CLARANS

Semesterprojekt im Fach Wissensexktraktion / Data Mining, Hochschule Wismar, Studiengang Multimedia Engineering, Sommersemester 2013

Daniel Schmidt
Mohamed Ibrahim
Sven Lautenschläger

Inhaltsverzeichnis

1. Beschreibung des Algorithmus

1.1 Allgemeines

CLARANS (**C**lustering **L**arge **A**pplications based on **RAN**domized **S**earch) ist ein Verfahren zur Segmentierung, bzw. Clusterung von großen Datenbeständen (Punkt- und polygonale Objekte), anhand von vorhandenen Ähnlichkeitsstrukturen. Dabei erfolgt die Unterteilung des Merkmalsraumes in eine vorgegebene Anzahl von Clustern (partitionierend), gefolgt von einer iterativen Verbesserung der initialisierten Zuordnung. Das Zentrum eines Clusters bildet ein vorhandener Datensatz. Die Zuordnung der restlichen Daten zu ihren Clustern bestimmt der geringste Abstand zum jeweiligen Zentrum eines Clusters.[1] Diese Einteilung wird *medoidbasiert* genannt. Als Distanzmaß (Metrik) wird in diesem Fall der *euklidische Abstand* verwendet. Möglich sind aber auch weitere, wie die Manhatten- oder Hammingdistanz. Das Ergebnis hat große Ähnlichkeit zu *k-Means*, da in beiden Verfahren versucht wird die Gesamtdistanz der Objekte zu ihrem Clusterzentrum zu minimieren.

Für das Verfahren werden folgende Parameter benötigt:

- *k* (Anzahl der gewünschten Cluster)
- *numlocal* (max. Anz. der Versuche zum Finden lokaler Minima)
- *maxNeighbours* (Anz. max. nicht durchgeführter Tausche aufeinander)
- die zu clusternden *Daten*

CLARANS benötigt einen geringen Speicheraufwand. Unter der Annahme dass jedes Objekt durch zwei 8 Byte lange Werte dargestellt wird, würde eine Clusterung von 1.000.000 Objekten ca. 16 MByte Speicher benötigen. Diese Größe wird heutzutage auch von Heimcomputern problemlos bereitgestellt und ermöglicht die vollständige Abarbeitung im Arbeitsspeicher des PC's.[2] Weiterhin ist es durch die Minimierung des Abstandes zum Clusterschwerpunkt lediglich möglich konvexe Cluster zu bilden. Die Gesamtlaufzeit verhält sich im Mittel Quadratisch zur Anzahl der Objekte $O(n^2)$. Die innere Schleife verhält sich linear zur Anzahl der Objekte $O(n)$. Die prinzipielle Funktionsweise wird in folgendem Bild dargestellt:

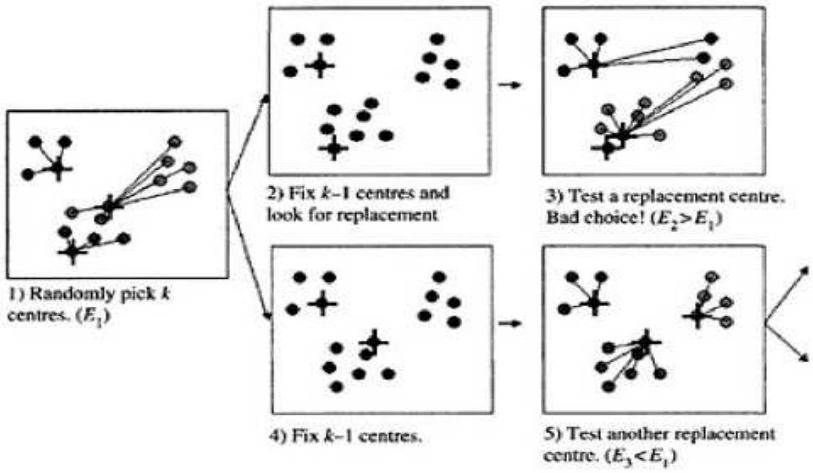

Abbildung 1: prinzipielle Funktionsweise CLARANS[3]

[1] vgl. Hofmann, 2008, S. 9ff.
[2] vgl. Ng, Han, 2002, S. 1004
[3] vgl. Miller, Han, 2001, S. 207

1.2 Ablaufdiagramm

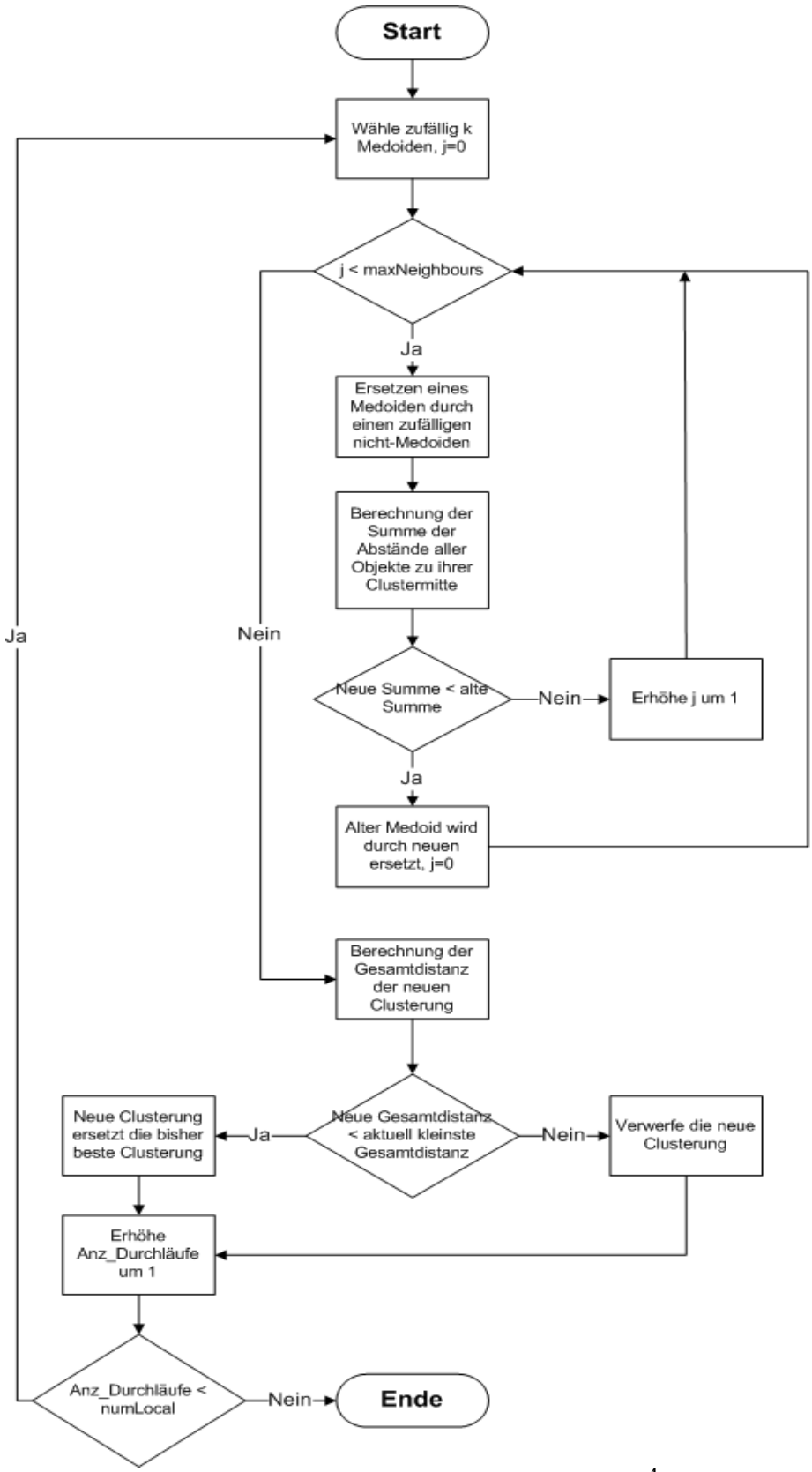

Abbildung 2: Ablaufdiagramm CLARANS[4]

[4] eigene Darstellung

Der Algorithmus besteht aus zwei ineinander verschachtelten Schleifen.

Die innere Schleife beginnt nach der zufälligen Auswahl von *k* (Anzahl der gewünschten Cluster) Medoiden. Anschließend werden alle Objekte den Clustern zugewiesen, zu dessen Medoid sie am nächsten gelegen sind, gefolgt von der Berechnung der Summe selbiger Abstände pro Cluster. Diese wird im ersten Durchlauf als lokales Minimum abgespeichert.

Im nächsten Durchlauf wird ein Medoid zufällig durch einen nicht-Medoiden ersetzt und sowohl die Zuweisung als auch die Berechnung der Abstandssumme erfolgt von vorn. Nun wird überprüft ob die neue Summe kleiner als das bisherige lokale Minimum ist. Ist das nicht der Fall, wird diese Clusterung verworfen und eine Zählvariable (die anzeigt wie viele Durchläufe am Stück keine kleinere Summe gefunden wurde) erhöht. Fällt das Ergebnis positiv aus, wird der alte Medoid durch den neuen ersetzt, die Zählvariable auf den Wert 0 zurückgesetzt und ein neuer Durchlauf gestartet. Dies geschieht solange bis die Zählvariable gleich dem übergebenen Wert *maxNeighbours* ist, also eine festgesetzte Anzahl von Durchläufen ohne endgültigen Austausch eines Medoiden durchläuft.

Ist das der Fall wird in die äußere Schleife gewechselt. Hier wird die neue Clusterung mit der alten Verglichen und bei einer geringeren Distanz als aktuelles globales Minimum gespeichert. Ist die Distanz größer als die alte wird die Clusterung verworfen. Anschließend beginnt der Algorithmus von vorne, solange bis die äußere Schleife eine fest vorgegebene Anzahl an Durchläufen (Parameter *numlocal*) abgearbeitet hat.

2. Experimente

2.1 Implementierung und Vorauswahl der Testdaten

Auf Grund des Umstandes das CLARANS als Funktion im Programm *knime* nicht zur Verfügung steht, wurde es in der Programmiersprache Python selbst verfasst und als aufrufbares Konsolenprogramm unter *Linux* implementiert.
Für die vorläufigen Testdaten wurde der Datensatz *Schwertlilien* gewählt, da er über eine überschaubare Anzahl von Objekten verfügt und bereits vorverarbeitet ist. Für Vergleichszwecke wurde eine zusätzliche Clusterung mit *k-Means* durchgeführt. Der Algorithmus für *k-Means* ist innerhalb einer Bibliothek (scikit-learn) in Python abrufbar. Im Folgenden ist der Quelltext für den CLARANS-Algorithmus abgebildet.

```
def clarans(X, k, maxneighbors, numlocal):
    min_med, min_assign, min_dist, min_cost = None, None, None, None

    for i in range(numlocal):
        '''k zufaellige Punkte als Medoiden'''
        medoids = np.random.permutation(np.arange(X.shape[0]))[0:k]
        '''Abstaende und Zuweisungen berechnen'''
        distances, cost, assign = assign(X,medoids,dist=None,change=None,k)
        j, p = 0, 0
        while j < maxneighbors:
            new_med = medoids.copy()
            '''Einen zufaelligen Medoiden durch einen anderen nicht Medoiden austauschen'''
            new_med[random.randint(0,k-1)] = random.choice(list(set(range(X.shape[0]))-set(medoids)))

            '''Neue Zuordnung durchfuehren'''
            new_dist, new_cost, new_assign = assign(X, new_med, distances, newneighbor,k)
            j += 1

            '''Wenn Gesamtkosten kleiner als aktuelles lokales Min., als neues lokales Min. setzen'''
            if new_cost < cost:
                medoids, assign, distances, cost = new_med, new_assign, new_dist, new_cost
                j = 0

        '''Wenn Gesamtkosten des lokalen Min. kleiner als akt. globales Min. sind, als neues globales Min. setzen'''
        if min_cost is None or cost < min_cost:
            min_med, min_assign, min_dist, min_cost = medoids.copy(), assign, distances, cost

def assign(X, medoids, dist, change, k):
    '''Zuordnung der Punkte zu den gegebenen Medoiden'''
    dist = np.zeros((X.shape[0], k))
    for i in range(k):
        dist[:, i] = np.sum((X-X[medoids[i],:])**2, axis=1)

    cost = np.amin(dist, axis=1).sum()
    assign = np.argmin(dist, axis=1)
    return dist, cost, assign
```

Abbildung 3: Quelltext für CLARANS in Python[5]

[5] eigene Darstellung

2.2 Ergebnisse der Testdaten

Beim Vergleich der Ergebnisclusterungen mit CLARANS und k-Means fällt die erwartete überwiegende Übereinstimmung auf. Sie unterscheiden sich lediglich an den Grenzen der beiden zusammenliegenden Cluster. Beide Datensätze wurden vorher, für eine bessere Visualisierung, mit PCA (Principal Component Analysis, deutsch: Hauptkomponentenanalyse) in den Dimensionen des Parameterraumes reduziert. Dabei wird die Reihenfolge der Koordinatenachsen (die Hauptkomponente) so sortiert, dass die erste Hauptkomponente den größten Anteil der Gesamtstreuung im Datensatz enthält, die zweite Hauptkomponente den zweitgrößten Anteil.

Die Grundannahme (Arbeitshypothese) für die Verwendung der PCA zur Clusteranalyse und Dimensionsreduktion lautet: Die Richtung mit der größten Streuung (Varianz) beinhaltet die meiste Information.[6] Es gibt jedoch auch Ausnahmen, bzw. Anwendungsbeispiele in denen dies nicht zutrifft.

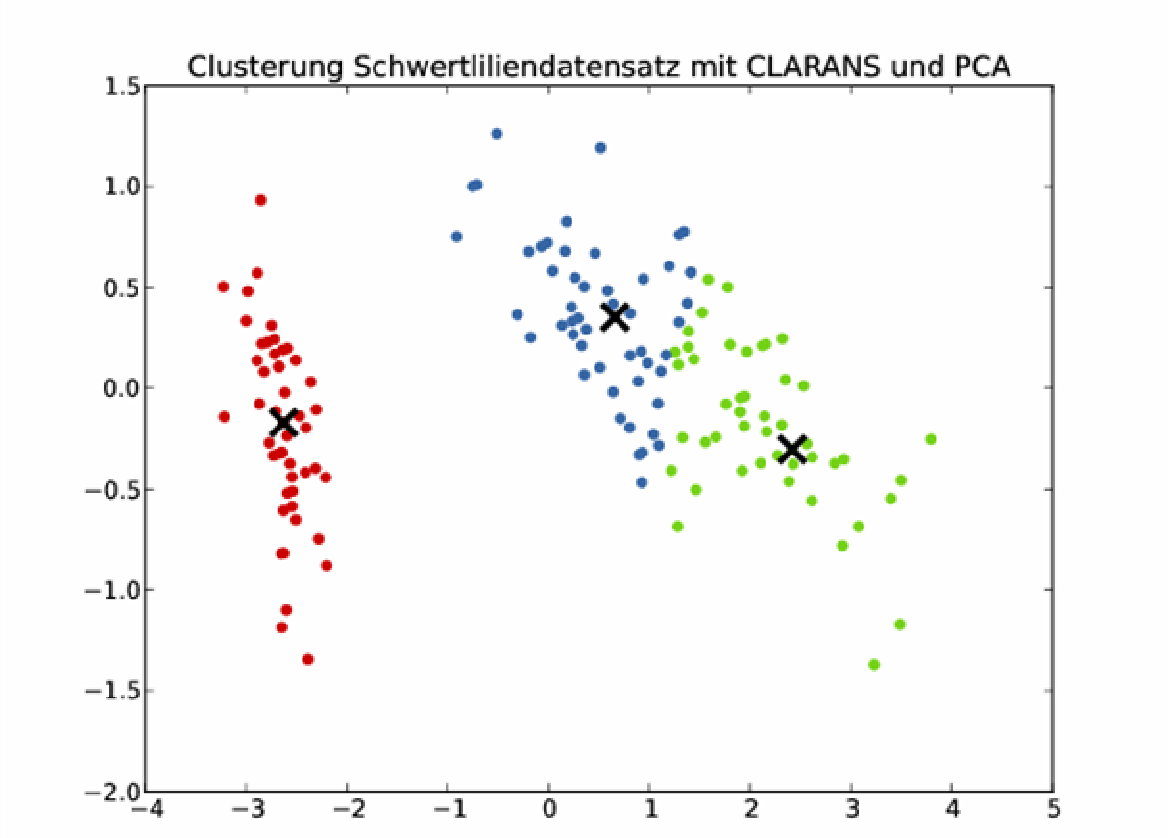

Abbildung 4: Ergebnis der Testdaten mit CLARANS[7]

[6] Wikipedia: Hauptkomponentenanalyse
[7] eigene Darstellung

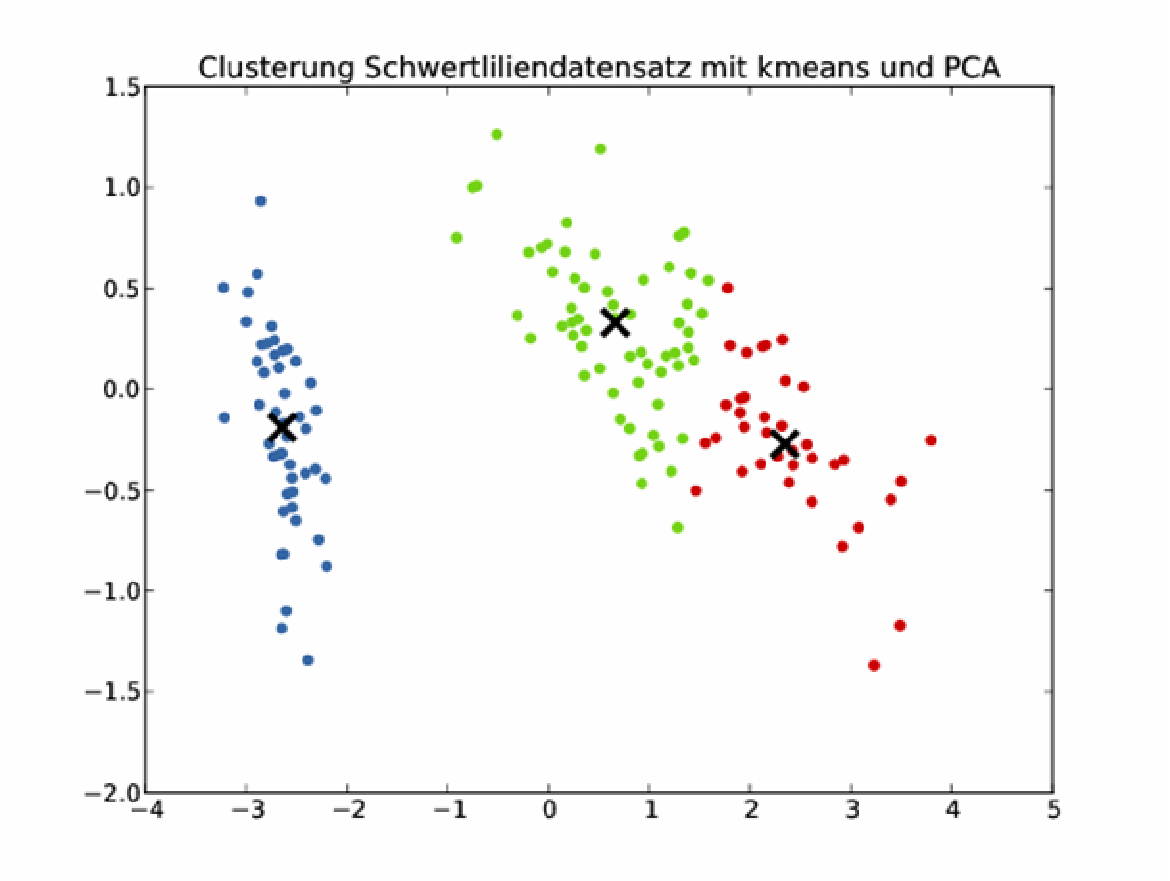

Abbildung 5: Ergebnis der Testdaten mit k-Means[8]

[8] eigene Darstellung

2.3 Clusterung eines großen Datensatzes

Als Beispiel für einen größeren Datensatz wurde das Exemplar des Energieversorgers gewählt. Dabei wurden innerhalb der Vorverarbeitung alle unvollständigen Datensätze gelöscht und die einzelnen Spalten wurden auf Werte zwischen 0 und 10 normiert. Abschließend blieben so noch etwa 7600 Datensätze übrig.

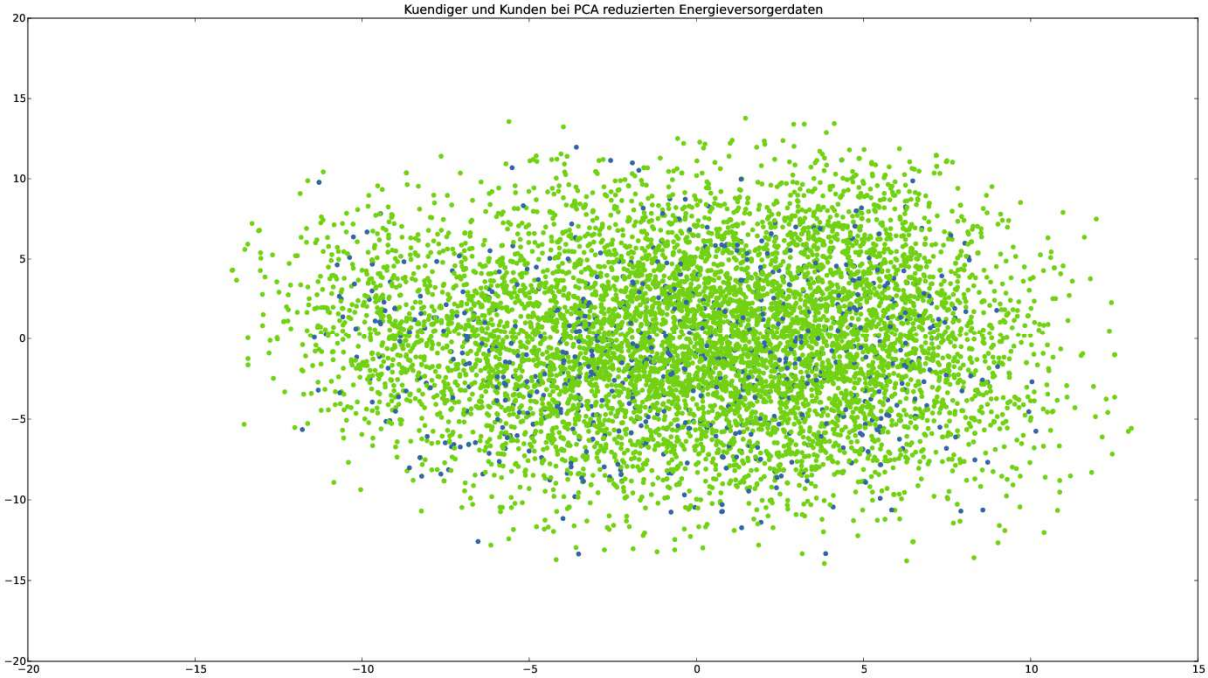

Abbildung 6: Übersicht über Kunden (grün) und Kündiger (blau)[9]

Die Daten wurden mittels PCA in ihren Dimensionen reduziert. In Abbildung 6 wird keine eindeutige Trennung zwischen Kunden und Kündigern offensichtlich. Anschließend folgt der Vergleich der Clusterungen mit k-Means (Abb. 7) und CLARANS (Abb. 8). Die Clusterungen unterscheiden sich im Ergebnis, jedoch ist aus keiner der Beiden eine strikte Trennung der Datensätze erkennbar. Mögliche Ansätze zur Verbesserung des Ergebnisses wären bspw. eine Reduzierung der ausgewählten Dimensionen (Spalten) der Daten, bzw. eine ungleichmäßige Gewichtung der normierten Spalten.

[9] eigene Darstellung

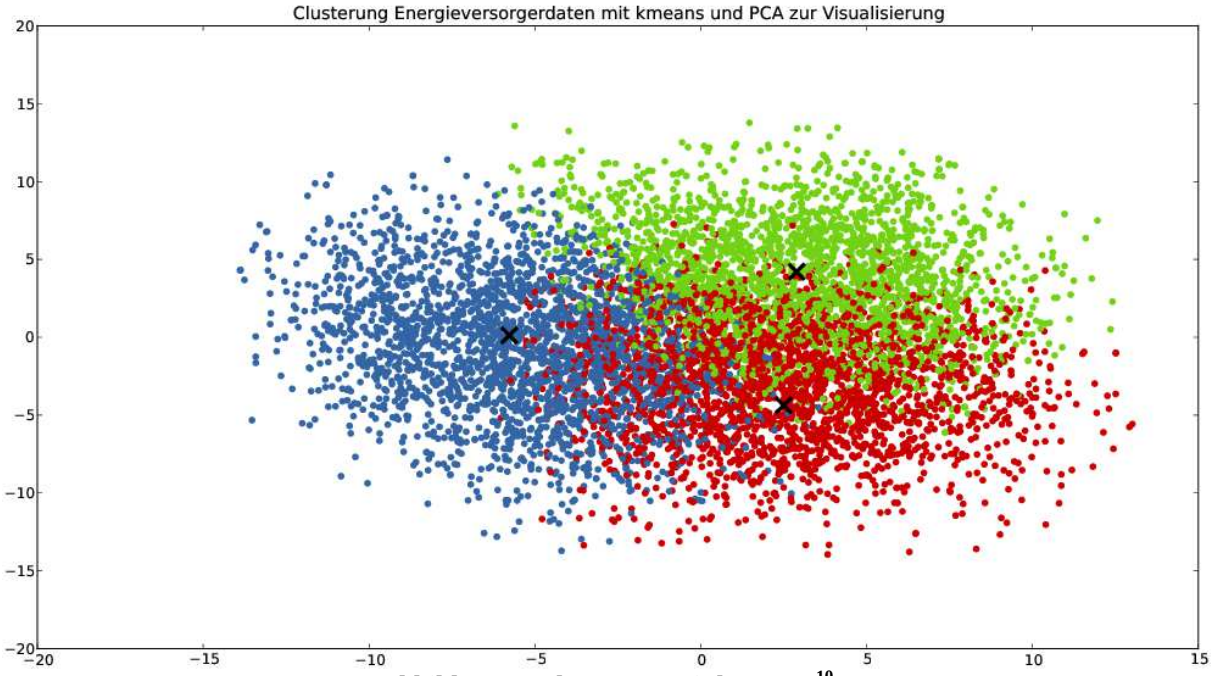

Abbildung 7: Clusterung mit k-Means[10]

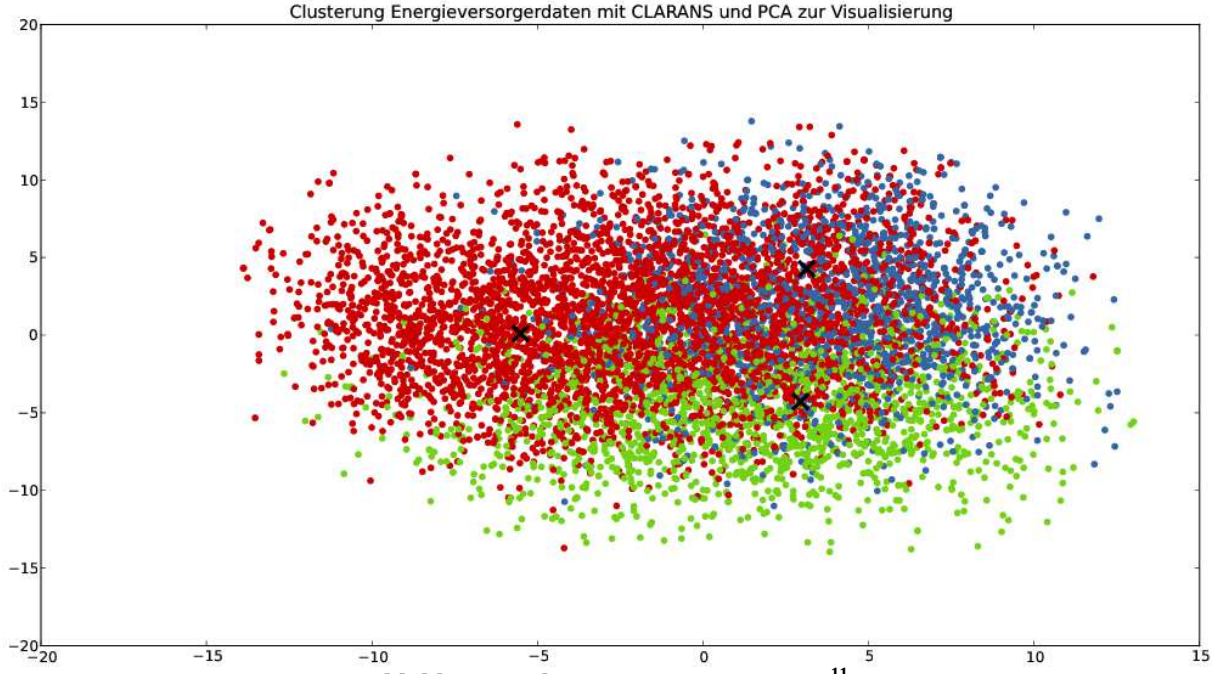

Abbildung 8: Clusterung mit CLARANS[11]

[10] eigene Darstellung
[11] eigene Darstellung

2.4 Optische Vergleiche verschiedener Clusterungsverfahren

Um CLARANS mit anderen Clusteralgorithmen vergleichen zu können, diente ein Diagramm[12] als Vorlage, welches um die Clusterung mit CLARANS erweitert wurde. Hierbei ist zu beachten, dass die Laufzeiten für CLARANS unter Umständen noch nicht optimiert sind. Gut erkennbar sind jedoch die verschiedenen Formen der Cluster.

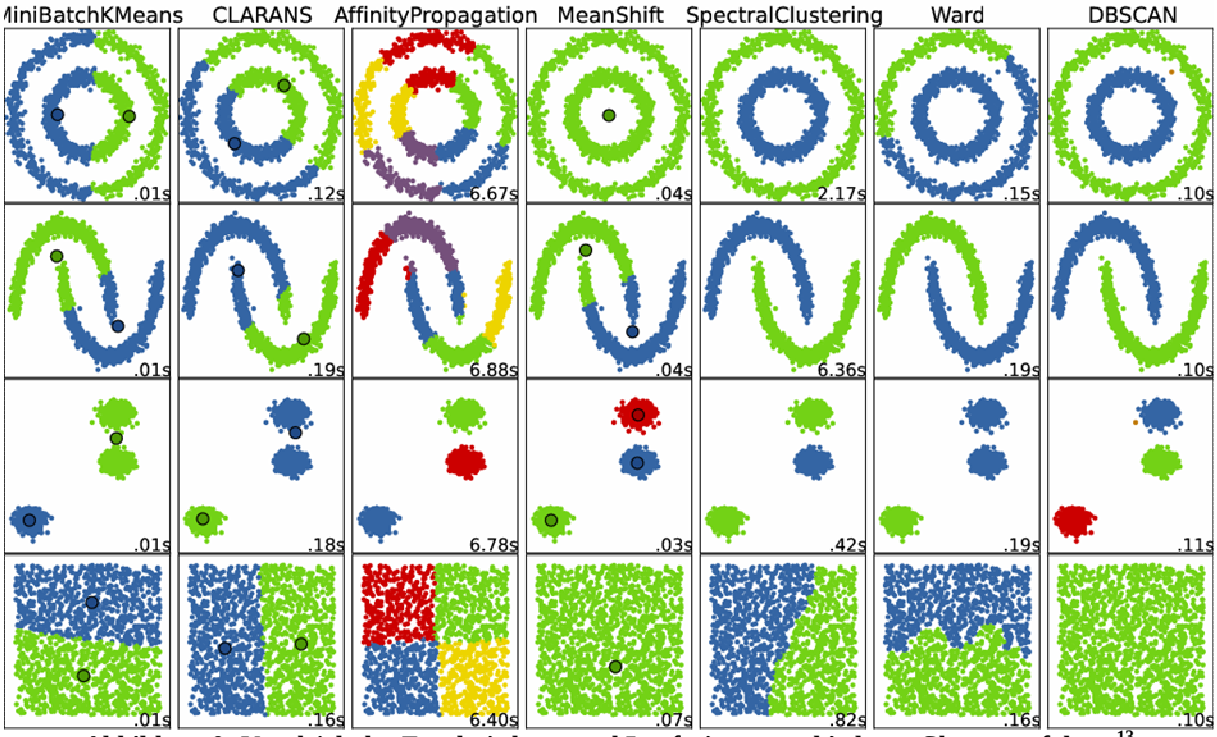

Abbildung 9: Vergleich der Ergebnisdaten und Laufzeiten verschiedener Clusterverfahren[13]

[12] Scikit-learn: Comparing different clustering algorithms on toy datasets
[13] eigene Darstellung

3. Auswertung

CLARANS basiert auf den Verfahren *PAM* & *CLARA* und ist darauf ausgelegt beide Verfahren durch ein besseres Laufzeitverhalten zu ersetzen. Gleiches gilt für das Verfahren *k-Means*, unter der Bedingung das die Menge der auszuwertenden Daten sehr groß ist.
PAM steht für **P**artitioning **A**round **M**edoids und ist die am weitesten verbreitete Realisierung des *k-Medoid* Algorithmus.[14] „Compared to the k-means approach, the function *pam* […] is more robust because it minimizes a sum of dissimilarities instead of a sum of squared euclidean distances.“[15] Für große Datensätze benötigt PAM nach heutigen Standards zu viel Speicherplatz, bzw. zu viel Rechenzeit (beide belaufen sich auf $O(n^2)$). In diesem Fall wird *CLARA* bevorzugt. CLARA steht für **C**lustering **LAR**ge **A**pplications.

“Instead of taking the whole data set into consideration, *CLARA* uses a random sample of the data set. The *PAM* algorithm is then applied to compute the best medoids from the sample. Ideally, the sample should closely represent the original data set. In many cases, a large sample works well if it is created so that each object has equal probability of being selected into the sample. The representative objects (medoids) chosen will likely be similar to those that would have been chosen from the whole data set. *CLARA* builds clusterings from multiple random samples and returns the best clustering as the output. The complexity of computing the medoids on a random sample is $O(ks\,2 + k\,(n - k))$, where s is the size of the sample, k is the number of clusters, and n is the total number of objects.[…]
The effectiveness of *CLARA* depends on the sample size. Notice that *PAM* searches for the best *k-medoids* among a given data set, whereas *CLARA* searches for the best *k-medoids* among the selected sample of the data set. *CLARA* cannot find a good clustering if any of the best sampled medoids is far from the best *k-medoids*.
If an object is one of the best *k-medoids* but is not selected during sampling, *CLARA* will never find the best clustering”[16]

Ein möglicher Optimierungsansatz für CLARANS: Anstatt bei jedem Durchlauf alle Abstände der Punkte zu den Medoiden zu berechnen, müssen nur die Abstände zu dem geänderten Medoiden berechnet werden.
Diese können dann mit den bereits vorher berechneten Abständen der anderen Medoiden verglichen werden. Dies reduziert die Laufzeit auf 1/k.

[14] vgl. Theodoridis, Koutroumbas, 2006, S. 635
[15] Eidgenössische Technische Hochschule Zürich: PAM
[16] vgl. Han, Kamber, Pei, 2012, S. 456

4. Visualisierung eines Clusterungsdurchlaufs

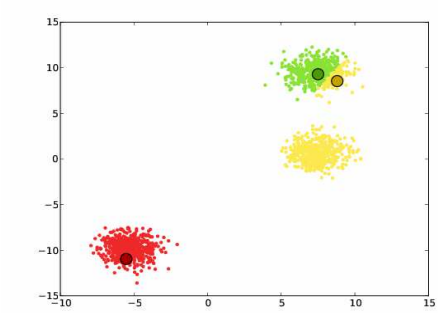

Abbildung 10: Initialisierung, k=3

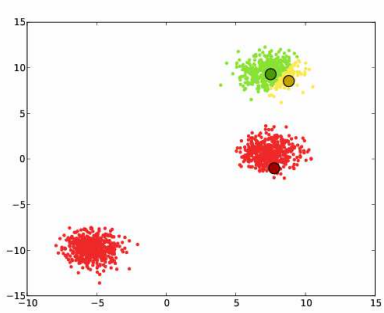

Abbildung 11: 1. Durchlauf, schlechter als Abb. 10, j=1

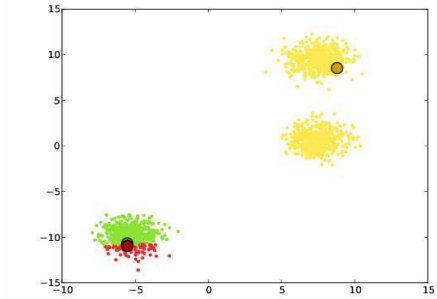

Abbildung 12: 2. Durchlauf, schlechter als Abb. 10, j=2

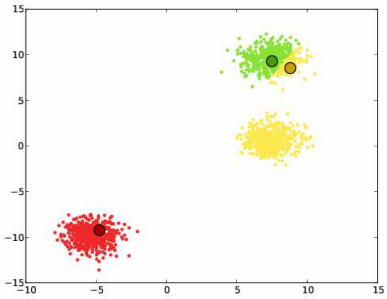

Abbildung 13: 3. Durchlauf, besser als Abb. 10, neues Minimum, j=0

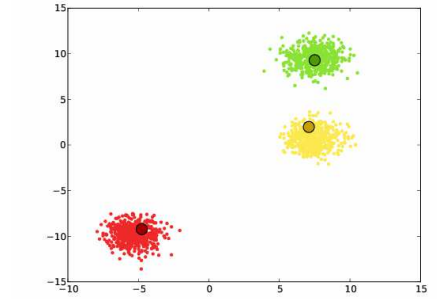

Abbildung 14: 4. Durchlauf, besser als Abb. 13, neues Minimum, j=0

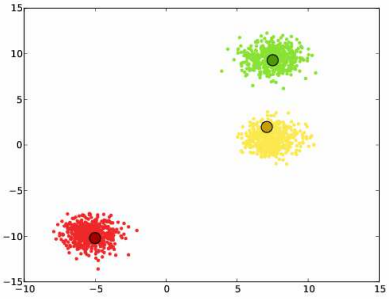

Abbildung 15: 5. Durchlauf, besser als Abb. 14, neues Minimum, j=0

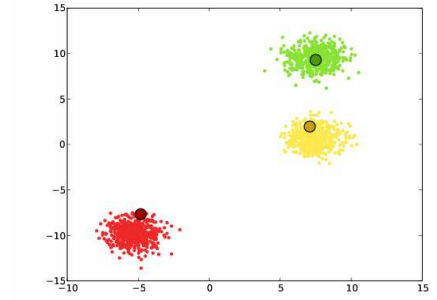

Abbildung 16: 6. Durchlauf, schlechter als Abb. 15, j=1

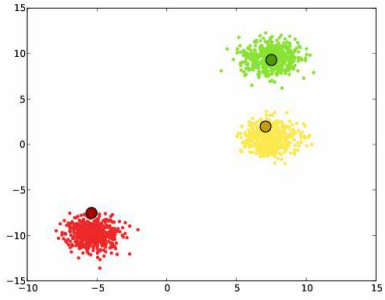

Abbildung 17: 7. Durchlauf, schlechter als Abb. 15, j=2

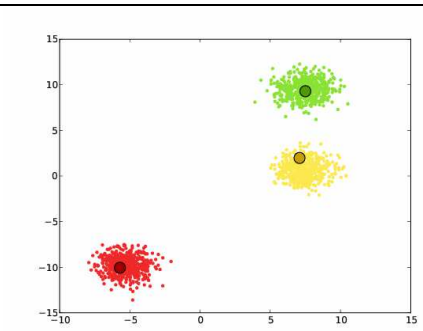

Abbildung 18: 8. Durchlauf, besser als Abb. 15, neues Minimum, j=0

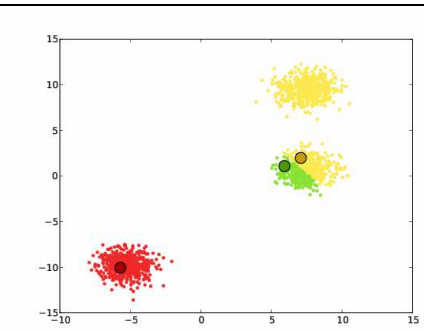

Abbildung 19: 9. Durchlauf, schlechter als Abb. 18, j=1

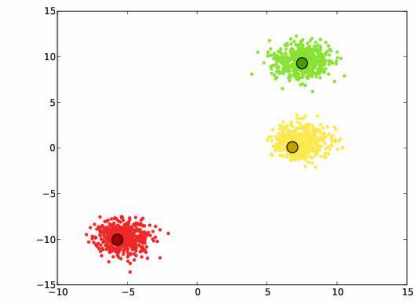

Abbildung 20: 10.Durchlauf, besser als Abb. 18, neues Minimum, j=0

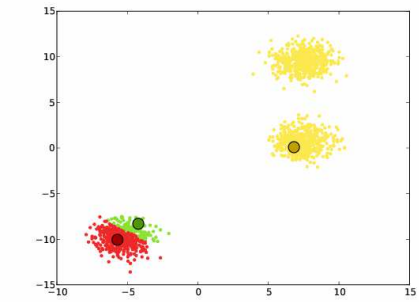

Abbildung 21: 11.Durchlauf, schlechter als Abb. 20, j=1

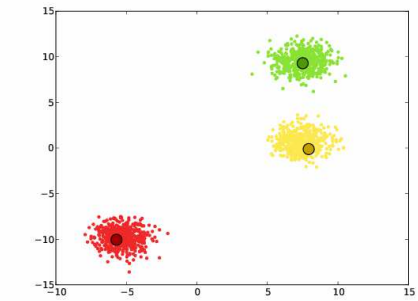

Abbildung 22: 12.Durchlauf, schlechter als Abb. 20, j=2

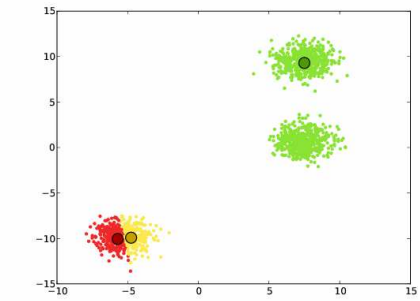

Abbildung 23: 13.Durchlauf, schlechter als Abb. 20, j=3

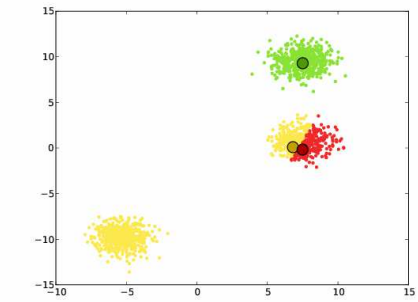

Abbildung 24: 14.Durchlauf, schlechter als Abb. 20, j=4

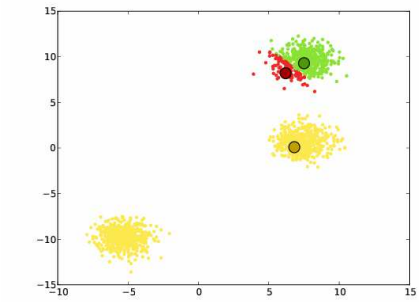

Abbildung 25: 15.Durchlauf, schlechter als Abb. 20, j=5, Ende

Beispiel für den einmaligen Durchlauf der äußeren Schleife mit *k*=3 (gewünschte Anzahl der Cluster) und *maxNeighbours*=5 (Abbruchbedingung der inneren Schleife). (Abb. 10 – 25: eigene Darstellung)

5. Innerer Schleifendurchlauf in Graphendarstellung

Beispielhafte Darstellung eines Durchlaufs der inneren Schleife. Die Anzahl der Cluster *k* wird auf 2 gesetzt, der Parameter *maxNeighbours* auf den Wert 3. Zur besseren Visualisierung wurden die *Datensätze* auf 6 Stück beschränkt, da die Anzahl der Knoten sich aus k*(n-k) berechnet. Die verbundenen Knoten stellen mögliche Nachbarn dar. Auf der linken Seite ist die Auswahl eines Medoiden mit der jeweiligen Summe der Abstände aller Clusterelemente zu ihrem Clustermittelpunkt (Kosten) in dunklem Blau dargestellt. Der Vergleichsmedoid ist in Grün dargestellt. Rechts ist die dazugehörige Clusterung zu sehen. (Abb. 26 – 39: eigene Darstellung)

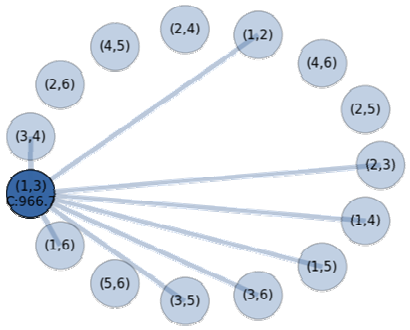

Abbildung 26: Initialisierung, zufällige Deklaration zweier Medoide, j=1

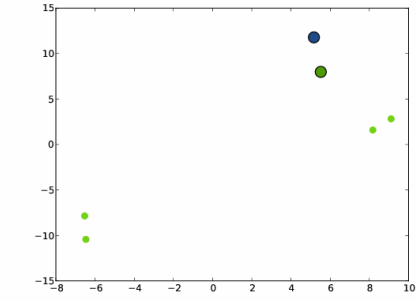

Abbildung 27: aktuelle Clusterung

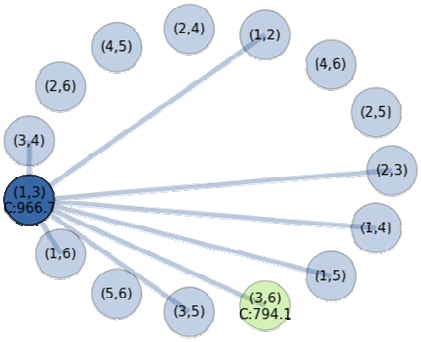

Abbildung 28: Nachbarknoten mit geringeren Kosten, neuer Medoid, j=1

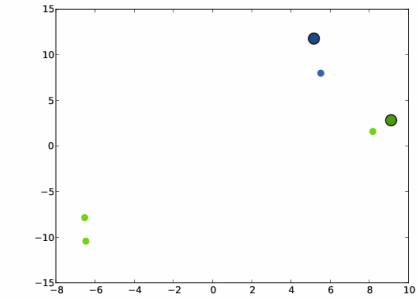

Abbildung 29: aktuelle Clusterung

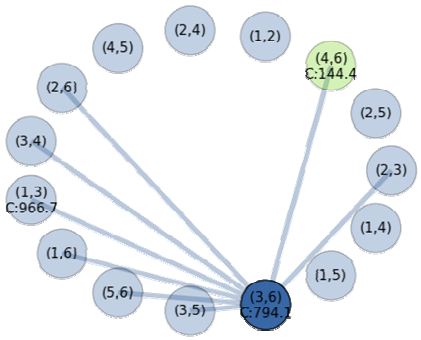

Abbildung 30: Nachbarknoten mit geringeren Kosten, neuer Medoid, j=1

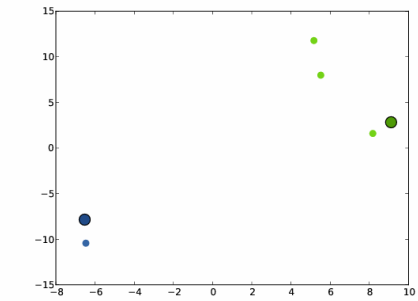

Abbildung 31: aktuelle Clusterung

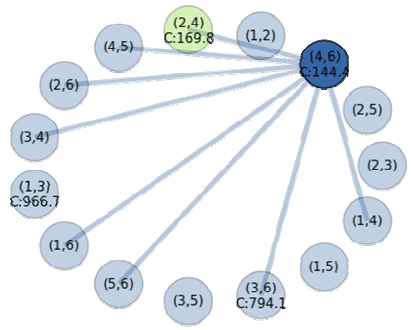

Abbildung 32: Nachbarknoten mit höheren Kosten, j=2

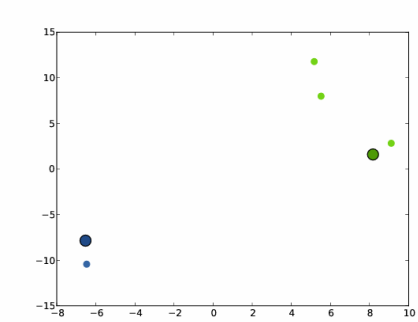

Abbildung 33: aktuelle Clusterung

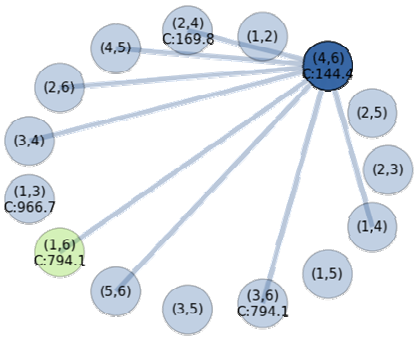

Abbildung 34: Nachbarknoten mit höheren Kosten, j=3

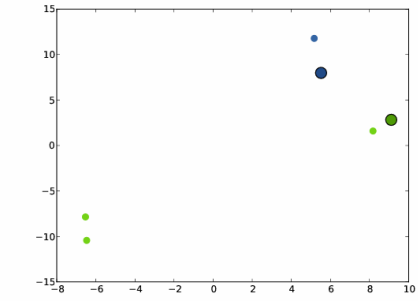

Abbildung 35: aktuelle Clusterung

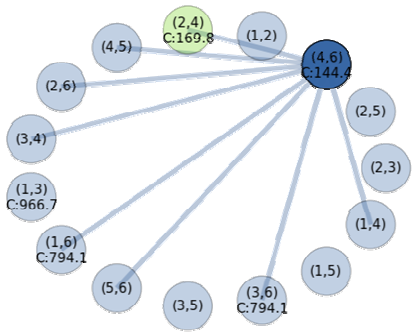

Abbildung 36: Nachbarknoten mit höheren Kosten, j=4

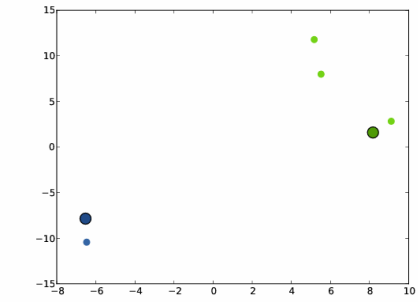

Abbildung 37: aktuelle Clusterung

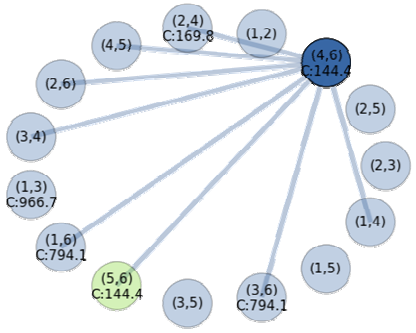

Abbildung 38: Nachbarknoten mit gleich hohen Kosten, kein neuer Medoid, j=*maxNeighbours*, Ende des Durchlaufs.

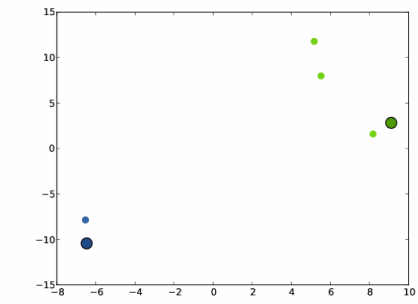

Abbildung 39: aktuelle Clusterung

6. Quellen

Hofmann, Constanze (2008): 1.3.1 Partitionierendes und Hierarchisches Clustern: CLARANS und BIRCH, FernUniversität Hagen, verfügbar unter: http://dna.fernuni-hagen.de/Lehre-offen/Seminare/1912-SS08/praesentationen/1-3-1_Custern_mit_CLARANS_und_BIRCH.pdf (letzter Zugriff: 27.05.2013), S. 9ff

Ng, Raymond T.; Han, Jiawei (2002): CLARANS: A Method for Clustering Objects for Spatial Data Mining, erschienen in: Knowledge and Data Enginering, IEEE Transactions, Vol. 14, No. 5, September/October 2002, S. 1003ff

Miller, Harvey J.; Han, Jiawei (2001): Geographic Data Mining and Knowledge Discovery, New York, NY, Taylor & Francis Inc., S. 207

Wikipedia: Hauptkomponentenanalyse, abrufbar unter: https://de.wikipedia.org/wiki/Hauptkomponentenanalyse#Anwendung_in_der_Clusteranalyse_und_Dimensionsreduktion (letzter Zugriff am: 5. Juni 2013)

Skikit learn: Comparing different clustering algorithms on toy datasets, abrufbar unter: http://scikit-learn.org/stable/auto_examples/cluster/plot_cluster_comparison.html (letzter Zugriff am: 5. Juni 2013)

Theodoridis Sergios; Koutroumbas, Konstantinos (2006): *Pattern Recognition 3rd ed.*, Academic Press, S. 635

Eidgenössische Technische Hochschule Zürich: PAM, abrufbar unter: http://stat.ethz.ch/R-manual/R-patched/library/cluster/html/pam.html (letzter Zugriff am: 6. Juni 2013)

Han, Jiawei; Kamber, Micheline; Pei, Jian (2012): Data mining: concepts and techniques - 3rd ed., Waltham, MA, Morgan Kaufmann Publishers, S. 456

7. Steckbrief

- CLARANS (**C**lustering **L**arge **A**pplications based on **RAN**domized **S**earch)
- **partitionierendes**, **k-Medoid** Verfahren
- **Distanzmaß:** euklidischer Abstand, weitere Metriken sind Möglich
- **Parameter:**
  - *k* (Anzahl der gewünschten Cluster)
  - *numlocal* (max. Anz. der Versuche zum Finden lokaler Minima)
  - *maxNeighbours* (Anz. max. nicht durchgeführter Tausche aufeinander)
  - die zu clusternden *Daten*
- **Algorithmus:**

  Für jeden Durchlauf – bis *numlocal* erreicht:
  - erzeuge zufällig *k* Medoiden

  - solange j kleiner *maxNeighbours*

    - Ersetzen eines zufällig ausgewählten Medoiden durch einen nicht-Medoiden

    - Berechnung der Summe der Distanzen aller Objekte zu ihrem Clusterzentrum

    - falls neue Summe kleiner als die aktuell optimale Summe: alter Medoid wird durch neuen ersetzt und j=1. Ansonsten j++

  - Berechnung der Gesamtdistanz der neuen Clusterung

  - Ist diese kleiner als die aktuell kleinste Gesamtdifferenz ersetzt die neue Clusterung die bis dato beste Clusterung

- **Gesamtlaufzeit** verhält sich im Mittel Quadratisch zur Anzahl der Objekte $O(n^2)$. Die innere Schleife verhält sich linear zur Anzahl der Objekte $O(n)$.

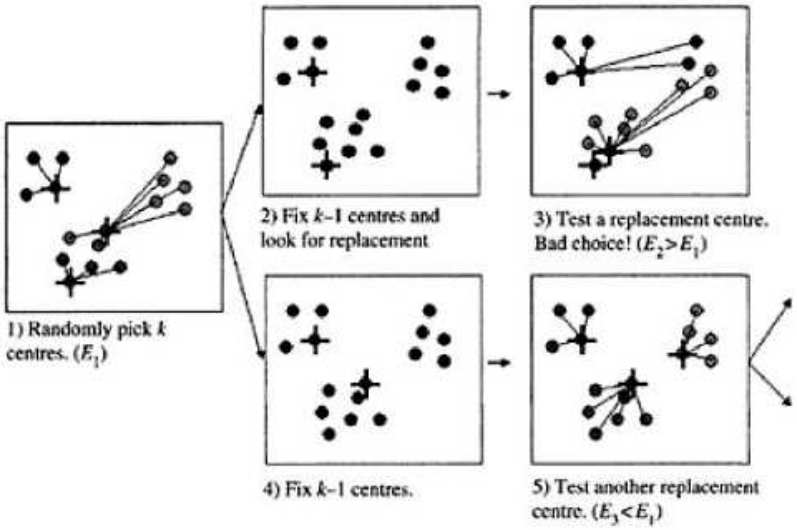

Bild 1: CLARANS Prinzip