07
cluto-t7-10k	30	88,73%	80,70%	3,81%	1,11E+07	85,71%	84,32%	5,50%	9,33E+06
cluto-t7-10k	35	84,21%	78,87%	14,73%	8,55E+06	82,61%	81,73%	9,56%	7,64E+06
cluto-t7-10k	40	72,50%	71,11%	3,53%	6,22E+06	87,50%	87,22%	5,26%	5,88E+06
cluto-t7-10k	45	82,14%	80,13%	5,11%	7,28E+06	78,13%	77,21%	1,95%	5,04E+06
cluto-t7-10k	50	68,97%	61,44%	1,69%	4,73E+06	78,38%	78,60%	4,41%	4,22E+06
cluto-t8-8k	5	55,56%	54,39%	0,09%	5,90E+07	63,16%	66,00%	0,22%	5,89E+07
cluto-t8-8k	10	75,76%	73,72%	5,51%	2,78E+07	82,35%	83,55%	17,85%	3,07E+07
cluto-t8-8k	15	83,02%	84,92%	5,44%	1,84E+07	80,00%	79,41%	5,83%	1,77E+07
cluto-t8-8k	20	84,31%	68,13%	3,43%	1,29E+07	86,54%	85,35%	8,63%	1,34E+07
cluto-t8-8k	25	83,64%	83,75%	6,96%	1,08E+07	77,78%	76,04%	1,01%	9,76E+06
cluto-t8-8k	30	75,00%	73,85%	6,67%	7,93E+06	85,11%	83,40%	3,65%	7,66E+06
cluto-t8-8k	35	78,72%	77,10%	4,29%	6,85E+06	77,27%	78,37%	1,84%	6,22E+06
cluto-t8-8k	40	76,32%	73,71%	2,52%	5,52E+06	74,19%	70,35%	1,73%	5,28E+06
cluto-t8-8k	45	82,46%	84,68%	6,03%	4,90E+06	81,82%	80,81%	2,81%	4,57E+06
cluto-t8-8k	50	84,48%	83,84%	24,90%	5,26E+06	71,88%	70,86%	4,62%	4,02E+06
D31	5	41,67%	45,00%	0,00%	1,96E+05	68,42%	73,08%	2,95%	2,03E+05
D31	10	82,93%	73,96%	33,15%	1,07E+05	45,45%	41,67%	0,00%	7,43E+04
D31	15	42,86%	40,91%	0,00%	4,43E+04	75,00%	73,68%	9,41%	5,34E+04
D31	20	53,85%	53,85%	0,07%	3,36E+04	70,59%	70,49%	8,21%	2,65E+04
D31	25	58,82%	57,69%	0,00%	2,19E+04	72,22%	64,44%	3,76%	2,50E+04
D31	30	68,18%	72,06%	6,59%	1,14E+04	70,59%	69,37%	0,03%	1,27E+04
D31	35	72,00%	72,78%	12,55%	1,31E+04	76,92%	73,38%	0,59%	7,60E+03

Tabelle A.1 Fortsetzung von vorheriger Seite

		K-means				K-means++			
Datensatz	k	%I	%A	%E	SSE_C	%I2	%A3	%E4	SSE_C
D31	40	50,00%	50,40%	0,30%	6,01E+03	79,17%	75,63%	1,74%	4,69E+03
D31	45	77,78%	77,19%	0,43%	8,49E+03	82,35%	81,02%	0,75%	3,56E+03
D31	50	66,67%	66,81%	0,80%	5,41E+03	88,68%	85,05%	0,44%	3,21E+03
diamond9	5	87,50%	92,08%	4,39%	3,62E+03	73,68%	75,86%	1,12%	3,49E+03
diamond9	10	72,00%	73,17%	0,60%	9,63E+02	84,85%	82,46%	1,00%	1,49E+03
diamond9	15	87,18%	87,76%	2,19%	7,55E+02	79,17%	80,70%	1,41%	7,80E+02
diamond9	20	78,57%	81,19%	3,96%	5,86E+02	71,43%	73,85%	1,36%	5,67E+02
diamond9	25	77,78%	77,69%	6,99%	4,79E+02	55,56%	57,14%	0,88%	4,02E+02
diamond9	30	83,33%	83,25%	8,60%	4,11E+02	73,08%	70,83%	2,83%	3,42E+02
diamond9	35	75,68%	76,09%	18,13%	3,22E+02	80,49%	74,41%	7,26%	2,49E+02
diamond9	40	69,70%	69,10%	1,37%	2,11E+02	73,08%	69,80%	4,40%	2,14E+02
diamond9	45	78,38%	80,37%	7,68%	1,84E+02	75,86%	73,40%	2,26%	1,53E+02
diamond9	50	77,14%	79,38%	9,90%	1,48E+02	75,76%	68,18%	1,46%	1,26E+02
engytime	5	82,35%	83,93%	1,91%	4,56E+03	86,54%	82,72%	1,09%	5,19E+03
engytime	10	82,98%	84,21%	2,03%	2,44E+03	88,14%	84,83%	8,07%	2,61E+03
engytime	15	76,92%	77,65%	6,68%	1,78E+03	86,00%	86,06%	2,33%	1,66E+03
engytime	20	84,62%	81,23%	2,65%	1,32E+03	85,45%	85,46%	7,37%	1,29E+03
engytime	25	82,14%	82,69%	4,40%	1,03E+03	84,78%	81,55%	5,64%	9,75E+02
engytime	30	77,78%	70,69%	2,68%	8,37E+02	86,67%	87,61%	8,58%	8,34E+02
engytime	35	87,32%	87,10%	5,37%	7,26E+02	81,82%	76,58%	2,83%	6,28E+02
engytime	40	79,17%	78,46%	3,55%	5,87E+02	75,68%	72,88%	3,59%	5,24E+02
engytime	45	70,59%	66,58%	3,83%	5,24E+02	86,44%	86,86%	2,42%	4,11E+02
engytime	50	76,74%	81,64%	1,52%	4,77E+02	77,14%	73,05%	10,98%	3,55E+02
R15	5	82,14%	87,50%	13,81%	5,63E+03	60,00%	66,67%	0,00%	5,99E+03
R15	10	68,75%	70,83%	5,85%	1,84E+03	63,64%	33,33%	0,00%	1,32E+03
R15	15	60,00%	62,50%	0,02%	4,63E+02	80,00%	61,90%	0,13%	4,14E+02
R15	20	66,67%	31,03%	0,00%	9,67E+02	80,00%	64,44%	16,70%	2,27E+02
R15	25	76,19%	76,47%	0,03%	3,54E+02	84,00%	64,58%	0,11%	1,63E+02
R15	30	83,87%	79,66%	2,53%	1,55E+02	77,27%	66,67%	1,12%	6,15E+01
R15	35	87,76%	88,64%	0,59%	3,84E+02	81,48%	57,78%	0,01%	4,16E+01
R15	40	88,37%	88,89%	0,44%	1,62E+02	90,48%	74,34%	0,21%	6,20E+01
R15	45	88,10%	86,81%	0,00%	1,01E+02	90,48%	78,45%	0,17%	2,14E+01
R15	50	82,76%	85,19%	0,00%	1,15E+02	84,85%	59,38%	0,00%	1,10E+01
s1	5	64,71%	60,00%	8,07%	1,23E+14	62,50%	68,29%	0,00%	1,26E+14
s1	10	50,00%	52,94%	0,00%	3,51E+13	58,33%	60,61%	0,00%	4,78E+13
s1	15	52,94%	58,21%	0,00%	1,52E+13	69,57%	67,05%	0,01%	2,05E+13

Tabelle A.1 Fortsetzung von vorheriger Seite

		K-means				K-means++			
Datensatz	k	%I	%A	%E	SSE_C	%I2	%A3	%E4	SSE_C
s1	20	89,80%	89,33%	1,26%	1,78E+13	88,24%	89,09%	0,81%	1,39E+13
s1	25	77,78%	76,51%	1,22%	7,45E+12	85,29%	85,28%	2,23%	7,30E+12
s1	30	81,25%	82,35%	3,74%	6,94E+12	83,33%	80,81%	2,09%	6,07E+12
s1	35	80,65%	82,19%	8,17%	5,85E+12	80,00%	76,86%	2,06%	5,46E+12
s1	40	72,41%	74,90%	2,26%	4,79E+12	75,00%	72,09%	6,11%	4,33E+12
s1	45	70,83%	57,05%	2,60%	4,94E+12	69,57%	67,22%	1,05%	3,87E+12
s1	50	82,98%	81,38%	3,60%	3,72E+12	75,00%	71,74%	0,98%	3,34E+12
s2	5	60,00%	46,94%	0,25%	1,05E+14	53,85%	52,17%	0,01%	9,90E+13
s2	10	70,83%	69,23%	1,20%	4,51E+13	71,43%	70,18%	1,20%	3,99E+13
s2	15	79,49%	81,93%	27,41%	2,61E+13	66,67%	58,44%	0,29%	1,87E+13
s2	20	82,86%	79,14%	1,29%	1,19E+13	76,19%	80,49%	0,43%	1,54E+13
s2	25	76,92%	70,50%	0,40%	1,71E+13	86,49%	82,17%	3,51%	1,03E+13
s2	30	79,41%	80,43%	2,82%	8,86E+12	76,00%	74,27%	1,11%	8,20E+12
s2	35	75,00%	75,34%	2,15%	7,31E+12	75,00%	71,49%	2,28%	7,05E+12
s2	40	76,47%	67,96%	7,51%	6,73E+12	75,00%	71,68%	1,25%	6,43E+12
s2	45	86,44%	87,96%	11,56%	5,95E+12	72,00%	72,58%	0,42%	5,00E+12
s2	50	76,32%	76,61%	3,52%	4,69E+12	69,23%	66,21%	1,19%	4,08E+12
s3	5	64,29%	42,42%	0,02%	7,50E+13	75,00%	75,00%	0,41%	7,64E+13
s3	10	78,13%	76,19%	16,05%	3,95E+13	75,86%	75,64%	1,07%	3,04E+13
s3	15	70,97%	62,02%	8,44%	1,83E+13	82,05%	82,00%	12,16%	2,11E+13
s3	20	82,93%	68,18%	1,87%	1,41E+13	78,57%	75,57%	0,97%	1,36E+13
s3	25	86,27%	71,86%	4,48%	1,21E+13	88,37%	88,10%	4,97%	1,20E+13
s3	30	88,73%	87,19%	9,58%	1,10E+13	84,62%	83,03%	2,31%	9,80E+12
s3	35	83,33%	81,46%	6,02%	8,87E+12	66,67%	68,46%	0,81%	8,00E+12
s3	40	85,96%	86,31%	3,01%	7,20E+12	58,33%	54,67%	0,53%	6,80E+12
s3	45	76,19%	75,75%	2,55%	5,93E+12	78,79%	74,41%	6,61%	6,51E+12
s3	50	81,63%	79,75%	9,74%	6,19E+12	68,00%	63,41%	1,42%	4,58E+12
s4	5	73,91%	69,35%	0,72%	6,31E+13	86,84%	74,85%	2,42%	6,47E+13
s4	10	83,67%	84,44%	13,30%	3,29E+13	72,73%	72,58%	1,62%	2,89E+13
s4	15	64,00%	44,14%	0,79%	1,69E+13	81,08%	80,00%	8,43%	1,77E+13
s4	20	82,61%	82,67%	2,37%	1,39E+13	79,41%	77,78%	3,58%	1,27E+13
s4	25	85,00%	85,43%	4,67%	1,13E+13	81,08%	76,03%	4,06%	1,11E+13
s4	30	87,84%	85,69%	6,36%	8,85E+12	88,89%	89,02%	4,93%	8,39E+12
s4	35	80,43%	79,10%	7,50%	7,77E+12	80,00%	78,37%	1,14%	6,54E+12
s4	40	77,14%	78,34%	6,10%	7,16E+12	75,86%	75,56%	11,30%	6,30E+12
s4	45	70,97%	70,03%	5,85%	5,55E+12	69,23%	63,54%	1,94%	4,46E+12

Tabelle A.1 Fortsetzung von vorheriger Seite

		K-means				K-means++			
Datensatz	k	%I	%A	%E	SSE_C	%I2	%A3	%E4	SSE_C
s4	50	76,32%	77,87%	1,71%	5,20E+12	74,07%	69,40%	10,16%	3,97E+12
spiral	5	56,25%	72,73%	0,18%	7,11E+03	73,08%	70,00%	7,97%	7,94E+03
spiral	10	61,90%	53,85%	1,18%	3,21E+03	68,42%	55,56%	0,46%	3,73E+03
spiral	15	68,00%	85,71%	0,76%	1,74E+03	61,90%	50,00%	0,51%	1,54E+03
spiral	20	86,67%	92,31%	0,20%	1,04E+03	86,36%	62,50%	0,87%	9,14E+02
spiral	25	84,44%	89,36%	0,00%	6,31E+02	81,25%	68,63%	0,00%	3,59E+02
spiral	30	74,07%	62,50%	0,96%	4,04E+02	83,87%	75,00%	0,00%	1,71E+02
spiral	35	82,61%	90,00%	0,01%	2,07E+02	88,89%	63,33%	0,01%	1,00E+02
spiral	40	87,23%	88,00%	0,01%	1,56E+02	87,50%	88,44%	0,00%	3,83E+01
spiral	45	84,62%	78,13%	0,00%	1,42E+02	83,87%	75,23%	0,01%	2,55E+01
spiral	50	85,71%	89,71%	0,46%	8,05E+01	94,20%	93,14%	0,04%	1,47E+01
unbalance	5	83,33%	87,10%	0,15%	2,20E+12	71,43%	67,74%	0,09%	1,32E+12
unbalance	10	91,30%	88,18%	0,05%	2,15E+12	88,89%	88,89%	52,74%	4,43E+11
unbalance	15	89,19%	89,18%	0,06%	1,48E+12	86,84%	85,38%	0,69%	1,64E+11
unbalance	20	85,71%	88,17%	0,09%	8,06E+11	89,80%	90,61%	1,88%	1,47E+11
unbalance	25	90,00%	70,83%	0,21%	7,90E+11	82,35%	82,20%	2,44%	9,88E+10
unbalance	30	90,63%	90,30%	0,12%	1,21E+12	82,35%	81,70%	1,02%	8,81E+10
unbalance	35	89,83%	84,14%	0,40%	1,44E+11	85,42%	85,10%	0,94%	7,07E+10
unbalance	40	86,96%	75,89%	0,02%	1,91E+12	84,44%	82,94%	1,83%	5,54E+10
unbalance	45	87,50%	85,70%	0,13%	1,01E+12	86,36%	84,17%	3,90%	3,62E+10
unbalance	50	90,57%	88,70%	14,37%	9,79E+10	81,82%	79,15%	2,65%	3,59E+10
Max		91,30%	92,31%	27,41%	1,05E+14	94,20%	93,14%	52,74%	9,90E+13
Median		82,12%	79,45%	1,79%	5,94E+12	79,10%	75,00%	1,19%	5,23E+12
Mittelwert		79,60%	77,11%	3,60%	1,20E+13	78,10%	74,38%	3,34%	1,10E+13
Min		41,67%	31,03%	0,00%	8,24E+00	42,11%	33,33%	0,00%	2,93E+00

Tabelle A.1.: Ergebnisse der Konvergenz für K-means und K-means++ für die synthetischen Datensätze

A.2. Darstellung des Spiral Datensatzes

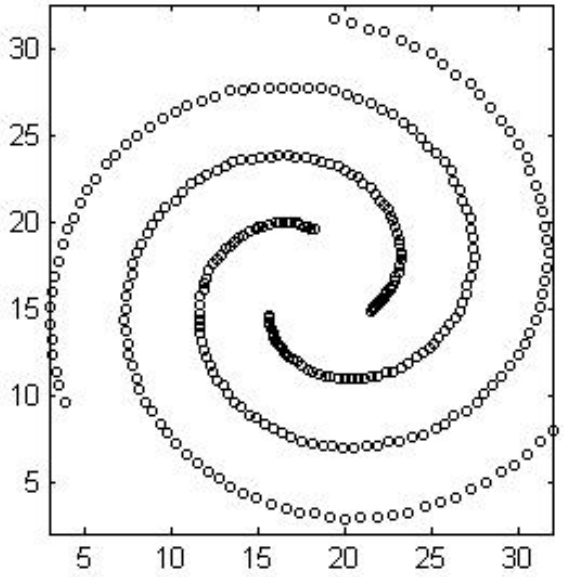


Abbildung A.1.: Darstellung des *Spiral* Datensatzes [CY08]

A.3. Ergebnisse der Visualisierung für synthetische Datensätze

Tabelle A.2.: Visualisierungen für synthetische Datensätze für den K-means und den K-means++

		K-means			K-means++		
Datensatz	#Cluster	#Iterationen	%Vis SE_i	%Vis GP_i	#Iterationen	%Vis SE_i	%Vis $\%GP_i$
a1	5	29	13,79%	17,24%	22	18,18%	22,73%
a1	10	31	6,45%	9,68%	24	12,50%	16,67%
a1	15	32	18,75%	18,75%	28	7,14%	10,71%
a1	20	27	18,52%	14,81%	29	10,34%	10,34%
a1	25	28	21,43%	14,29%	25	8,00%	8,00%
a1	30	25	12,00%	12,00%	26	7,69%	7,69%
a1	35	27	11,11%	11,11%	26	3,85%	7,69%
a1	40	24	12,50%	12,50%	24	4,17%	8,33%
a1	45	25	16,00%	16,00%	23	8,70%	8,70%
a1	50	29	6,90%	10,34%	24	8,33%	12,50%
a2	5	25	8,00%	16,00%	29	6,90%	6,90%
a2	10	31	9,68%	12,90%	29	6,90%	13,79%
a2	15	33	9,09%	9,09%	28	10,71%	14,29%
a2	20	33	15,15%	15,15%	34	5,88%	5,88%
a2	25	34	11,76%	14,71%	27	7,41%	11,11%
a2	30	31	12,90%	12,90%	30	13,33%	13,33%
a2	35	31	9,68%	12,90%	30	13,33%	13,33%
a2	40	27	11,11%	11,11%	29	10,34%	10,34%
a2	45	28	17,86%	14,29%	24	12,50%	12,50%
a2	50	26	15,38%	11,54%	25	12,00%	12,00%
a3	5	22	13,64%	18,18%	26	3,85%	7,69%
a3	10	40	7,50%	12,50%	39	5,13%	10,26%
a3	15	45	11,11%	15,56%	42	9,52%	11,90%
a3	20	32	9,38%	12,50%	41	4,88%	9,76%
a3	25	38	13,16%	15,79%	39	7,69%	10,26%
a3	30	35	8,57%	11,43%	34	11,76%	11,76%
a3	35	35	14,29%	14,29%	29	6,90%	6,90%
a3	40	29	13,79%	13,79%	29	6,90%	10,34%
a3	45	29	13,79%	13,79%	29	10,34%	10,34%
a3	50	29	13,79%	13,79%	31	9,68%	9,68%
Aggregation	5	16	18,75%	18,75%	19	10,53%	15,79%

Tabelle A.2 Fortsetzung von vorheriger Seite

		K-means			K-means++		
Datensatz	#Cluster	#Iterationen	%Vis sse	%Vis Gpi	#Iterationen	%Vis SSE	%Vis Gpi
Aggregation	10	19	15,79%	15,79%	21	9,52%	14,29%
Aggregation	15	24	8,33%	12,50%	21	9,52%	14,29%
Aggregation	20	22	9,09%	13,64%	24	8,33%	12,50%
Aggregation	25	28	17,86%	14,29%	22	9,09%	9,09%
Aggregation	30	27	14,81%	11,11%	28	10,71%	10,71%
Aggregation	35	30	13,33%	10,00%	29	3,45%	3,45%
Aggregation	40	34	8,82%	8,82%	42	2,38%	2,38%
Aggregation	45	41	7,32%	4,88%	31	6,45%	3,23%
Aggregation	50	32	9,38%	6,25%	29	10,34%	6,90%
D31	5	19	15,79%	26,32%	15	13,33%	20,00%
D31	10	18	27,78%	22,22%	14	21,43%	21,43%
D31	15	17	23,53%	17,65%	15	13,33%	6,67%
D31	20	19	15,79%	15,79%	15	13,33%	13,33%
D31	25	22	13,64%	13,64%	18	11,11%	11,11%
D31	30	23	17,39%	17,39%	22	9,09%	9,09%
D31	35	25	20,00%	12,00%	20	5,00%	5,00%
D31	40	23	17,39%	13,04%	24	8,33%	8,33%
D31	45	25	12,00%	8,00%	26	7,69%	7,69%
D31	50	24	16,67%	8,33%	27	7,41%	7,41%
R15	5	11	18,18%	18,18%	11	9,09%	18,18%
R15	10	19	21,05%	15,79%	15	20,00%	20,00%
R15	15	22	9,09%	9,09%	23	8,70%	4,35%
R15	20	22	9,09%	9,09%	24	8,33%	4,17%
R15	25	27	11,11%	11,11%	25	4,00%	4,00%
R15	30	24	16,67%	8,33%	23	8,70%	4,35%
R15	35	42	2,38%	4,76%	29	6,90%	3,45%
R15	40	42	9,52%	7,14%	30	6,67%	3,33%
R15	45	41	7,32%	4,88%	38	7,89%	2,63%
R15	50	42	11,90%	7,14%	33	6,06%	3,03%
s1	5	20	15,00%	20,00%	15	20,00%	20,00%
s1	10	27	22,22%	18,52%	18	11,11%	11,11%
s1	15	25	4,00%	4,00%	20	15,00%	10,00%
s1	20	31	9,68%	9,68%	32	3,13%	6,25%
s1	25	32	9,38%	12,50%	32	3,13%	6,25%
s1	30	30	10,00%	13,33%	30	3,33%	6,67%
s1	35	30	10,00%	13,33%	29	6,90%	10,34%

Tabelle A.2 Fortsetzung von vorheriger Seite

		K-means			K-means++		
Datensatz	#Cluster	#Iterationen	%Vis sse	%Vis Gpi	#Iterationen	%Vis SSE	%Vis Gpi
s1	40	32	9,38%	12,50%	27	3,70%	11,11%
s1	45	31	9,68%	12,90%	26	11,54%	11,54%
s1	50	34	5,88%	11,76%	29	6,90%	6,90%
s2	5	22	9,09%	13,64%	18	22,22%	16,67%
s2	10	23	17,39%	17,39%	25	4,00%	8,00%
s2	15	27	14,81%	11,11%	21	14,29%	14,29%
s2	20	41	9,76%	7,32%	27	7,41%	7,41%
s2	25	33	12,12%	12,12%	32	6,25%	9,38%
s2	30	43	6,98%	9,30%	34	5,88%	8,82%
s2	35	35	5,71%	8,57%	33	6,06%	12,12%
s2	40	35	8,57%	11,43%	36	5,56%	8,33%
s2	45	30	13,33%	13,33%	30	6,67%	10,00%
s2	50	33	9,09%	12,12%	30	6,67%	10,00%
s3	5	27	14,81%	18,52%	32	6,25%	12,50%
s3	10	27	11,11%	14,81%	29	6,90%	10,34%
s3	15	28	14,29%	17,86%	39	7,69%	12,82%
s3	20	35	8,57%	14,29%	37	10,81%	13,51%
s3	25	47	6,38%	10,64%	38	5,26%	10,53%
s3	30	46	6,52%	13,04%	40	5,00%	10,00%
s3	35	37	8,11%	13,51%	34	5,88%	11,76%
s3	40	36	8,33%	11,11%	32	9,38%	9,38%
s3	45	38	10,53%	10,53%	34	8,82%	11,76%
s3	50	31	12,90%	16,13%	33	9,09%	9,09%
s4	5	23	8,70%	17,39%	25	8,00%	16,00%
s4	10	44	6,82%	11,36%	29	17,24%	20,69%
s4	15	43	11,63%	16,28%	32	12,50%	15,63%
s4	20	38	7,89%	10,53%	34	8,82%	11,76%
s4	25	42	7,14%	9,52%	37	8,11%	13,51%
s4	30	44	6,82%	13,64%	49	6,12%	8,16%
s4	35	39	7,69%	12,82%	40	5,00%	10,00%
s4	40	40	7,50%	10,00%	28	10,71%	10,71%
s4	45	35	11,43%	14,29%	28	10,71%	10,71%
s4	50	37	8,11%	10,81%	30	10,00%	10,00%
spiral	5	20	15,00%	25,00%	22	13,64%	22,73%
spiral	10	23	13,04%	13,04%	23	13,04%	13,04%
spiral	15	25	16,00%	16,00%	22	13,64%	13,64%

Tabelle A.2 Fortsetzung von vorheriger Seite

		K-means			K-means++		
Datensatz	#Cluster	#Iterationen	%Vis sse	%Vis Gpi	#Iterationen	%Vis SSE	%Vis Gpi
spiral	20	25	12,00%	8,00%	29	6,90%	6,90%
spiral	25	34	14,71%	8,82%	32	9,38%	6,25%
spiral	30	42	11,90%	7,14%	32	6,25%	3,13%
spiral	35	45	4,44%	4,44%	44	4,55%	2,27%
spiral	40	44	6,82%	4,55%	40	7,50%	2,50%
spiral	45	44	6,82%	4,55%	42	2,38%	2,38%
spiral	50	45	4,44%	2,22%	38	5,26%	2,63%
unbalance	5	29	6,90%	6,90%	15	6,67%	6,67%
unbalance	10	36	2,78%	8,33%	35	2,86%	5,71%
unbalance	15	50	2,00%	12,00%	29	3,45%	10,34%
unbalance	20	58	1,72%	10,34%	42	4,76%	14,29%
unbalance	25	67	2,99%	8,96%	53	1,89%	9,43%
unbalance	30	49	4,08%	14,29%	46	4,35%	10,87%
unbalance	35	47	4,26%	14,89%	50	2,00%	10,00%
unbalance	40	48	8,33%	12,50%	44	6,82%	9,09%
unbalance	45	51	1,96%	9,80%	49	4,08%	12,24%
unbalance	50	40	2,50%	12,50%	37	8,11%	13,51%
chainlink	5	22	9,09%	18,18%	26	7,69%	11,54%
chainlink	10	20	15,00%	20,00%	26	15,38%	15,38%
chainlink	15	32	15,63%	15,63%	24	12,50%	16,67%
chainlink	20	24	12,50%	12,50%	22	9,09%	9,09%
chainlink	25	24	16,67%	16,67%	21	9,52%	9,52%
chainlink	30	25	20,00%	20,00%	22	9,09%	9,09%
chainlink	35	26	19,23%	11,54%	22	9,09%	9,09%
chainlink	40	30	10,00%	10,00%	20	20,00%	15,00%
chainlink	45	27	14,81%	11,11%	38	5,26%	2,63%
chainlink	50	33	18,18%	9,09%	28	3,57%	3,57%
cluto-t4-8k	5	21	14,29%	19,05%	20	20,00%	25,00%
cluto-t4-8k	10	36	5,56%	11,11%	32	9,38%	12,50%
cluto-t4-8k	15	43	9,30%	16,28%	29	10,34%	17,24%
cluto-t4-8k	20	38	10,53%	10,53%	32	9,38%	12,50%
cluto-t4-8k	25	35	14,29%	17,14%	35	5,71%	11,43%
cluto-t4-8k	30	35	11,43%	14,29%	40	10,00%	12,50%
cluto-t4-8k	35	38	7,89%	15,79%	39	7,69%	12,82%
cluto-t4-8k	40	34	8,82%	11,76%	34	8,82%	11,76%
cluto-t4-8k	45	48	6,25%	10,42%	34	5,88%	11,76%

Tabelle A.2 Fortsetzung von vorheriger Seite

		K-means			K-means++		
Datensatz	#Cluster	#Iterationen	%Vis sse	%Vis Gpi	#Iterationen	%Vis SSE	%Vis Gpi
cluto-t4-8k	50	34	14,71%	14,71%	37	5,41%	8,11%
cluto-t5-8k	5	14	14,29%	14,29%	15	6,67%	6,67%
cluto-t5-8k	10	25	12,00%	16,00%	26	3,85%	3,85%
cluto-t5-8k	15	30	3,33%	6,67%	28	3,57%	7,14%
cluto-t5-8k	20	30	13,33%	16,67%	31	6,45%	12,90%
cluto-t5-8k	25	42	4,76%	7,14%	29	10,34%	13,79%
cluto-t5-8k	30	35	11,43%	11,43%	37	5,41%	8,11%
cluto-t5-8k	35	30	10,00%	13,33%	38	7,89%	10,53%
cluto-t5-8k	40	44	4,55%	11,36%	32	6,25%	9,38%
cluto-t5-8k	45	36	8,33%	13,89%	42	4,76%	7,14%
cluto-t5-8k	50	42	9,52%	11,90%	31	9,68%	9,68%
cluto-t7-10k	5	19	26,32%	26,32%	17	17,65%	17,65%
cluto-t7-10k	10	68	5,88%	8,82%	64	3,13%	6,25%
cluto-t7-10k	15	40	7,50%	15,00%	39	5,13%	10,26%
cluto-t7-10k	20	52	5,77%	9,62%	49	4,08%	8,16%
cluto-t7-10k	25	48	6,25%	10,42%	48	6,25%	10,42%
cluto-t7-10k	30	58	5,17%	8,62%	54	7,41%	9,26%
cluto-t7-10k	35	45	6,67%	11,11%	45	6,67%	11,11%
cluto-t7-10k	40	48	12,50%	14,58%	52	5,77%	9,62%
cluto-t7-10k	45	50	8,00%	12,00%	44	6,82%	11,36%
cluto-t7-10k	50	51	9,80%	9,80%	42	9,52%	9,52%
cluto-t8-8k	5	28	10,71%	14,29%	37	10,81%	18,92%
cluto-t8-8k	10	38	13,16%	13,16%	31	6,45%	9,68%
cluto-t8-8k	15	42	11,90%	16,67%	45	6,67%	8,89%
cluto-t8-8k	20	66	4,55%	9,09%	44	4,55%	9,09%
cluto-t8-8k	25	61	6,56%	11,48%	40	5,00%	12,50%
cluto-t8-8k	30	60	8,33%	11,67%	47	6,38%	10,64%
cluto-t8-8k	35	52	7,69%	13,46%	44	6,82%	11,36%
cluto-t8-8k	40	47	6,38%	10,64%	37	5,41%	10,81%
cluto-t8-8k	45	56	7,14%	10,71%	45	6,67%	11,11%
cluto-t8-8k	50	48	8,33%	10,42%	32	9,38%	15,63%
diamond9	5	27	11,11%	14,81%	27	3,70%	11,11%
diamond9	10	17	35,29%	29,41%	31	12,90%	12,90%
diamond9	15	35	8,57%	11,43%	25	8,00%	12,00%
diamond9	20	36	8,33%	13,89%	29	6,90%	13,79%
diamond9	25	35	11,43%	14,29%	32	9,38%	12,50%

Tabelle A.2 Fortsetzung von vorheriger Seite

		K-means			K-means++		
Datensatz	#Cluster	#Iterationen	%Vis sse	%Vis Gpi	#Iterationen	%Vis SSE	%Vis Gpi
diamond9	30	32	9,38%	15,63%	30	6,67%	10,00%
diamond9	35	30	10,00%	16,67%	27	7,41%	11,11%
diamond9	40	38	10,53%	10,53%	29	10,34%	10,34%
diamond9	45	28	10,71%	17,86%	27	11,11%	14,81%
diamond9	50	25	12,00%	16,00%	25	8,00%	12,00%
engytime	5	33	6,06%	15,15%	42	7,14%	19,05%
engytime	10	41	4,88%	14,63%	46	4,35%	15,22%
engytime	15	46	6,52%	15,22%	51	5,88%	13,73%
engytime	20	61	3,28%	11,48%	52	5,77%	13,46%
engytime	25	54	7,41%	16,67%	44	4,55%	11,36%
engytime	30	57	5,26%	10,53%	50	6,00%	12,00%
engytime	35	42	7,14%	14,29%	36	5,56%	13,89%
engytime	40	37	8,11%	13,51%	42	7,14%	11,90%
engytime	45	39	10,26%	12,82%	58	3,45%	6,90%
engytime	50	35	8,57%	11,43%	38	5,26%	10,53%
Min		11	1,72%	2,22%	11	1,89%	2,27%
Max		68	35,29%	29,41%	64	22,22%	25,00%
Median		33	9,78%	12,66%	30	7,41%	10,38%
Mittelwert		34,42631579	10,82%	12,86%	31,7	8,16%	10,60%

Literaturverzeichnis

[ADHP09] D. Aloise, A. Deshpande, P. Hansen, P. Popat. „NP-hardness of Euclidean sum-of-squares clustering“. en. In: *Machine Learning* 75.2 (Mai 2009), S. 245–248. ISSN: 0885-6125, 1573-0565. DOI: 10.1007/s10994-009-5103-0. URL: https://link.springer.com/article/10.1007/s10994-009-5103-0 (zitiert auf S. 19).

[AV07] D. Arthur, S. Vassilvitskii. „k-means++: The advantages of careful seeding“. In: *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial und Applied Mathematics, 2007, S. 1027–1035. URL: http://dl.acm.org/citation.cfm?id=128338 (zitiert auf S. 25, 26).

[Ber+06] P. Berkhin et al. „A survey of clustering data mining techniques.“ In: *Grouping multidimensional data* 25 (2006), S. 71. URL: http://link.springer.com/content/pdf/10.1007/3-540-28349-8.pdf#page=34 (zitiert auf S. 21).

[BP98] J. C. Bezdek, N. R. Pal. „Some new indexes of cluster validity“. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 28.3 (Juni 1998), S. 301–315. ISSN: 1083-4419. DOI: 10.1109/3477.678624 (zitiert auf S. 40).

[CCT10] S.-S. Choi, S.-H. Cha, C. C. Tappert. „A Survey of Binary Similarity and Distance Measures“. In: *Journal of Systemics, Cybernetics and Informatics* 8.1 (2010), S. 43–48. URL: http://pdixon.public.iastate.edu/stat415/Choi.pdf (zitiert auf S. 20).

[Cha16] P. Chamoni. *Data Mining — Enzyklopaedie der Wirtschaftsinformatik*. Nov. 2016. URL: http://www.enzyklopaedie-der-wirtschaftsinformatik.de/lexikon/daten-wissen/Business-Intelligence/Analytische-Informationssysteme--Methoden-der-/Data-Mining/index.html (zitiert auf S. 17).

[CJ85] J. M. Coggins, A. K. Jain. „A spatial filtering approach to texture analysis“. In: *Pattern Recognition Letters* 3.3 (Mai 1985), S. 195–203. ISSN: 0167-8655. DOI: 10.1016/0167-8655(85)90053-4. URL: http://www.sciencedirect.com/science/article/pii/0167865585900534 (zitiert auf S. 42).

[CKV13] M. E. Celebi, H. A. Kingravi, P. A. Vela. „A comparative study of efficient initialization methods for the k-means clustering algorithm“. In: *Expert Systems with Applications* 40.1 (Jan. 2013), S. 200–210. ISSN: 0957-4174. DOI: 10.1016/j.eswa.2012.07.021. URL: http://www.sciencedirect.com/science/article/pii/S0957417412008767 (zitiert auf S. 25, 34).

[CY08] H. Chang, D.-Y. Yeung. „Robust path-based spectral clustering“. In: *Pattern Recognition* 41.1 (Jan. 2008), S. 191–203. ISSN: 0031-3203. DOI: 10.1016/j.patcog.2007.04.010. URL: http://www.sciencedirect.com/science/article/pii/S0031320307002038 (zitiert auf S. 53, 89).

[DB79] D. L. Davies, D. W. Bouldin. „A Cluster Separation Measure“. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-1.2 (Apr. 1979), S. 224–227. ISSN: 0162-8828. DOI: 10.1109/TPAMI.1979.4766909 (zitiert auf S. 41).

[DHJ+07] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, W. Vogels. „Dynamo: Amazon's Highly Available Key-value Store“. In: *Proceedings of Twenty-first ACM SIGOPS Symposium on Operating Systems Principles*. SOSP '07. New York, NY, USA: ACM, 2007, S. 205–220. ISBN: 978-1-59593-591-5. DOI: 10.1145/1294261.1294281. URL: http://doi.acm.org/10.1145/1294261.1294281 (zitiert auf S. 13).

[Dix53] W. J. Dixon. „Processing Data for Outliers“. In: *Biometrics* 9.1 (1953), S. 74–89. ISSN: 0006-341X. DOI: 10.2307/3001634. URL: http://www.jstor.org/stable/3001634 (zitiert auf S. 62).

[Dun74] J. C. Dunn. „Well-Separated Clusters and Optimal Fuzzy Partitions“. In: *Journal of Cybernetics* 4.1 (Jan. 1974), S. 95–104. ISSN: 0022-0280. DOI: 10.1080/01969727408546059. URL: http://dx.doi.org/10.1080/01969727408546059 (zitiert auf S. 40).

[EKS+96] M. Ester, H.-P. Kriegel, J. Sander, X. Xu et al. „A density-based algorithm for discovering clusters in large spatial databases with noise.“ In: *Kdd*. Bd. 96. 1996, S. 226–231. URL: https://ocs.aaai.org/Papers/KDD/1996/KDD96-037.pdf (zitiert auf S. 26).

[Est09] M. Ester. „Density-based Clustering“. en. In: *Encyclopedia of Database Systems*. Hrsg. von L. LIU, M. T. ÖZSU. DOI: 10.1007/978-0-387-39940-9_605. Springer US, 2009, S. 795–799. ISBN: 978-0-387-35544-3 978-0-387-39940-9. URL: http://link.springer.com/referenceworkentry/10.1007/978-0-387-39940-9_605 (zitiert auf S. 22, 31).

[FA12] A. Frank, A. Asuncion. *UCI Machine Learning Repository*. 2012. URL: https://archive.ics.uci.edu/ml/datasets.html?format=&task=cla&att=num&area=&numAtt=&numIns=&type=&sort=instDown&view=table (zitiert auf S. 51).

[Fas] D. Fasel. *Big Data - Grundlagen, Systeme und Nutzungspotenziale*. Springer. ISBN: 978-3-658-11588-3. URL: http://www.springer.com/de/book/9783658115883 (zitiert auf S. 13).

[FAT+14] A. Fahad, N. Alshatri, Z. Tari, A. Alamri, I. Khalil, A. Y. Zomaya, S. Foufou, A. Bouras. „A Survey of Clustering Algorithms for Big Data: Taxonomy and Empirical Analysis“. In: *IEEE Transactions on Emerging Topics in Computing* 2.3 (Sep. 2014), S. 267–279. ISSN: 2168-6750. DOI: 10.1109/TETC.2014.2330519 (zitiert auf S. 13, 21).

[FPS96a] U. Fayyad, G. Piatetsky-Shapiro, P. Smyth. „From Data Mining to Knowledge Discovery in Databases“. In: *AI Magazine* 17.3 (März 1996), S. 37. ISSN: 0738-4602. URL: https://vvvvw.aaai.org/ojs/index.php/aimagazine/article/view/1230 (zitiert auf S. 18).

[FPS96b] U. Fayyad, G. Piatetsky-Shapiro, P. Smyth. „The KDD Process for Extracting Useful Knowledge from Volumes of Data“. In: *Commun. ACM* 39.11 (Nov. 1996), S. 27–34. ISSN: 0001-0782. DOI: 10.1145/240455.240464. URL: http://doi.acm.org/10.1145/240455.240464 (zitiert auf S. 17).

[FV06] P. Fränti, O. Virmajoki. „Iterative shrinking method for clustering problems“. en. In: *Pattern Recognition* 39.5 (Mai 2006), S. 761–775. ISSN: 00313203. DOI: 10.1016/j.patcog.2005.09.012. URL: http://linkinghub.elsevier.com/retrieve/pii/S0031320305003778 (zitiert auf S. 53).

[GMT07] A. Gionis, H. Mannila, P. Tsaparas. „Clustering aggregation“. en. In: *ACM Transactions on Knowledge Discovery from Data* 1.1 (März 2007), 4–es. ISSN: 15564681. DOI: 10.1145/1217299.1217303. URL: http://portal.acm.org/citation.cfm?doid=1217299.1217303 (zitiert auf S. 53).

[GMW07] G. Gan, C. Ma, J. Wu. *Data Clustering: Theory, Algorithms, and Applications (ASA-SIAM Series on Statistics and Applied Probability)*. Philadelphia, PA, USA: Society for Industrial und Applied Mathematics, 2007. ISBN: 978-0-89871-623-8 (zitiert auf S. 19, 33).

[HBV02] M. Halkidi, Y. Batistakis, M. Vazirgiannis. „Cluster Validity Methods: Part I“. In: *SIGMOD Rec.* 31.2 (Juni 2002), S. 40–45. ISSN: 0163-5808. DOI: 10.1145/565117.565124. URL: http://doi.acm.org/10.1145/565117.565124 (zitiert auf S. 40).

[HPK11] J. Han, J. Pei, M. Kamber. *Data mining: concepts and techniques*. Elsevier, 2011 (zitiert auf S. 13, 18, 20, 25).

[Jai10] A. K. Jain. „Data clustering: 50 years beyond K-means“. In: *Pattern Recognition Letters*. Award winning papers from the 19th International Conference on Pattern Recognition (ICPR) 31.8 (Juni 2010), S. 651–666. ISSN: 0167-8655. DOI: 10.1016/j.patrec.2009.09.011. URL: http://www.sciencedirect.com/science/article/pii/S0167865509002323 (zitiert auf S. 34).

[JD88] A. K. Jain, R. C. Dubes. *Algorithms for Clustering Data*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1988. ISBN: 978-0-13-022278-7 (zitiert auf S. 21, 32).

[JMF99] A. K. Jain, M. N. Murty, P. J. Flynn. „Data Clustering: A Review“. In: *ACM Comput. Surv.* 31.3 (Sep. 1999), S. 264–323. ISSN: 0360-0300. DOI: 10.1145/331499.331504. URL: http://doi.acm.org/10.1145/331499.331504 (zitiert auf S. 13, 18, 22).

[Kar02] G. Karypis. *CLUTO-a clustering toolkit*. Techn. Ber. MINNESOTA UNIV MINNEAPOLIS DEPT OF COMPUTER SCIENCE, 2002. URL: http://www.dtic.mil/docs/citations/ADA439508 (zitiert auf S. 53).

[KF02] I. Kärkkäinen, P. Fränti. *Dynamic local search algorithm for the clustering problem*. English. OCLC: 58380784. Joensuu: University of Joensuu, 2002. ISBN: 978-952-458-143-1 (zitiert auf S. 53).

[KHK99] G. Karypis, E.-H. Han, V. Kumar. „Chameleon: hierarchical clustering using dynamic modeling“. In: *Computer* 32.8 (Aug. 1999), S. 68–75. ISSN: 0018-9162. DOI: 10.1109/2.781637 (zitiert auf S. 53).

[Llo82] S. Lloyd. „Least squares quantization in PCM“. In: *IEEE Transactions on Information Theory* 28.2 (März 1982), S. 129–137. ISSN: 0018-9448. DOI: 10.1109/TIT.1982.1056489 (zitiert auf S. 25).

[Mac67] J. MacQueen. „Some methods for classification and analysis of multivariate observations“. In: *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*. Bd. 1. Oakland, CA, USA., 1967, S. 281–297 (zitiert auf S. 22, 25).

[MJ96] J. Mao, A. K. Jain. „A self-organizing network for hyperellipsoidal clustering (HEC)“. In: *IEEE Transactions on Neural Networks* 7.1 (Jan. 1996), S. 16–29. ISSN: 1045-9227. DOI: 10.1109/72.478389 (zitiert auf S. 20).

[MRC+16] A. Mexicano, R. Rodríguez, S. Cervantes, P. Montes, M. Jiménez, N. Almanza, A. Abrego. „The early stop heuristic: A new convergence criterion for K-means“. In: *AIP Conference Proceedings* 1738.1 (Juni 2016), S. 310003. ISSN: 0094-243X. DOI: 10.1063/1.4952103. URL: http://aip.scitation.org/doi/abs/10.1063/1.4952103 (zitiert auf S. 34, 35).

[Phi02] S. J. Phillips. „Acceleration of K-Means and Related Clustering Algorithms“. en. In: *Algorithm Engineering and Experiments*. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, Jan. 2002, S. 166–177. ISBN: 978-3-540-43977-6 978-3-540-45643-8. DOI: 10.1007/3-540-45643-0_13. URL: https://link.springer.com/chapter/10.1007/3-540-45643-0_13 (zitiert auf S. 34).

[Pie13] C. Piech. *Stanford Artificial Intelligence Course*. 2013. URL: http://stanford.edu/~cpiech/cs221/handouts/kmeans.html (zitiert auf S. 23).

[PM99] D. Pelleg, A. Moore. „Accelerating Exact K-means Algorithms with Geometric Reasoning“. In: *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '99. New York, NY, USA: ACM, 1999, S. 277–281. ISBN: 978-1-58113-143-7. DOI: 10.1145/312129.312248. URL: http://doi.acm.org/10.1145/312129.312248 (zitiert auf S. 34).

[RF16] M. Rezaei, P. Fränti. „Set matching measures for external cluster validity“. In: *IEEE Transactions on Knowledge and Data Engineering* 28.8 (2016), S. 2173–2186. URL: http://ieeexplore.ieee.org/abstract/document/7448949/ (zitiert auf S. 53).

[RGR17] D. Reinsel, J. Gantz, J. Rydning. *Data Age 2025: The Evolution of Data to Life-Critical*. Apr. 2017. URL: http://www.seagate.com/files/www-content/our-story/trends/files/Seagate-WP-DataAge2025-March-2017.pdf (zitiert auf S. 13).

[Rou87] P. J. Rousseeuw. „Silhouettes: A graphical aid to the interpretation and validation of cluster analysis“. In: *Journal of Computational and Applied Mathematics* 20.Supplement C (Nov. 1987), S. 53–65. ISSN: 0377-0427. DOI: 10.1016/0377-0427(87)90125-7. URL: http://www.sciencedirect.com/science/article/pii/0377042787901257 (zitiert auf S. 40).

[SC04] S. Salvador, P. Chan. „Determining the number of clusters/segments in hierarchical clustering/segmentation algorithms“. In: *Tools with Artificial Intelligence, 2004. ICTAI 2004. 16th IEEE International Conference on*. IEEE, 2004, S. 576–584. URL: http://ieeexplore.ieee.org/abstract/document/1374239/ (zitiert auf S. 53).

[STK+03] M. Steinbach, P.-N. Tan, V. Kumar, S. Klooster, C. Potter. „Discovery of climate indices using clustering“. In: *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2003, S. 446–455. URL: http://dl.acm.org/citation.cfm?id=956801 (zitiert auf S. 34).

[Ult05] A. Ultsch. „CLUSTERING WIH SOM: U* C“. In: *Proc. Workshop on Self-Organizing Maps* (Jan. 2005) (zitiert auf S. 53).

[VCH10] L. Vendramin, R. J. G. B. Campello, E. R. Hruschka. „Relative clustering validity criteria: A comparative overview“. en. In: *Statistical Analysis and Data Mining* 3.4 (Aug. 2010), S. 209–235. ISSN: 1932-1872. DOI: 10.1002/sam.10080. URL: http://onlinelibrary.wiley.com/doi/10.1002/sam.10080/abstract (zitiert auf S. 40–42).

[VRB02] C. J. Veenman, M. J. T. Reinders, E. Backer. „A maximum variance cluster algorithm“. In: *IEEE Transactions on pattern analysis and machine intelligence* 24.9 (2002), S. 1273–1280. URL: http://ieeexplore.ieee.org/abstract/document/1033218/ (zitiert auf S. 53).

[WZWD14] X. Wu, X. Zhu, G. Q. Wu, W. Ding. „Data mining with big data“. In: *IEEE Transactions on Knowledge and Data Engineering* 26.1 (Jan. 2014), S. 97–107. ISSN: 1041-4347. DOI: 10.1109/TKDE.2013.109 (zitiert auf S. 13).

Alle URLs wurden zuletzt am 16. 10. 2017 geprüft.

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Ort, Datum, Unterschrift