

4.1 Einleitung

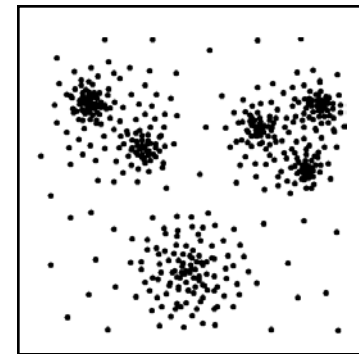
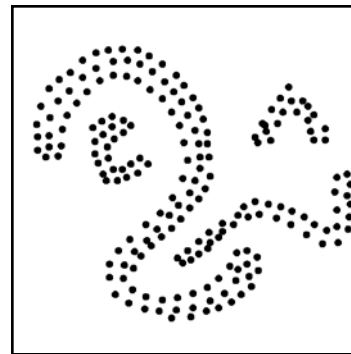
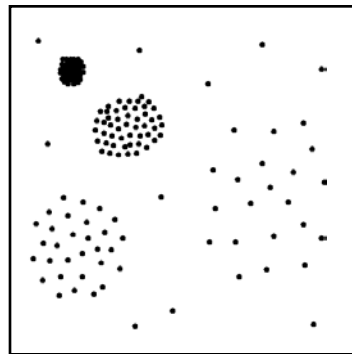
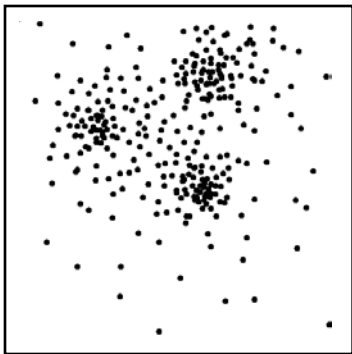
4.2 Clustering

4.3 Klassifikation

Ziel des Clustering

Ziel: Identifikation einer endlichen Menge von Kategorien, Klassen oder Gruppen (*Cluster*) in den Daten

- Objekte im *gleichen* Cluster sollen möglichst ähnlich sein
- Objekte aus *verschiedenen* Clustern sollen möglichst unähnlich zueinander sein



Herausforderungen:



- Cluster unterschiedlicher Größe, Form und Dichte
- hierarchische Cluster
- Rauschen (Noise)

=> unterschiedliche Clustering-Algorithmen

Typen von Clustering-Verfahren

- Partitionierende Verfahren
 - Parameter: Anzahl k der Cluster, Distanzfunktion
 - sucht ein „flaches“ Clustering in k Cluster mit minimalen Kosten
- Dichtebasierte Verfahren
 - Parameter: minimale Dichte in einem Cluster, Distanzfunktion
 - erweitert Punkte um ihre Nachbarn solange Dichte groß genug

Partitionierende Verfahren

Grundlagen

- Ziel
 - Partitionierung in k Cluster so dass eine Kostenfunktion minimiert wird (Gütekriterium)
- Lokal optimierendes Verfahren
 - wähle k initiale Cluster-Repräsentanten
 - optimiere diese Repräsentanten iterativ
 - ordne jedes Objekt seinem ähnlichsten Repräsentanten zu
- Typen von Cluster-Repräsentanten
 - Mittelwert des Clusters (*Centroid*)
 - Element des Clusters (*Medoid*)
 - Wahrscheinlichkeitsverteilung des Clusters (*Erwartungsmaximierung*)

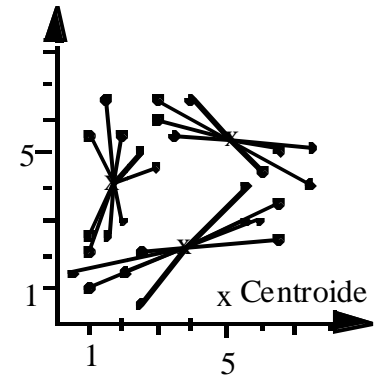
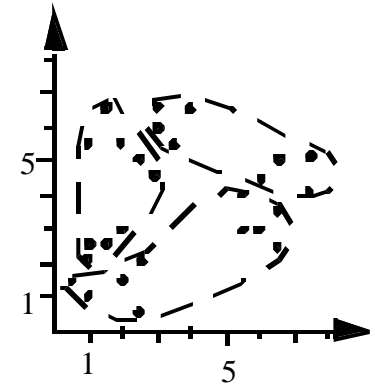
Partitionierende Verfahren

Beispiel

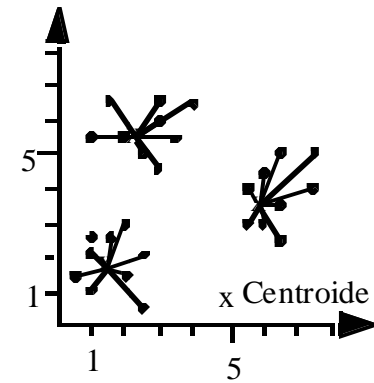
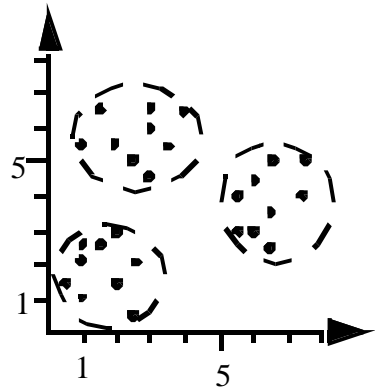
Cluster

Cluster-Repräsentanten

schlechtes Clustering



optimales Clustering



Partitionierende Verfahren: k-means

Grundbegriffe

- Objekte sind Punkte $p=(p_1, \dots, p_d)$ in einem euklidischen Vektorraum
- euklidische Distanz
- *Centroid* μ_C : Mittelwert aller Punkte im Cluster C
- *Maß für die Kosten* (Kompaktheit) eines Clusters C

$$TD^2(C) = \sum_{p \in C} dist(p, \mu_C)^2$$

- *Maß für die Kosten* (Kompaktheit) eines Clustering

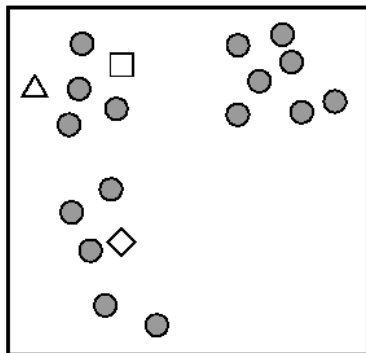
$$TD^2 = \sum_{i=1}^k TD^2(C_i)$$

Partitionierende Verfahren: k-means

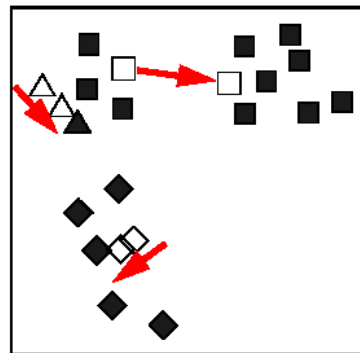
Idee des Algorithmus

- Algorithmus startet z.B. mit zufällig gewählten Punkten als Cluster-Repräsentanten
- Der Algorithmus besteht aus zwei alternierenden Schritten:
 - Zuordnung jedes Datenpunktes zum räumlich nächsten Repräsentanten
 - Neuberechnung der Repräsentanten (Centroid der zugeordneten Punkte)
- Diese Schritte werden so lange wiederholt, bis sich keine Änderung mehr ergibt

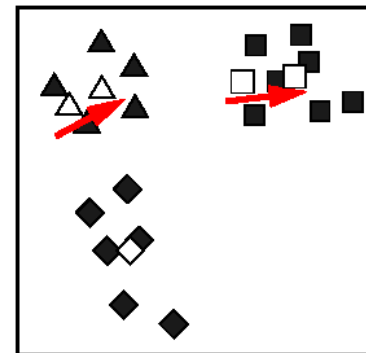
(a) Initialization



(b) First Iteration



(c) Convergence



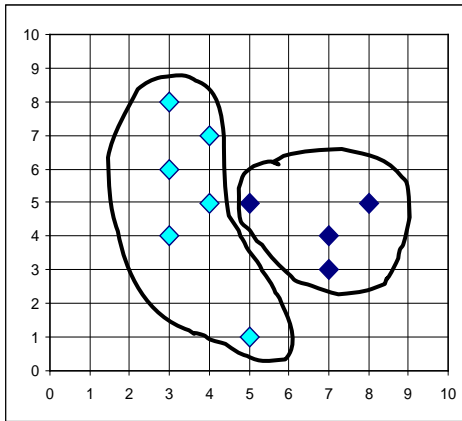
Partitionierende Verfahren: k-means

Algorithmus

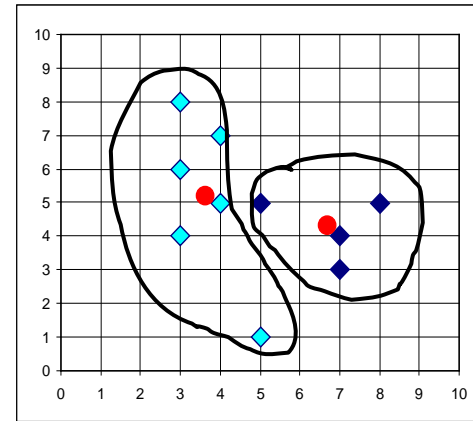
```
ClusteringDurchVarianzMinimierung(Punktmenge D, Integer k)
  Erzeuge eine „initiale“ Zerlegung der Punktmenge D in k Klassen;
  Berechne die Menge  $C' = \{C'_1, \dots, C'_k\}$  der Zentroide für die k Klassen;
  C = {};
  repeat
    C = C';
    Bilde k Klassen durch Zuordnung jedes Punktes zum
      nächstliegenden Zentroid aus C;
    Berechne die Menge  $C' = \{C'_1, \dots, C'_k\}$  der Zentroide für die neu
      bestimmten Klassen;
  until C = C';
  return C;
```


Partitionierende Verfahren: k-means

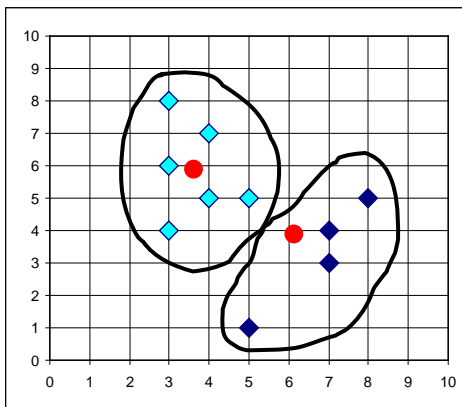
Beispiel



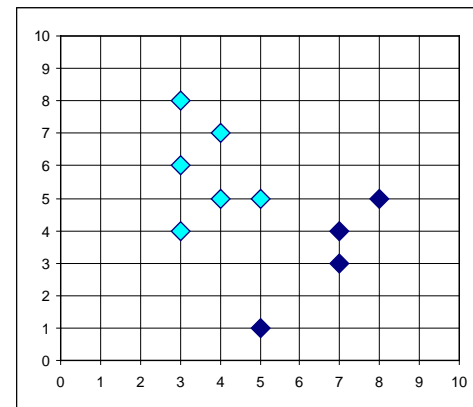
Berechnung der neuen Zentroide



Zuordnung zum nächsten Zentroid



Berechnung der neuen Zentroide



Partitionierende Verfahren: k-means

Diskussion

- + Effizienz:
 - Anzahl der Iterationen ist im allgemeinen klein (~ 5 - 10).
- + einfache Implementierung:
 - k-means ist das populärste partitionierende Clustering-Verfahren
- Anfälligkeit gegenüber Rauschen und Ausreißern
(alle Objekte gehen ein in die Berechnung des Zentroids)
- Cluster müssen konvexe Form haben
- die Anzahl k der Cluster muss bekannt sein
- starke Abhängigkeit von der initialen Zerlegung
(sowohl Ergebnis als auch Laufzeit)