Ludwig-Maximilians-Universität München
Lehrstuhl für Datenbanksysteme und Data Mining
Prof. Dr. Thomas Seidl

Knowledge Discovery and Data Mining I

Winter Semester 2018/19

Agenda

What is Clustering?

Clustering

Grouping a set of data objects into clusters (=collections of data objects).

- *Similar* to one another within the same cluster
- *Dissimilar* to the objects in other clusters

Typical Usage

- As a *stand-alone tool* to get insight into data distribution
- As a *preprocessing* step for other algorithms

General Applications of Clustering

- Preprocessing – as a data reduction (instead of sampling)
  - Image data bases (color histograms for filter distances)
  - Stream clustering (handle endless data sets for offline clustering)
- Pattern Recognition and Image Processing
- Spatial Data Analysis:
  - create thematic maps in Geographic Information Systems by clustering feature spaces
  - detect spatial clusters and explain them in spatial data mining
- Business Intelligence (especially market research)
- WWW
  - Documents (Web Content Mining)
  - Web-logs (Web Usage Mining)
- Biology, e.g. Clustering of gene expression data

Application Example: Downsampling Images

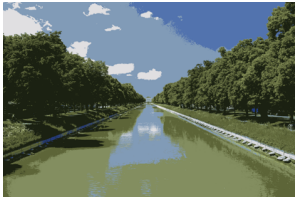
- Reassign color values to k distinct colors
- Cluster pixels using color difference, not spatial data

65536

256

16

8



4

2

Major Clustering Approaches

- Partitioning algorithms: Find k partitions, minimizing some objective function
- Probabilistic Model-Based Clustering (EM)
- Density-based: Find clusters based on connectivity and density functions
- Hierarchical algorithms: Create a hierarchical decomposition of the set of objects
- Other methods:
  - Grid-based
  - Neural networks (SOMs)
  - Graph-theoretical methods
  - Subspace Clustering

Agenda

Partitioning Algorithms: Basic Concept

Partition

Given a set D, a partitioning $\mathcal{C} = \{C_1, \ldots, C_k\}$ of D fulfils:

- $C_i \subseteq D$ for all $1 \leq i \leq k$
- $C_i \cap C_j = \emptyset \iff i \neq j$
- $\bigcup C_i = D$

(i.e. each element of D is in exactly one set C_i)

Goal

Construct a partitioning of a database D of n objects into a set of k ($k \leq n$) clusters minimizing an objective function.

Exhaustively enumerating all possible partitionings into k sets in order to find the global minimum is too expensive.

Partitioning Algorithms: Basic Concept

Popular Heuristic Methods

- Choose k representatives for clusters, e.g., randomly
- Improve these initial representatives iteratively:
  - Assign each object to the cluster it "fits best" in the current clustering
  - Compute new cluster representatives based on these assignments
  - Repeat until the change in the objective function from one iteration to the next drops below a threshold

Example

- k-means: Each cluster is represented by the center of the cluster
- k-medoid: Each cluster is represented by one of its objects

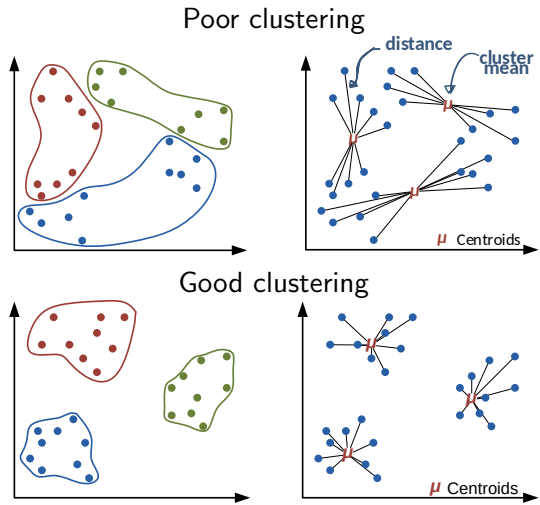
k-Means Clustering: Basic Idea

Idea[1]

Find a clustering such that the within-cluster variation of each cluster is small and use the centroid of a cluster as representative.

Objective

For a given k, form k groups so that the sum of the (squared) distances between the mean of the groups and their elements is minimal



[1] S.P. Lloyd: Least squares quantization in PCM. In IEEE Information Theory, 1982 (original version: technical report, Bell Labs, 1957)

k-Means Clustering: Basic Notions

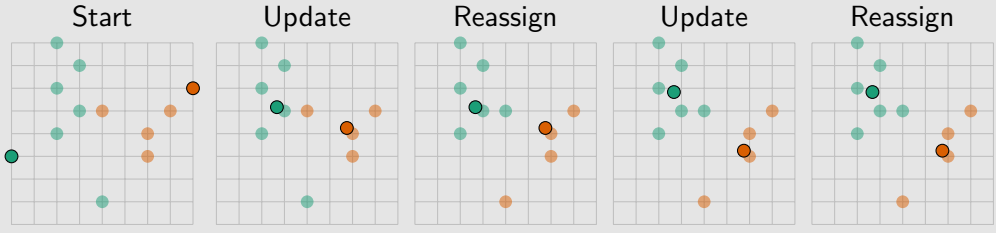
- Objects $p = (p_1, \ldots, p_d)$ are points in a d-dimensional vector space (the mean μ_S of a set of points S must be defined: $\mu_S = \frac{1}{|S|} \sum_{p \in S} p$)
- Measure for the compactness of a *cluster* C_j (sum of squared distances): $SSE(C_j) = \sum_{p \in C_j} ||p - \mu_{C_j}||_2^2$
- Measure for the compactness of a *clustering* $\mathcal{C}$: $SSE(\mathcal{C}) = \sum_{C_j \in \mathcal{C}} SSE(C_j) = \sum_{p \in D} ||p - \mu_{C(p)}||_2^2$
- Optimal Partitioning: $\operatorname{argmin}_{\mathcal{C}} SSE(\mathcal{C})$
- Optimizing the within-cluster variation is computationally challenging (NP-hard) $\rightsquigarrow$ use efficient heuristic algorithms

k-Means Clustering: Algorithm

k-Means Algorithm: Lloyd's algorithm

1: Given: k
2: Initialization: Choose k arbitrary representatives
3: **repeat**
4: Assign each object to the cluster with the nearest representative.
5: Compute the centroids of the clusters of the current partitioning.
6: **until** representatives do not change

Example



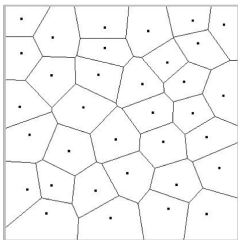
k-Means: Voronoi Model for Convex Cluster Regions

Voronoi Diagram

- For a given set of points $P = \{p_1, \ldots, p_k\}$ (here: cluster representatives), a *Voronoi diagram* partitions the data space into *Voronoi cells*, one cell per point
- The cell of a point $p \in P$ covers all points in the data space for which p is the nearest neighbors among the points from P

Observations

- The Voronoi cells of two neighboring points $p_i, p_j \in P$ are separated by the perpendicular hyperplane ("Mittelsenkrechte") between p_i and p_j.
- Voronoi cells are intersections of half spaces and thus convex regions



k-Means: Discussion

Strength

- Relatively efficient: $\mathcal{O}(tkn)$ (n: #obj., k: #clus., t: #it.; typically: $k, t \ll n$)
- Easy implementation

Weaknesses

- Applicable only when mean is defined
- Need to specify k, the number of clusters, in advance
- Sensitive to noisy data and outliers
- Clusters are forced to convex space partitions (Voronoi Cells)
- Result and runtime strongly depend on the initial partition; often terminates at a local optimum – however: methods for a good initialization exist

Variants: Basic Idea

One Problem of k-Means

Applicable only when mean is defined (vector space)

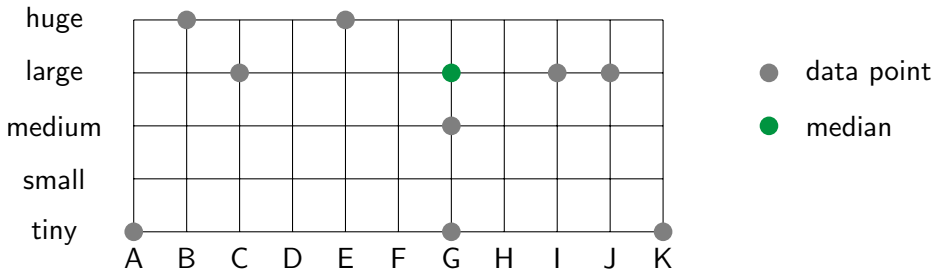
Alternatives for *Mean* representatives

- *Median*: (Artificial) Representative object "in the middle"
- *Mode*: Value that appears most often
- *Medoid*: Representative object "in the middle"

Objective

Find k representatives so that the sum of total distances (TD) between objects and their closest representative is minimal (more robust against outliers).

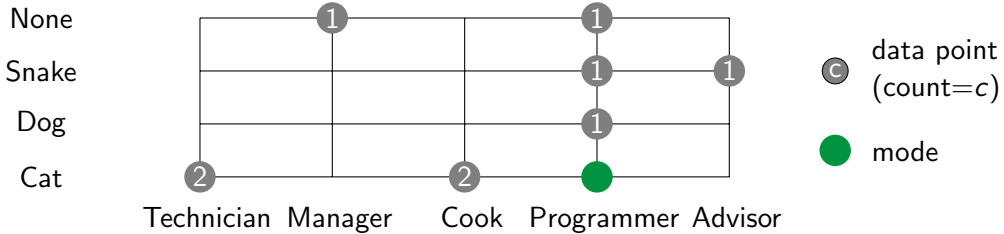
k-Median



Idea

- If there is an ordering on the data use median instead of mean.
- Compute median separately per dimension ($\rightsquigarrow$ efficient computation)

k-Mode



Mode

- Given: categorical data $D \subseteq \Omega = A_1 \times \cdots \times A_d$ where A_i are categorical attributes
- A *mode* of D is a vector $M = (m_1, \ldots, m_d) \in \Omega$ that minimizes $d(M, D) = \sum_{p \in D} d(p, M)$ where d is a distance function for categorical values (e.g. Hamming distance)
- Note: M is not necessarily an element of D

k-Mode

Theorem to determine a mode

Let $f(c, j, D) = \frac{1}{n}|\{p \in D \mid p[j] = c\}|$ be the relative frequency of category c of attribute A_j in the data, then:

$$d(M, D) \text{ is minimal } \Leftrightarrow \forall j \in \{1, \ldots, d\} \forall c \in A_j : f(m_j, j, D) \geq f(c, j, D)$$

- This allows to use the k-Means paradigm to cluster categorical data without losing its efficiency
- k-Modes algorithm[1] proceeds similar to k-Means algorithm
- Note: The mode of a dataset might be not unique

[1] Huang, Z. "A Fast Clustering Algorithm to Cluster very Large Categorical Data Sets in Data Mining" DMKD (1997)

k-Medoid

Potential problems with previous methods:

- Artificial centroid object might not make sense (e.g. education="high school" and occupation="professor")
- There might only be a distance function available but no explicit attribute-based data representations (e.g. Edit Distance on strings)

Partitioning Around Medoids [1]: Initialization

Given k, the *k-medoid* algorithm is initialized as follows:

- Select k objects arbitrarily as initial medoids (representatives)
- Assign each remaining (non-medoid) object to the cluster with the nearest representative
- Compute current $TD_{current}$

[1] Kaufman, Leonard, and Peter Rousseeuw. "Clustering by means of medoids." (1987)

k-Medoid

Partitioning Around Medoids (PAM) Algorithm

procedure PAM(Set D, Integer k)
  Initialize k medoids
  $\Delta_{TD} = -\infty$
  while $\Delta_{TD} < 0$ **do**
    Compute $TD_{N \leftrightarrow M}$ for each pair (medoid M, non-medoid N), i.e., TD after swapping M with N
    Choose pair (M, N) with minimal $\Delta_{TD} = TD_{N \leftrightarrow M} - TD_{current}$
    if $\Delta_{TD} < 0$ **then**
      Replace medoid M with non-medoid N
      $TD_{current} \leftarrow TD_{N \leftrightarrow M}$
      Store current medoids and assignments as best partitioning so far
  return medoids

- Problem with PAM: high complexity $\mathcal{O}\left(tk(n-k)^2\right)$
- Several heuristics can be employed, e.g. CLARANS [1]: randomly select (medoid, non-medoid)-pairs instead of considering all pairs

[1] Ng, Raymond T., and Jiawei Han. "CLARANS: A method for clustering objects for spatial data mining." IEEE TKDE (2002)

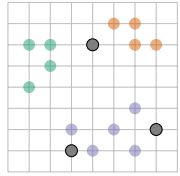
K-Means/Median/Mode/Medoid Clustering: Discussion

	k-**Means**	k-**Median**	k-**Mode**	k-**Medoid**
data	numerical (mean)	ordinal	categorical	metric
efficiency	high $\mathcal{O}(tkn)$			low $\mathcal{O}\left(tk(n-k)^2\right)$
sensitivity to outliers	high	low		

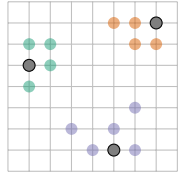
- Strength: Easy implementation (many variations and optimizations exist)
- Weaknesses
  - Need to specify k in advance
  - Clusters are forced to convex space partitions (Voronoi Cells)
  - Result and runtime strongly depend on the initial partition; often terminates at a local optimum – however: methods for good initialization exist

Initialization of Partitioning Clustering Methods

- Naive
  - Choose sample A of the dataset
  - Cluster A and use centers as initialization
- k-means++[1]
  - Select first center uniformly at random
  - Choose next point with probability proportional to the squared distance to the nearest center already chosen
  - Repeat until k centers have been selected
  - Guarantees an approximation ratio of $\mathcal{O}(\log k)$ (standard k-means can generate arbitrarily bad clusterings)
- In general: Repeat with different initial centers and choose result with lowest clustering error



Bad initialization



Good initialization

[1] Arthur, D., Vassilvitskii, S. "k-means++: The Advantages of Careful Seeding." ACM-SIAM Symposium on Discrete Algorithms (2007)

Choice of the Parameter k

- Idea for a method:
  - Determine a clustering for each $k = 2, \ldots, n-1$
  - Choose the "best" clustering
- But how to measure the quality of a clustering?
  - A measure should not be monotonic over k
  - The measures for the compactness of a clustering SSE and TD are monotonously decreasing with increasing value of k.

Silhouette-Coefficient [1]

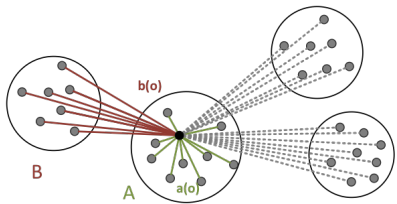
Quality measure for k-means or k-medoid clusterings that is not monotonic over k.

[1] Rousseeuw, P. "Silhouettes: A Graphical Aid to the Interpretation and Validation of Cluster Analysis". Computational and Applied Mathematics (1987)

The Silhouette Coefficient

Basic idea

- How good is the clustering = how appropriate is the mapping of objects to clusters
- Elements in cluster should be "similar" to their representative
  - Measure the average distance of objects to their representative: $a(o)$
- Elements in different clusters should be "dissimilar"
  - Measure the average distance of objects to alternative clusters (i.e. second closest cluster): $b(o)$



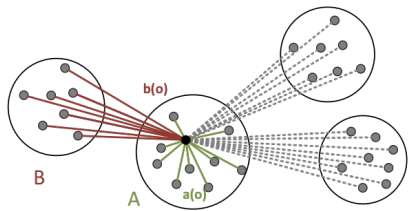
The Silhouette Coefficient

- $a(o)$ = "Avg. distance between o and objects in its cluster A."

$$a(o) = \frac{1}{|C(o)|} \sum_{p \in C(o)} d(o, p)$$

- $b(o)$: "Smallest avg. distance between o and objects in other cluster."

$$b(o) = \min_{C_i \neq C(o)} \left\{ \frac{1}{|C_i|} \sum_{p \in C_i} d(o, p) \right\}$$



The Silhouette Coefficient

- The silhouette of o is then defined as

$$s(o) = \begin{cases} 0 & \text{if } a(o) = 0, \text{ e.g. } |C_i| = 1 \\ \frac{b(o)-a(o)}{max(a(o),b(o))} & \text{else} \end{cases}$$

- The value range of the silhouette coefficient is $[-1, 1]$
- The silhouette of a cluster C_i is defined as

$$s(C_i) = \frac{1}{|C_i|} \sum_{o \in C_i} s(o)$$

- The silhouette of a clustering $\mathcal{C} = (C_1, \ldots, C_k)$ is defined as

$$s(\mathcal{C}) = \frac{1}{|D|} \sum_{o \in D} s(o)$$

where D denotes the whole dataset

The Silhouette Coefficient

- "Reading" the silhouette coefficient: Let $a(o) \neq 0$
  - $b(o) \gg a(o) \implies s(o) \approx 1$: good assignment of o to its cluster A
  - $b(o) \approx a(o) \implies s(o) \approx 0$: o is in-between A and B
  - $b(o) \ll a(o) \implies s(o) \approx -1$: bad, on average o is closer to members of B
- Silhouette coefficient $s(\mathcal{C})$ of a clustering: Average silhouette of all objects
  - $0.7 < s(\mathcal{C}) \leq 1.0$: strong structure
  - $0.5 < s(\mathcal{C}) \leq 0.7$: medium structure
  - $0.25 < s(\mathcal{C}) \leq 0.5$: weak structure
  - $s(\mathcal{C}) \leq 0.25$: no structure

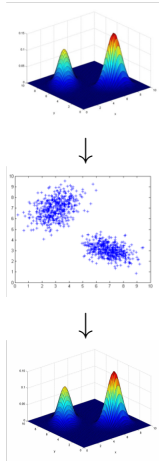
Silhouette Coefficient: Example

Image from Tan, Steinbach, Kumar: Introduction to Data Mining (Pearson, 2006)

Agenda

Expectation Maximization (EM)

- Statistical approach for finding maximum likelihood estimates of parameters in probabilistic models.
- Here: Using EM as clustering algorithm
- Approach: Observations are drawn from one of several components of a mixture distribution.
- Main idea:
  - Define clusters as probability distributions $\rightarrow$ each object has a certain probability of belonging to each cluster
  - Iteratively improve the parameters of each distribution (e.g. center, "width" and "height" of a Gaussian distribution) until some quality threshold is reached



Additional Literature: C. M. Bishop "Pattern Recognition and Machine Learning", Springer, 2009

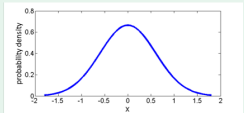
Excursus: Gaussian Mixture Distributions

Note: EM is not restricted to Gaussian distributions, but they will serve as example in this lecture.

Gaussian Distribution

- Univariate: single variable $x \in \mathbb{R}$:

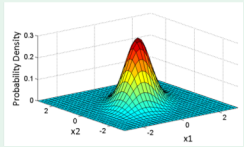
$$p(x \mid \mu, \sigma^2) = \mathcal{N}(x \mid \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(x-\mu)^2\right)$$



with *mean* $\mu \in \mathbb{R}$ and *variance* $\sigma^2 \in \mathbb{R}$

- Multivariate: d-dimensional vector $x \in \mathbb{R}^d$:

$$p(x \mid \mu, \Sigma) = \mathcal{N}(x \mid \mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^d |\Sigma|}} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right)$$



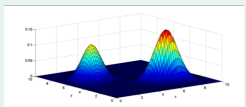
with *mean vector* $\mu \in \mathbb{R}^d$ and *covariance matrix* $\Sigma \in \mathbb{R}^{d \times d}$

Excursus: Gaussian Mixture Distributions

Gaussian mixture distribution with k components

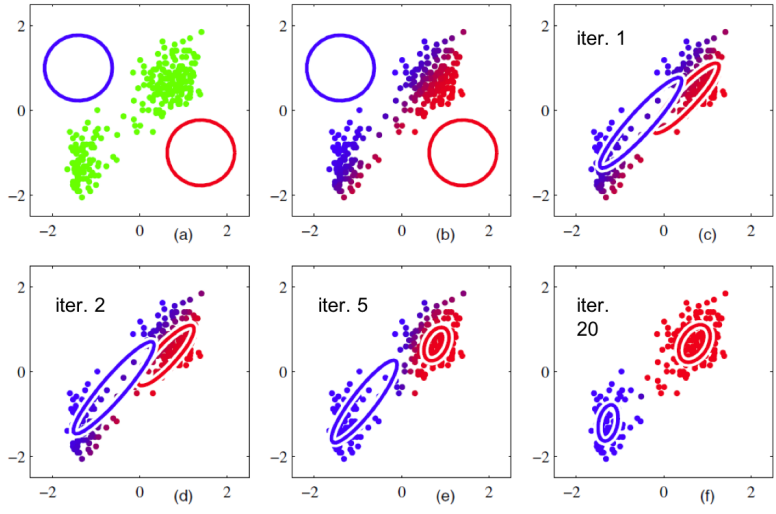
- For d-dimensional vector $x \in \mathbb{R}^d$:

$$p(x) = \sum_{l=1}^{k} \pi_l \cdot \mathcal{N}(x \mid \mu_l, \Sigma_l)$$



with *mixing coefficients* $\pi_l \in \mathbb{R}$, $\sum_l \pi_l = 1$ and $0 \leq \pi_l \leq 1$

EM: Exemplary Application



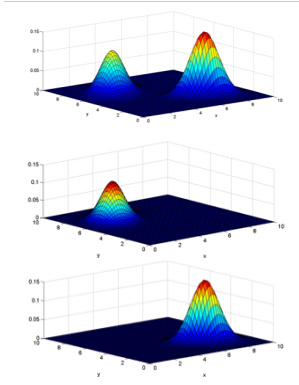
Example taken from: C. M. Bishop "Pattern Recognition and Machine Learning", 2009

EM: Clustering Model

Clustering

A clustering $\mathcal{M} = (C_1, \ldots, C_k)$ is represented by a mixture distribution with parameters $\theta = (\pi_1, \mu_1, \Sigma_1, \ldots, \pi_k, \mu_k, \Sigma_k)$:

$$p(x \mid \theta) = \sum_{l=1}^{k} \pi_l \cdot \mathcal{N}(x \mid \mu_l, \Sigma_l)$$



Cluster

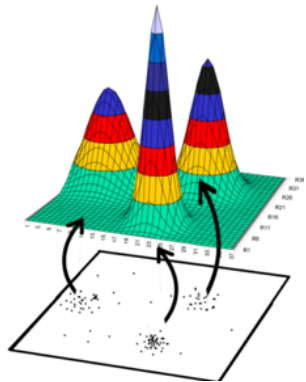
Each cluster is represented by one component of the mixture distribution:

$$p(x \mid \mu_l, \Sigma_l) = \mathcal{N}(x \mid \mu_l, \Sigma_l)$$

EM: Maximum Likelihood Estimation

- Given a dataset $X = \{x_1, \ldots, x_n\} \subseteq \mathbb{R}^d$, the *likelihood* that all data points $x_i \in X$ are generated (independently) by the mixture model with parameters θ is given as:

$$p(X \mid \theta) = \prod_{i=1}^{n} p(x_i \mid \theta)$$



Goal

Find the *maximum likelihood estimate (MLE)*, i.e., the parameters θ_{ML} with maximal likelihood:

$$\theta_{ML} = \underset{\theta}{\operatorname{argmax}} \{p(X \mid \theta)\}$$

EM: Maximum Likelihood Estimation

- Goal: Find MLE. For convenience, we use the log-likelihood:

$$\begin{aligned}\theta_{ML} &= \operatorname*{argmax}_{\theta} \{p(X \mid \theta)\} \\ &= \operatorname*{argmax}_{\theta} \{\log p(X \mid \theta)\}\end{aligned}$$

- The log-likelihood can be written as

$$\begin{aligned}\log p(X \mid \theta) &= \log \prod_{i=1}^{n} \sum_{l=1}^{k} \pi_l \cdot p(x_i \mid \mu_l, \Sigma_l) \\ &= \sum_{i=1}^{n} \log \sum_{l=1}^{k} \pi_l \cdot p(x_i \mid \mu_l, \Sigma_l)\end{aligned}$$

EM: Maximum Likelihood Estimation

- Maximization w.r.t. the means:

$$\frac{\partial \log p(X \mid \theta)}{\partial \mu_j} = \sum_{i=1}^{n} \frac{\partial \log p(x_i \mid \theta)}{\partial \mu_j} = \sum_{i=1}^{n} \frac{\frac{\partial \log p(x_i|\theta)}{\partial \mu_j}}{p(x_i \mid \theta)} = \sum_{i=1}^{n} \frac{\frac{\partial \log p(x_i|\theta)}{\partial \mu_j}}{\sum_{l=1}^{k} p(x_i \mid \mu_l, \Sigma_l)}$$

$$= \sum_{i=1}^{n} \frac{\pi_j \cdot \Sigma_j^{-1}(x_i - \mu_j) \cdot \mathcal{N}(x_i \mid \mu_j, \Sigma_j)}{\sum_{l=1}^{k} p(x_i \mid \mu_l, \Sigma_l)}$$

$$= \Sigma_j^{-1} \sum_{i=1}^{n} (x_i - \mu_j) \frac{\pi_j \cdot \mathcal{N}(x_i \mid \mu_j, \Sigma_j)}{\sum_{l=1}^{k} \pi_l \cdot \mathcal{N}(x_i \mid \mu_l, \Sigma_l)} \stackrel{!}{=} 0$$

- Use $\frac{\partial}{\partial \mu_j} \mathcal{N}(x_i \mid \mu_j, \Sigma_j) = \Sigma_j^{-1}(x_i - \mu_j) \cdot \mathcal{N}(x_i \mid \mu_j, \Sigma_j)$
- Define $\gamma_j(x_i) := \pi_j \cdot \mathcal{N}(x_i \mid \mu_j, \Sigma_j)$: Probability that component j generated x_i

EM: Maximum Likelihood Estimation

- Maximization w.r.t. the means yields

$$\mu_j = \frac{\sum_{i=1}^{n} \gamma_j(x_i) x_i}{\sum_{i=1}^{n} \gamma_j(x_i)}$$

- Maximization w.r.t. the covariance matrices yields

$$\Sigma_j = \frac{\sum_{i=1}^{n} \gamma_j(x_i)(x_i - \mu_j)(x_i - \mu_j)^T}{\sum_{i=1}^{n} \gamma_j(x_i)}$$

- Maximization w.r.t. the mixing coefficients yields

$$\pi_j = \frac{\sum_{i=1}^{n} \gamma_j(x_i)}{\sum_{l=1}^{k} \sum_{i=1}^{n} \gamma_l(x_i)}$$

EM: Maximum Likelihood Estimation

Problem with finding the optimal parameters θ_{ML}:

$$\mu_j = \frac{\sum_{i=1}^{n} \gamma_j(x_i) x_i}{\sum_{i=1}^{n} \gamma_j(x_i)} \quad \text{and} \quad \gamma_j(x_i) = \frac{\pi_j \cdot \mathcal{N}(x_i \mid \mu_j, \Sigma_j)}{\sum_{l=1}^{k} \pi_j \cdot \mathcal{N}(x_i \mid \mu_l, \Sigma_k)}$$

- Non-linear mutual dependencies
- Optimizing the Gaussian of cluster j depends on all other Gaussians.
- There is no closed-form solution!
- Approximation through iterative optimization procedures
- Break the mutual dependencies by optimizing μ_j and $\gamma_j(x_i)$ independently

EM: Iterative Optimization

Iterative Optimization

1. Initialize means μ_j, covariances Σ_j, and mixing coefficients π_j and evaluate the initial log-likelihood.
2. **E-step**: Evaluate the responsibilities using the current parameter values:

$$\gamma_j^{new}(x_i) = \frac{\pi_j \cdot \mathcal{N}(x_i \mid \mu_j, \Sigma_j)}{\sum_{l=1}^{k} \pi_j \cdot \mathcal{N}(x_i \mid \mu_l, \Sigma_l)}$$

3. **M-step**: Re-estimate the parameters using the current responsibilities:

$$\mu_j^{new} = \frac{\sum_{i=1}^{n} \gamma_j^{new}(x_i) x_i}{\sum_{i=1}^{n} \gamma_j^{new}(x_i)}$$

$$\vdots$$

EM: Iterative Optimization

Iterative Optimization

$$\vdots$$

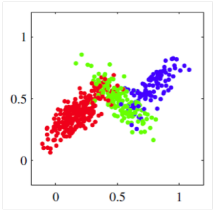
$$\Sigma_j^{new} = \frac{\sum_{i=1}^{n} \gamma_j^{new}(x_i)(x_i - \mu_j^{new})(x_i - \mu_j^{new})^T}{\sum_{i=1}^{n} \gamma_j^{new}(x_i)}$$

$$\pi_j^{new} = \frac{\sum_{i=1}^{n} \gamma_j^{new}(x_i)}{\sum_{l=1}^{k} \sum_{i=1}^{n} \gamma_l^{new}(x_i)}$$

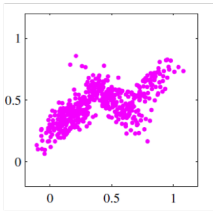
4. Evaluate the new log-likelihood $\log p(X \mid \theta^{new})$ and check for convergence of parameters or log-likelihood ($|\log p(X \mid \theta^{new}) - \log p(X \mid \theta)| \leq \epsilon$). If the convergence criterion is not satisfied, set $\theta = \theta^{new}$ and go to step 2.

EM: Turning the Soft Clustering into a Partitioning

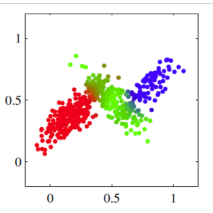
- EM obtains a soft clustering (each object belongs to each cluster with a certain probability) reflecting the uncertainty of the most appropriate assignment



original data



input for EM



soft clustering result of EM

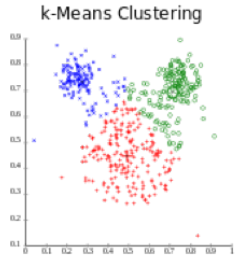
- Modification to obtain a partitioning variant: Assign each object to the cluster to which it belongs with the highest probability

$$C(x_i) = \underset{l \in \{1,\ldots,k\}}{\operatorname{argmax}} \{\gamma_l(x_i)\}$$

Example taken from: C. M. Bishop "Pattern Recognition and Machine Learning", 2009

EM: Discussion

- Superior to k-Means for clusters of varying size or clusters having differing variances
  - More accurate data representation
- Convergence to (possibly local) maximum
- Computational effort for t iterations: $\mathcal{O}(tnk)$
  - t is quite high in many cases
- Both, result and runtime, strongly depend on
  - the initial assignment
    - Do multiple random starts and choose the final estimate with highest likelihood
    - Initialize with clustering algorithms (e.g., k-Means): usually converges much faster
    - Local maxima and initialization issues have been addressed in various extensions of EM
  - a proper choice of k (next slide)



EM: Model Selection for Determining Parameter k

Problem

Classical trade-off problem for selecting the proper number of components k:

- If k is too high, the mixture may overfit the data
- If k is too low, the mixture may not be flexible enough to approximate the data

Idea

Determine candidate models θ_k for $k \in \{k_{min}, \ldots, k_{max}\}$ and select the model according to some quality measure *qual*:

$$\theta_{k^*} = \max_{k \in \{k_{min}, \ldots, k_{max}\}} \{qual(\theta_k)\}$$

- Silhouette Coefficient (as for k-Means) only works for partitioning approaches
- The likelihood is nondecreasing in k

EM: Model Selection for Determining Parameter k

Solution

Deterministic or stochastic *model selection* methods [1] which try to balance the goodness of fit with simplicity.

- Deterministic:

$$qual(\theta_k) = \log p(X \mid \theta_k) + \mathcal{P}(k)$$

  where $\mathcal{P}(k)$ is an increasing function penalizing higher values of k
- Stochastic: Based on Markov Chain Monte Carlo (MCMC)

[1] G. McLachlan and D. Peel. Finite Mixture Models. Wiley, New York, 2000.

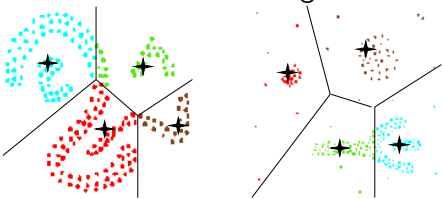
Agenda

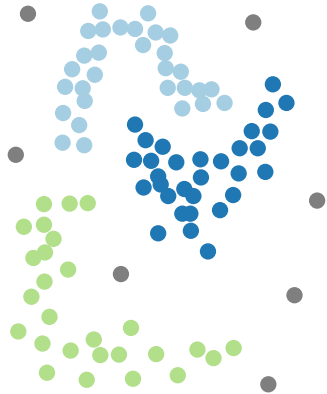
Density-Based Clustering

Basic Idea

Clusters are dense regions in the data space, separated by regions of lower density

Results of a k-medoid algorithm for $k = 4$:





Density-Based Clustering: Basic Concept

Note

Different density-based approaches exist in the literature. Here we discuss the ideas underlying the DBSCAN algorithm.

Intuition for Formalization

- For any point in a cluster, the local point density around that point has to exceed some threshold
- The set of points from one cluster is spatially connected

Density-Based Clustering: Basic Concept

Local Point Density

Local point density at a point q defined by two parameters:

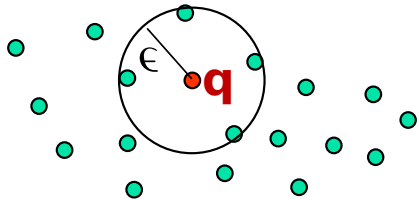
- ϵ-radius for the neighborhood of point q

$$N_\epsilon(q) = \{p \in D \mid dist(p, q) \leq \epsilon\} \tag{1}$$

In this chapter, we assume that $q \in N_\epsilon(q)$!

- *MinPts*: minimum number of points in the given neighbourhood $N_\epsilon(q)$.

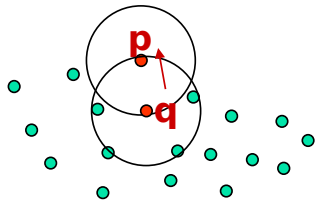
Density-Based Clustering: Basic Concept

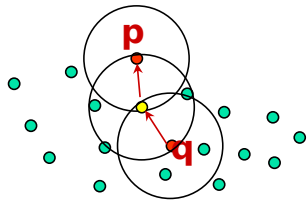


Core Point

q is called a core object (or core point) w.r.t. ϵ, *MinPts* if $|N_\epsilon(q)| \geq minPts$

Density-Based Clustering: Basic Definitions





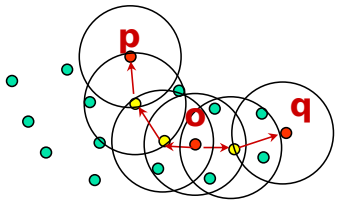
(Directly) Density-Reachable

p directly density-reachable from q w.r.t. ϵ, *MinPts* if:

1. $p \in N_\epsilon(q)$ and
2. q is core object w.r.t. ϵ, *MinPts*

Density-reachable is the transitive closure of directly density-reachable

Density-Based Clustering: Basic Definitions



Density-Connected

p is *density-connected* to a point q w.r.t. ϵ, *MinPts* if there is a point o such that both, p and q are density-reachable from o w.r.t. ϵ, *MinPts*

Density-Based Clustering: Basic Definitions

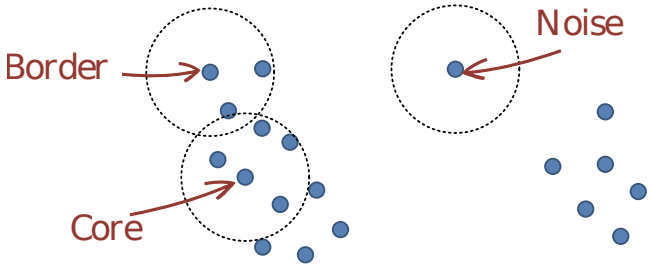
Density-Based Cluster

$\emptyset \subset C \subseteq D$ with database D satisfying:

Maximality: If $q \in C$ and p is density-reachable from q then $p \in C$

Connectivity: Each object in C is density-connected to all other objects in C

Density-Based Clustering: Basic Definitions



Density-Based Clustering

A partitioning $\{C_1, \ldots, C_k, N\}$ of the database D where

- $C_1, \ldots, C_k$ are all density-based clusters
- $N = D \setminus (C_1 \cup \ldots \cup C_k)$ is called the *noise* (objects not in any cluster)

Density-Based Clustering: DBSCAN Algorithm

Basic Theorem

- Each object in a density-based cluster C is density-reachable from any of its core-objects
- Nothing else is density-reachable from core objects.

Density-Based Clustering: DBSCAN Algorithm

Density-**B**ased **S**patial **C**lustering of **A**pplications with **N**oise[12]

1: **for all** $o \in D$ **do**
2: **if** o is not yet classified **then**
3: **if** o is a core-object **then**
4: Collect all objects density-reachable from o and assign them to a new cluster.
5: **else**
6: Assign o to noise N

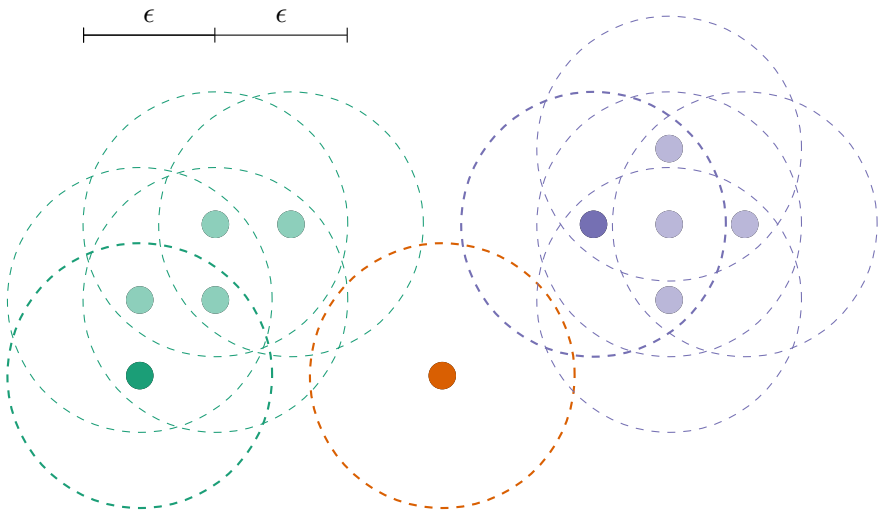
Note

Density-reachable objects are collected by performing successive ϵ-neighborhood queries.

[12] Ester M., Kriegel H.-P., Sander J., Xu X.: "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise", In KDD 1996 , pp. 226-231.

DBSCAN: Example

Parameters: $\epsilon = 1.75$, $minPts = 3$. Clusters: C_1, C_2; Noise: N



Determining the Parameters ϵ and *MinPts*

Recap

Cluster: Point density higher than specified by ϵ and *MinPts*

Idea

Use the point density of the least dense cluster in the data set as parameters.

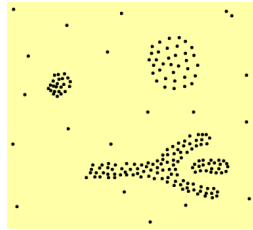
Problem

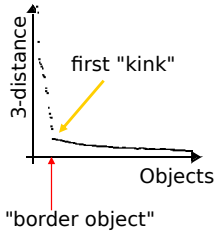
How to determine this?

Determining the Parameters ϵ and *MinPts*

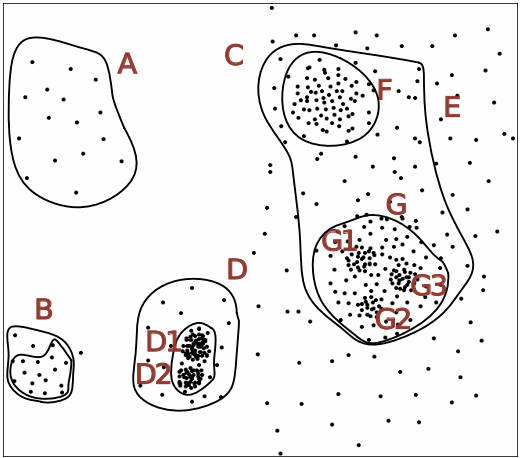
Heuristic

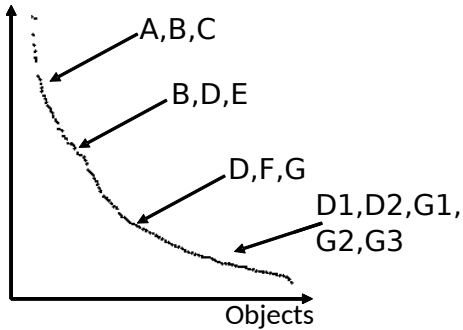
1. Fix a value for *MinPts* (default: $2d - 1$ where d is the dimension of the data space)
2. Compute the k-distance for all points $p \in D$ (distance from p to the its k-nearest neighbor), with $k = minPts$.
3. Create a k-distance plot, showing the k-distances of all objects, sorted in decreasing order
4. The user selects "border object" o from the *MinPts*-distance plot: ϵ is set to *MinPts*-distance(o).





Determining the Parameters ϵ and *MinPts*: Problematic Example





Database Support for Density-Based Clustering

Standard DBSCAN evaluation is based on recursive database traversal. Böhm et al.[13] observed that DBSCAN, among other clustering algorithms, may be efficiently built on top of similarity join operations.

ϵ-Similarity Join

An ϵ-*similarity join* yields all pairs of ϵ-similar objects from two data sets Q, P:

$$Q \bowtie_\epsilon P = \{(q, p) \in Q \times P \mid dist(q, p) \leq \epsilon\}$$

SQL Query

SELECT $*$ FROM Q, P WHERE $dist(Q, P) \leq \epsilon$

[13] Böhm C., Braunmüller, B., Breunig M., Kriegel H.-P.: *High performance clustering based on the similarity join*. CIKM 2000: 298-305.

Database Support for Density-Based Clustering

ϵ-Similarity Self-Join

An ϵ-similarity *self* join yields all pairs of ϵ-similar objects from a database D.

$$D \bowtie_\epsilon D = \{(q, p) \in D \times D \mid dist(q, p) \leq \epsilon\}$$

SQL Query

SELECT $*$ FROM D q, D p WHERE $dist(q, p) \leq \epsilon$

Database Support for Density-Based Clustering

The relation "directly ϵ, *MinPts*-density reachable" may be expressed in terms of an ϵ-similarity self join (abbreviate *minPts* with μ):

$$\begin{aligned} ddr_{\epsilon,\mu} &= \{(q,p) \in D \times D \mid q \text{ is } \epsilon,\mu\text{-core-point} \wedge p \in N_\epsilon(q)\} \\ &= \{(q,p) \in D \times D \mid dist(q,p) \leq \epsilon \wedge \exists_{\geq \mu} p' \in D : dist(q,p') \leq \epsilon\} \\ &= \{(q,p) \in D \times D \mid (q,p) \in D \bowtie_\epsilon D \wedge \exists_{\geq \mu} p'(q,p') \in D \bowtie_\epsilon D\} \\ &= \sigma_{|\pi_q(D \bowtie_\epsilon D)| \geq \mu}(D \bowtie_\epsilon D) =: D \bowtie_{\epsilon,\mu} D \end{aligned}$$

SQL Query

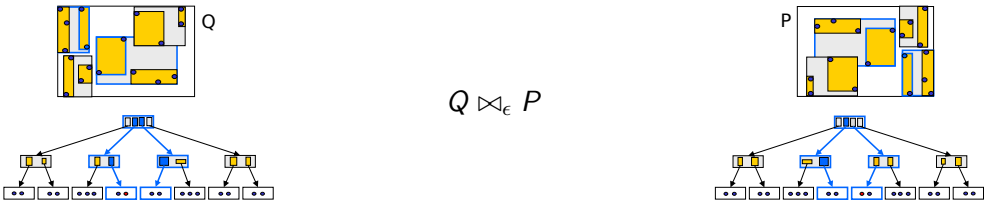
SELECT $*$ FROM D q, D p WHERE $dist(q,p) \leq \epsilon$ GROUP BY $q.id$ HAVING $count(q.id) \geq \mu$

Afterwards, DBSCAN computes the connected components of $D \bowtie_{\epsilon,\mu} D$.

Efficient Similarity Join Processing

For very large databases, efficient join techniques are available

- Block nested loop or index-based nested loop joins exploit secondary storage structure of large databases.
- Dedicated similarity join, distance join, or spatial join methods based on spatial indexing structures (e.g., R-Tree) apply particularly well. They may traverse their hierarchical directories in parallel (see illustration below).
- Other join techniques including sort-merge join or hash join are not applicable.



DBSCAN: Discussion

Advantages

- Clusters can have arbitrary shape and size; no restriction to convex shapes
- Number of clusters is determined automatically
- Can separate clusters from surrounding noise
- Complexity: N_ϵ-query: $\mathcal{O}(n)$, DBSCAN: $\mathcal{O}(n^2)$.
- Can be supported by spatial index structures ($\rightsquigarrow$ N_ϵ-query: $\mathcal{O}(\log n)$)

Disadvantages

- Input parameters may be difficult to determine
- In some situations very sensitive to input parameter setting

Agenda

Iterative Mode Search

Idea

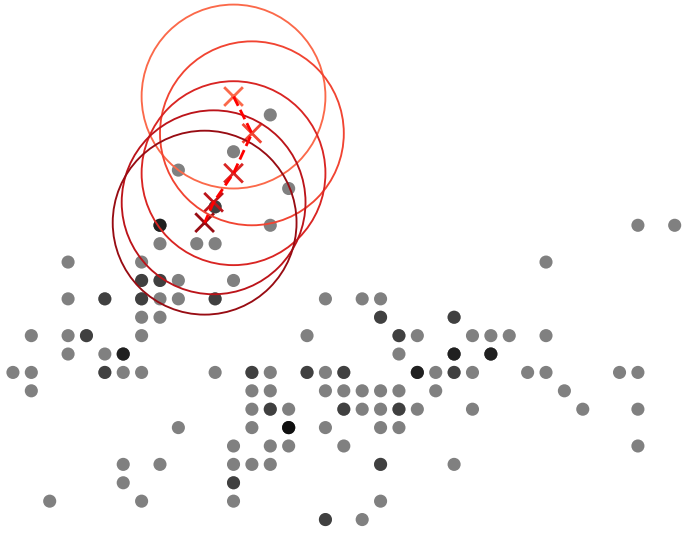
Find modes in the point density.

Algorithm[14]

1. Select a window size ϵ, starting position m
2. Calculate the mean of all points inside the window $W(m)$.
3. Shift the window to that position
4. Repeat until convergence.

[14] K. Fukunaga, L. Hostetler: *The Estimation of the Gradient of a Density Function, with Applications in Pattern Recognition*, IEEE Trans Information Theory, 1975

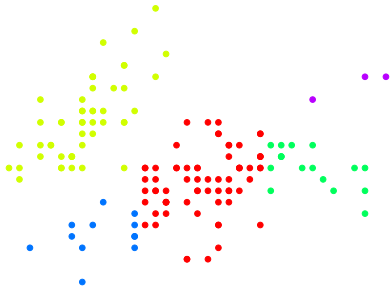
Iterative Mode Search: Example



Mean Shift: Core Algorithm

Algorithm[15]

Apply iterative mode search for each data point. Group those that converge to the same mode (called *Basin of Attraction*).



[15] D. Comaniciu, P. Meer. *Mean shift: A robust approach toward feature space analysis*. IEEE Trans. on pattern analysis and machine intelligence, 2002

Mean Shift: Extensions

Weighted Mean

Use different weights for the points in the window calculated by some kernel κ

$$m^{(i+1)} = \frac{\sum_{x \in W(m^{(i)})} \kappa(x) x}{\sum_{x \in W(m^{(i)})} \kappa(x)}$$

Binning

First quantise data points to grid. Apply iterative mode seeking only once per bin.

Mean Shift: Discussion

Disadvantages

- Relatively high complexity: N_ϵ-query (=windowing): $\mathcal{O}(n)$. Algorithm: $\mathcal{O}(tn^2)$

Advantages

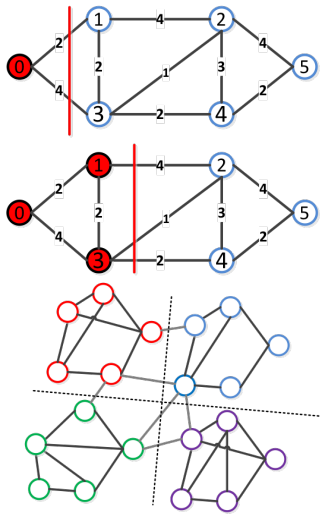
- Clusters can have arbitrary shape and size; no restriction to convex shapes
- Number of clusters is determined automatically
- Robust to outliers
- Easy implementation and parallelisation
- Single parameter: ϵ
- Support by spatial index: N_ϵ-query (=windowing): $\mathcal{O}(\log n)$. Algorithm: $\mathcal{O}(tn \log n)$

Agenda

Clustering as Graph Partitioning

Approach

- Data is modeled by a similarity graph $G = (V, E)$
  - Vertices $v \in V$: Data objects
  - Weighted edges $\{v_i, v_j\} \in E$: Similarity of v_i and v_j
  - Common variants: ϵ-neighborhood graph, k-nearest neighbor graph, fully connected graph
- Cluster the data by partitioning the similarity graph
  - Idea: Find global minimum cut
    - Only considers inter-cluster edges, tends to cut small vertex sets from the graph
    - Partitions graph into two clusters
  - Instead, we want a *balanced multi-way partitioning*
  - Such problems are NP-hard, use approximations



Spectral Clustering

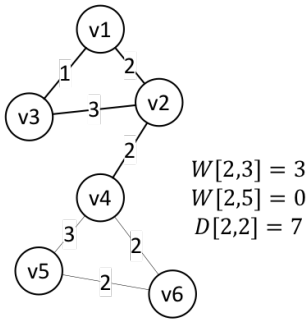
Given

Undirected graph G with weighted edges

- Let W be the (weighted) adjacency matrix of the graph
- And D its degree matrix with $D_{ii} = \sum_{j=1}^{n} W_{ij}$; other entries are 0

Aim

Partition G into k subsets, minimizing a function of the edge weights between/within the partitions.



Spectral Clustering

Idea

- Consider the *indicator vector* f_C for the cluster C, i.e.

$$f_{C\,i} = \begin{cases} 1 & \text{if } v_i \in C \\ 0 & \text{else} \end{cases}$$

  and the *Laplacian* matrix $L = D - W$
- Further, consider the function $fLf^T = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} W_{ij}(f_i - f_j)^2$ (derivation on next slide)
  - Small if f corresponds to a good partitioning
  - Given an indicator vector f_C, the function $f_C L f_C^T$ measures the weight of the inter-cluster edges!
  - Since L is positive semi-definite we have $fLf^T \geq 0$
  - Try to minimize fLf^T

Spectral Clustering

$$
\begin{aligned}
fLf^T &= fDf^T - fWf^T \\
&= \sum_i d_i f_i^2 - \sum_{ij} w_{ij} f_i f_j \\
&= \frac{1}{2}\left(\sum_i (\sum_j w_{ij}) f_i^2 - 2\sum_{ij} w_{ij} f_i f_j + \sum_j (\sum_i w_{ij}) f_j^2 \right) \\
&= \frac{1}{2}\left(\sum_{ij} w_{ij} f_i^2 - 2\sum_{ij} w_{ij} f_i f_j + \sum_{ij} w_{ij} f_j^2 \right) \\
&= \frac{1}{2} \sum_{ij} w_{ij} (f_i^2 - 2 f_i f_j + f_j^2) \\
&= \frac{1}{2} \sum_{ij} w_{ij} (f_i - f_j)^2
\end{aligned}
$$

Spectral Clustering: Example for Special Case

- Special case: The graph consists of k connected components (here: $k = 3$)
- The k components yield a "perfect" clustering (no edges between clusters), i.e. optimal clustering by indicator vectors $f_{C_1} = (1, 1, 1, 0, 0, 0, 0, 0, 0)$, $f_{C_2} = (0, 0, 0, 1, 1, 1, 0, 0, 0)$ and $f_{C_1} = (0, 0, 0, 0, 0, 0, 1, 1, 1)$

0	1	1	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0
0	0	0	0	1	1	0	0	0
0	0	0	1	0	2	0	0	0
0	0	0	1	2	0	0	0	0
0	0	0	0	0	0	0	3	1
0	0	0	0	0	0	3	0	1
0	0	0	0	0	0	1	1	0

Adjacency matrix W

2	0	0	0	0	0	0	0	0
0	2	0	0	0	0	0	0	0
0	0	2	0	0	0	0	0	0
0	0	0	2	0	0	0	0	0
0	0	0	0	3	0	0	0	0
0	0	0	0	0	3	0	0	0
0	0	0	0	0	0	4	0	0
0	0	0	0	0	0	0	4	0
0	0	0	0	0	0	0	0	2

Degree matrix D

2	-1	-1	0	0	0	0	0	0
-1	2	-1	0	0	0	0	0	0
-1	-1	2	0	0	0	0	0	0
0	0	0	2	-1	-1	0	0	0
0	0	0	-1	3	-2	0	0	0
0	0	0	-1	-2	3	0	0	0
0	0	0	0	0	0	4	-3	-1
0	0	0	0	0	0	-3	4	-1
0	0	0	0	0	0	-1	-1	2

Laplacian matrix $L = D - W$

- Because of the block form of L, we get $f_C L f_C^T = 0$ for each component C

Connected Components and Eigenvectors

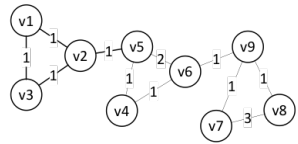
- General goal: find indicator vectors minimizing function fLf^T besides the trivial indicator vector $f_C = (1, \ldots, 1)$
- Problem: Finding solution is NP-hard (cf. graph cut problems)
- How can we relax the problem to find a (good) solution more efficiently?
- Observation: For the special case with k connected components, the k indicator vectors fulfilling $f_C L f_C^T = 0$ yield the perfect clustering
  - The indicator vector for each component is an eigenvector of L with eigenvalue 0
  - The k indicator vectors are orthogonal to each other (linearly independent)

Lemma

The number of linearly independent eigenvectors with eigenvalue 0 for L equals the number of connected components in the graph.

Spectral Clustering: General Case

- In general: L does not have zero-eigenvectors
  - One large connected component, no perfect clustering
  - Determine the (linear independent) eigenvectors with the *k smallest eigenvalues*!
- Example: The 3 clusters are now connected by additional edges



0	1	1	0	0	0	0	0	0
1	0	1	0	1	0	0	0	0
1	1	0	0	0	0	0	0	0
0	0	0	0	1	1	0	0	0
0	1	0	1	0	2	0	0	0
0	0	0	1	2	0	0	0	1
0	0	0	0	0	0	0	3	1
0	0	0	0	0	0	3	0	1
0	0	0	0	0	1	1	1	0

Adjacency matrix W

2	0	0	0	0	0	0	0	0
0	3	0	0	0	0	0	0	0
0	0	2	0	0	0	0	0	0
0	0	0	2	0	0	0	0	0
0	0	0	0	4	0	0	0	0
0	0	0	0	0	4	0	0	0
0	0	0	0	0	0	4	0	0
0	0	0	0	0	0	0	4	0
0	0	0	0	0	0	0	0	3

Degree matrix D

2	-1	-1	0	0	0	0	0	0
-1	3	-1	0	-1	0	0	0	0
-1	-1	2	0	0	0	0	0	0
0	0	0	2	-1	-1	0	0	0
0	-1	0	-1	4	-2	0	0	0
0	0	0	-1	-2	4	0	0	-1
0	0	0	0	0	0	4	-3	-1
0	0	0	0	0	0	-3	4	-1
0	0	0	0	0	-1	-1	-1	3

Laplacian matrix L

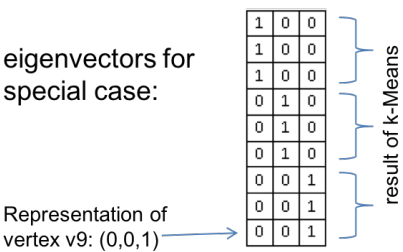
-1.3	3.3	0.4
-1	1	-1
-1.3	3.3	0.4
0	-6.6	0
-0.2	-4.3	-0.4
-0.2	-4.3	0.4
1.3	3.3	-0.4
1.3	3.3	-0.4
1	1	1

Eigenvectors of L

- Smallest eigenvalues of L: $(0.23, 0.70, 3.43)$

Spectral Clustering: Data Transformation

- How to find the clusters based on the eigenvectors?
  - Easy in special setting: 0-1 values; now: arbitrary real numbers
- Data transformation: Represent each vertex by a vector of its corresponding components in the eigenvectors
  - In the special case, the representations of vertices from the same connected component are equal, e.g. v_1, v_2, v_3 are transformed to $(1, 0, 0)$
  - In general case only *similar* eigenvector representations
- Clustering (e.g. k-Means) on transformed data points yields final result



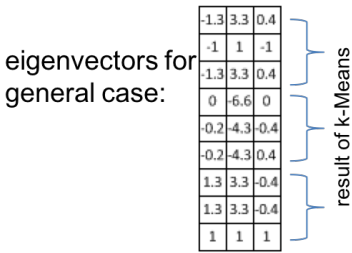
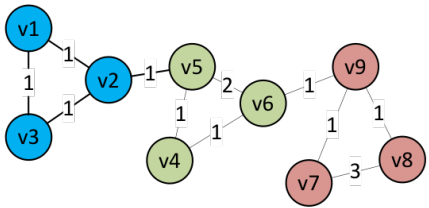
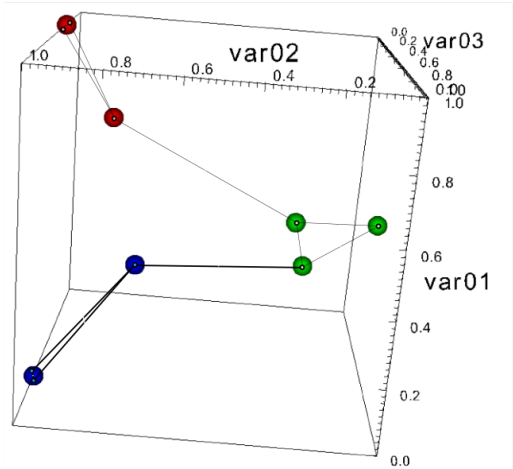


Illustration: Embedding of Vertices to a Vector Space

Spectral layout of previous example





Spectral Clustering: Discussion

Advantages

- No assumptions on the shape of the clusters
- Easy to implement

Disadvantages

- May be sensitive to construction of the similarity graph
- Runtime: k smallest eigenvectors can be computed in $\mathcal{O}(n^3)$ (worst case)
  - However: Much faster on sparse graphs, faster variants have been developed

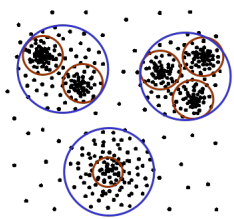
- Several variations of spectral clustering exist, using different Laplacian matrices which can be related to different graph cut problems [1]

[1] Von Luxburg, U.: A tutorial on spectral clustering, in Statistics and Computing, 2007

Agenda

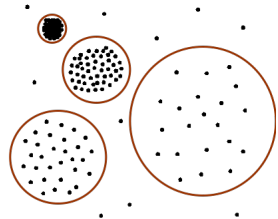
From Partitioning to Hierarchical Clustering

Global parameters to separate all clusters with a partitioning clustering method may not exist:



hierarchical cluster structure

and/or

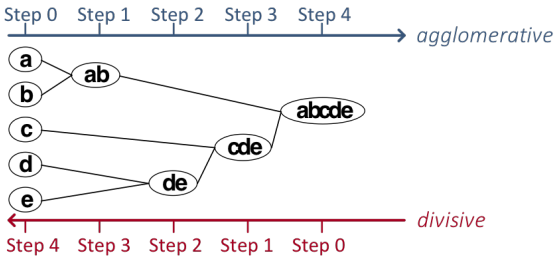


largely differing densities and sizes

Need a hierarchical clustering algorithm in these situations

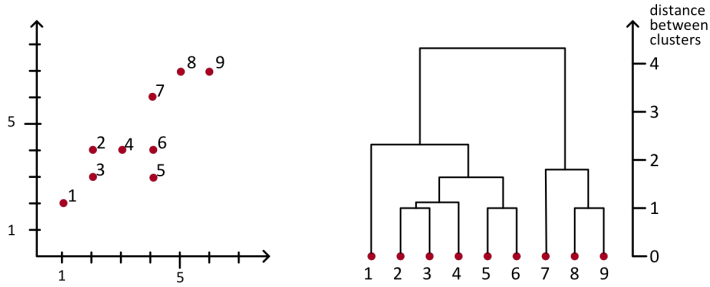
Hierarchical Clustering: Basic Notions

- Hierarchical decomposition of the data set (with respect to a given similarity measure) into a set of nested clusters
- Result represented by a so called *dendrogram* (greek $\delta\epsilon\nu\delta\rho o$ = tree)
  - Nodes in the dendrogram represent possible clusters
  - Dendrogram can be constructed bottom-up (agglomerative approach) or top down (divisive approach)



Hierarchical Clustering: Example

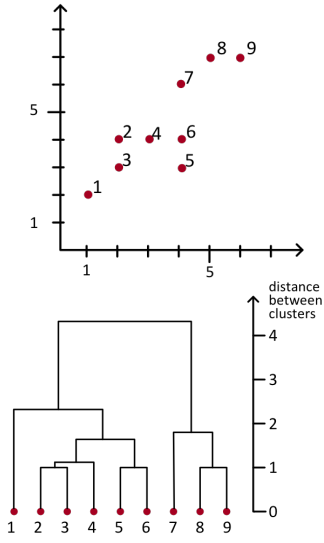
- Interpretation of the dendrogram
  - The root represents the whole data set
  - A leaf represents a single object in the data set
  - An internal node represents the union of all objects in its sub-tree
  - The height of an internal node represents the distance between its two child nodes



Agglomerative Hierarchical Clustering

Generic Algorithm

1. Initially, each object forms its own cluster
2. Consider all pairwise distances between the initial clusters (objects)
3. Merge the closest pair (A, B) in the set of the current clusters into a new cluster $C = A \cup B$
4. Remove A and B from the set of current clusters; insert C into the set of current clusters
5. If the set of current clusters contains only C (i.e., if C represents all objects from the database): STOP
6. Else: determine the distance between the new cluster C and all other clusters in the set of current clusters and go to step 3.



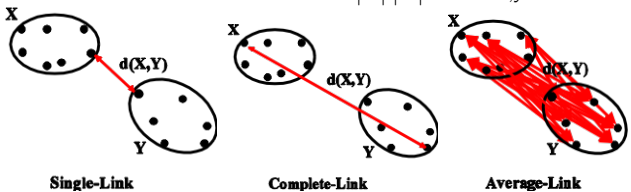
Single-Link Method and Variants

- Agglomerative hierarchical clustering requires a distance function for clusters
- Given: a distance function $dist(p, q)$ for database objects
- The following distance functions for clusters (i.e., sets of objects) X and Y are commonly used for hierarchical clustering:

Single-Link: $dist_{sl}(X, Y) = \min_{x \in X, y \in Y} dist(x, y)$

Complete-Link: $dist_{cl}(X, Y) = \max_{x \in X, y \in Y} dist(x, y)$

Average-Link: $dist_{al}(X, Y) = \frac{1}{|X| \cdot |Y|} \sum_{x \in X, y \in Y} dist(x, y)$



Divisive Hierarchical Clustering

General Approach: Top Down

- Initially, all objects form one cluster
- Repeat until all clusters are singletons
  - Choose a cluster to split $\rightarrow$ *how?*
  - Replace the chosen cluster with the sub-clusters and split into two $\rightarrow$ *how to split?*

Example solution: DIANA

- Select the cluster C with largest diameter for splitting
- Search the most disparate object o in C (highest average dissimilarity)
  - Splinter group $S = \{o\}$
  - Iteratively assign the $o' \notin S$ with the highest $D(o') > 0$ to the splinter group until $D(o') \leq 0$ for all $o' \notin S$, where

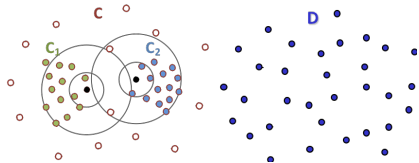
$$D(o') = \sum_{o_j \in C \setminus S} \frac{d(o', o_j)}{|C \setminus S|} - \sum_{o_i \in S} \frac{d(o', o_i)}{|S|}$$

Discussion Agglomerative vs. Divisive HC

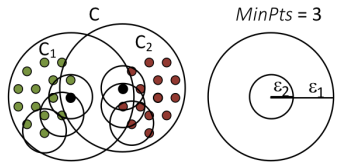
- Divisive and Agglomerative HC need $n-1$ steps
  - Agglomerative HC has to consider $\frac{n(n-1)}{2} = \binom{n}{2}$ combinations in the first step
  - Divisive HC potentially has $2^{n-1}-1$ many possibilities to split the data in its first step. Not every possibility has to be considered (DIANA)
- Divisive HC is conceptually more complex since it needs a second "flat" clustering algorithm (splitting procedure)
- Agglomerative HC decides based on local patterns
- Divisive HC uses complete information about the global data distribution $\rightsquigarrow$ able to provide better clusterings than Agglomerative HC?

Density-Based Hierarchical Clustering

- *Observation:* Dense clusters are completely contained by less dense clusters



- *Idea:* Process objects in the "right" order and keep track of point density in their neighborhood



Core Distance and Reachability Distance

Parameters: "generating" distance ϵ, fixed value *MinPts*

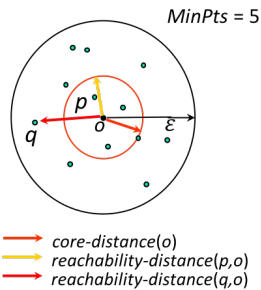
core-dist$_{\epsilon,MinPts}(o)$

- "smallest distance such that o is a core object"
- if core-dist $> \epsilon$: *undefined*

reach-dist$_{\epsilon,MinPts}(p, o)$

- "smallest dist. s.t. p is directly density-reachable from o"
- if reach-dist $> \epsilon$: ∞

$$\text{reach-dist}(p,o) = \begin{cases} dist(p,o) & , dist(p,o) \geq \text{core-dist}(o) \\ \text{core-dist}(o) & , dist(p,o) < \text{core-dist}(o) \\ \infty & , dist(p,o) > \epsilon \end{cases}$$

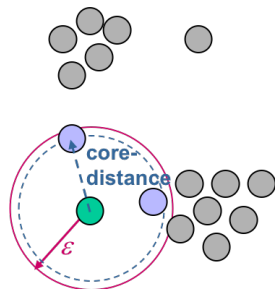


The Algorithm OPTICS

OPTICS[1]: Main Idea

"Ordering Points To Identify the Clustering Structure"

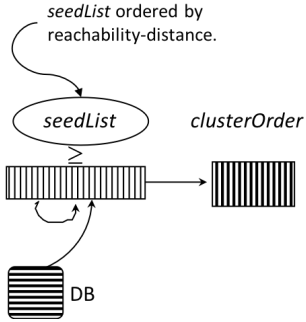
- Maintain two data structures
  - *seedList*: Stores all objects with shortest reachability distance seen so far ("distance of a jump to that point") in ascending order; organized as a heap
  - *clusterOrder*: Resulting cluster order is constructed sequentially (order of objects + reachability-distances)
- Visit each point
  - Always make a shortest jump



[1] Ankerst M., Breunig M., Kriegel H.-P., Sander J. "OPTICS: Ordering Points To Identify the Clustering Structure". SIGMOD (1999)

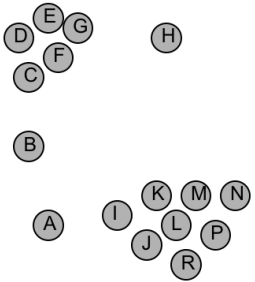
The Algorithm OPTICS

1: $seedList = \emptyset$
2: **while** there are unprocessed objects in DB **do**
3: **if** $seedList = \emptyset$ **then**
4: insert arbitrary unprocessed object into *clusterOrder* with reach-dist $= \infty$
5: **else**
6: remove first object from *seedList* and insert into *clusterOrder* with its current reach-dist
7: // *Let o be the last object inserted into clusterOrder*
8: mark o as processed
9: **for** $p \in range(o, \epsilon)$ **do**
10: // Insert/update p in *seedList*
11: compute reach-dist(p, o)
12: *seedList*.update$(p,$ reach-dist$(p, o))$



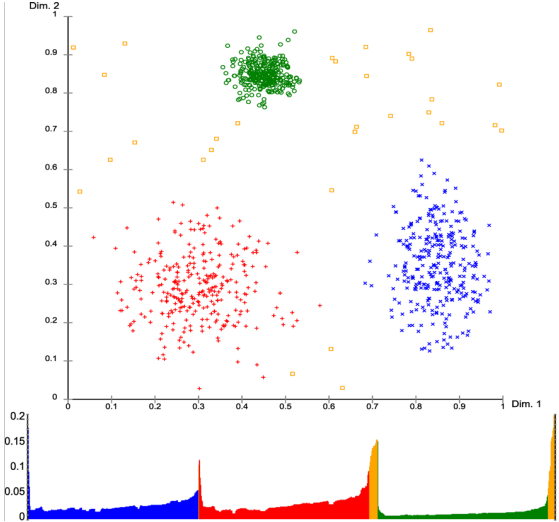
OPTICS: Example

$\epsilon = 44, MinPts = 3$



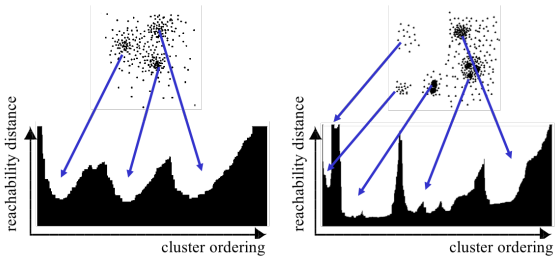
seed list:

OPTICS: The Reachability Plot



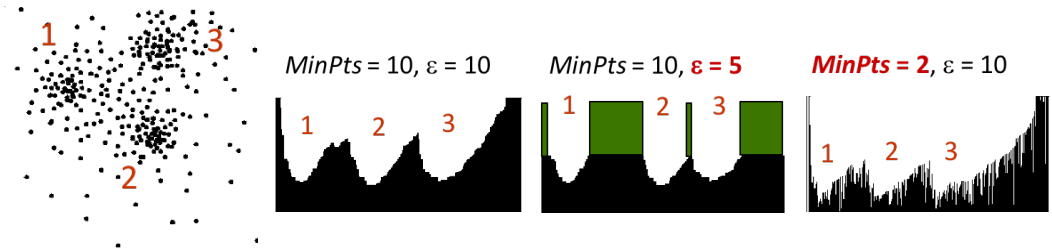
OPTICS: The Reachability Plot

- Plot the points together with their reachability-distances. Use the order in which they where returned by the algorithm
  - Represents the density-based clustering structure
  - Easy to analyze
  - Independent of the dimensionality of the data



OPTICS: Parameter Sensitivity

- Relatively insensitive to parameter settings
- Good result if parameters are just "large enough"



Hierarchical Clustering: Discussion

Advantages

- Does not require the number of clusters to be known in advance
- No (standard methods) or very robust parameters (OPTICS)
- Computes a complete hierarchy of clusters
- Good result visualizations integrated into the methods
- A "flat" partition can be derived afterwards (e.g. via a cut through the dendrogram or the reachability plot)

Disadvantages

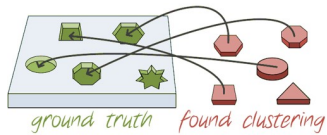
- May not scale well
  - Runtime for the standard methods: $\mathcal{O}(n^2 \log n^2)$
  - Runtime for OPTICS: without index support $\mathcal{O}(n^2)$
- User has to choose the final clustering

Agenda

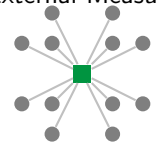
Evaluation of Clustering Results

Type	Positive	Negative
Expert's Opinion	may reveal new insight into the data	very expensive, results are not comparable
External Measures	objective evaluation	needs "ground truth"
Internal Measures	no additional information needed	approaches optimizing the evaluation criteria will always be preferred

Expert's Opinion



External Measure



Internal Measure

External Measures

Notation

Given a data set D, a clustering $\mathcal{C} = \{C_1, \ldots, C_k\}$ and ground truth $\mathcal{G} = \{G_1, \ldots, G_l\}$.

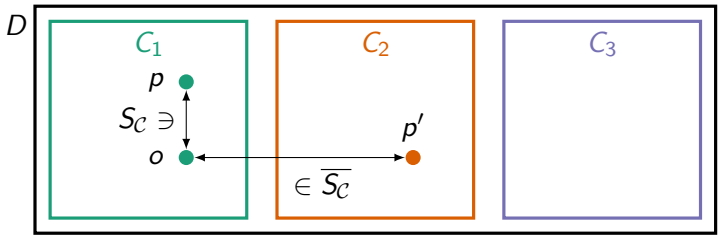
Problem

Since the cluster labels are "artificial", permuting them should not change the score.

Solution

Instead of comparing cluster and ground truth labels directly, consider all pairs of objects. Check whether they have the same label in $\mathcal{G}$ and if they have the same in $\mathcal{C}$.

Formalisation as Retrieval Problem



With $P = \{(o, p) \in D \times D \mid o \neq p\}$ define:

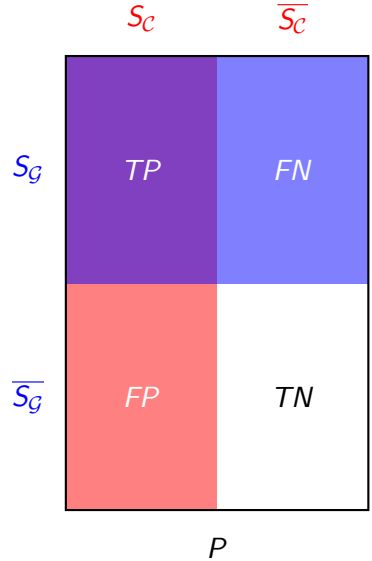
- *Same* cluster label: $S_C = \{(o, p) \in P \mid \exists C_i \in C : \{o, p\} \subseteq C_i\}$
- *Different* cluster label: $\overline{S_C} = P \setminus S_C$

and analogously for $\mathcal{G}$.

Formalisation as Retrieval Problem

Define

- $TP = |S_{\mathcal{C}} \cap S_{\mathcal{G}}|$
  (same cluster in both, "true positives")
- $FP = |S_{\mathcal{C}} \cap \overline{S_{\mathcal{G}}}|$
  (same cluster in $\mathcal{C}$, different cluster in $\mathcal{G}$, "false positives")
- $TN = |\overline{S_{\mathcal{C}}} \cap \overline{S_{\mathcal{G}}}|$
  (different cluster in both, "true negatives")
- $FN = |\overline{S_{\mathcal{C}}} \cap S_{\mathcal{G}}|$
  (different cluster in $\mathcal{C}$, same cluster in $\mathcal{G}$, "false negatives")



External Measures

- *Recall* ($0 \leq rec \leq 1$, larger is better)

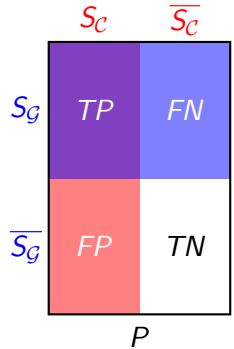
$$rec = \frac{TP}{TP + FN} = \frac{|S_{\mathcal{C}} \cap S_{\mathcal{G}}|}{|S_{\mathcal{G}}|}$$

- *Precision* ($0 \leq prec \leq 1$, larger is better)

$$prec = \frac{TP}{TP + FP} = \frac{|S_{\mathcal{C}} \cap S_{\mathcal{G}}|}{|S_{\mathcal{C}}|}$$

- F_1-*Measure* ($0 \leq F_1 \leq 1$, larger is better)

$$F_1 = \frac{2 \cdot rec \cdot prec}{rec + prec} = \frac{2|S_{\mathcal{C}} \cap S_{\mathcal{G}}|}{|S_{\mathcal{C}}| + |S_{\mathcal{G}}|}$$



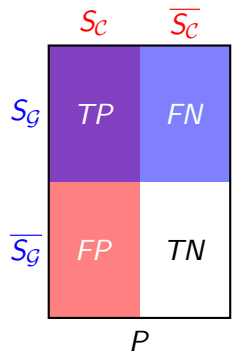
External Measures

- *Rand Index* ($0 \leq RI \leq 1$, larger is better):

$$RI(\mathcal{C} \mid \mathcal{G}) = \frac{TP + TN}{TP + TN + FP + FN} = \frac{|S_{\mathcal{C}} \cap S_{\mathcal{G}}| + |\overline{S_{\mathcal{C}}} \cap \overline{S_{\mathcal{G}}}|}{|P|}$$

- *Adjusted Rand Index* (ARI): Compares $RI(\mathcal{C}, \mathcal{G})$ against expected $(\mathcal{R}, \mathcal{G})$ of random cluster assignment $\mathcal{R}$.
- *Jaccard Coefficient* ($0 \leq JC \leq 1$, larger is better):

$$JC = \frac{TP}{TP + FP + FN} = \frac{|S_{\mathcal{C}} \cap S_{\mathcal{G}}|}{|P| - |\overline{S_{\mathcal{C}}} \cap \overline{S_{\mathcal{G}}}|}$$



External Measures

- Confusion Matrix / Contingency Table $N \in \mathbb{N}^{k \times l}$ with $N_{ij} = |C_i \cap G_j|$

	G_1	$\dots$	G_l
C_1	$\lvert C_1 \cap G_1 \rvert$	$\dots$	$\lvert C_1 \cap G_l \rvert$
$\vdots$	$\vdots$	$\ddots$	
C_k	$\lvert C_k \cap G_1 \rvert$		$\lvert C_k \cap G_l \rvert$

- Define $N_i = \sum_{j=1}^{l} N_{ij}$ (i.e. $N_i = |C_i|$)
- Define $N = \sum_{i=1}^{k} N_i$ (i.e. $N = |D|$)

External Measures

- (Shannon) Entropy:

$$H(\mathcal{C}) = -\sum_{C_i \in \mathcal{C}} p(C_i) \log p(C_i) = -\sum_{C_i \in \mathcal{C}} \frac{|C_i|}{|D|} \log \frac{|C_i|}{|D|} = -\sum_{i=1}^{k} \frac{N_i}{N} \log \frac{N_i}{N}$$

- Mutual Entropy:

$$\begin{aligned} H(\mathcal{C} \mid \mathcal{G}) &= -\sum_{C_i \in \mathcal{C}} p(C_i) \sum_{G_j \in \mathcal{G}} p(G_j \mid C_i) \log p(G_j \mid C_i) \\ &= -\sum_{C_i \in \mathcal{C}} \frac{|C_i|}{|D|} \sum_{G_j \in \mathcal{G}} \frac{|C_i \cap G_j|}{|C_i|} \log \frac{|C_i \cap G_j|}{|C_i|} \\ &= -\sum_{i=1}^{k} \frac{N_i}{N} \sum_{j=1}^{l} \frac{N_{ij}}{N_i} \log \frac{N_{ij}}{N_i} \end{aligned}$$

External Measures

- Mutual Information:

$$I(\mathcal{C}, \mathcal{G}) = H(\mathcal{C}) - H(\mathcal{C} \mid \mathcal{G}) = H(\mathcal{G}) - H(\mathcal{G} \mid \mathcal{C})$$

- Normalized Mutual Information (NMI) ($0 \leq NMI \leq 1$, larger is better):

$$NMI(\mathcal{C}, \mathcal{G}) = \frac{I(\mathcal{C}, \mathcal{G})}{\sqrt{H(\mathcal{C})H(\mathcal{G})}}$$

- Adjusted Mutual Information (AMI): Compares $MI(\mathcal{C}, \mathcal{G})$ against expected $MI(\mathcal{R}, \mathcal{G})$ of random cluster assignment $\mathcal{R}$.

Internal Measures: Cohesion

Notation

Let D be a set of size $n = |D|$, and let $\mathcal{C} = \{C_1, \ldots, C_k\}$ be a partitioning of D.

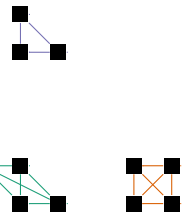
Cohesion

Average distance between objects of the same cluster.

$$coh(C_i) = \binom{|C_i|}{2}^{-1} \sum_{o,p \in C_i, o \neq p} d(o,p)$$

Cohesion of clustering is equal to weighted mean of the clusters' cohesions.

$$coh(\mathcal{C}) = \sum_{i=1}^{k} \frac{|C_i|}{n} coh(C_i)$$



Internal Measures: Separation

Separation

Separation between to clusters: Average distance between pairs

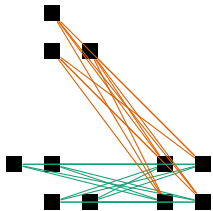
$$sep(C_i, C_j) = \frac{1}{|C_i||C_j|} \sum_{o \in C_i, p \in C_j} d(o, p)$$

Separation of one cluster: Minimum separation to another cluster:

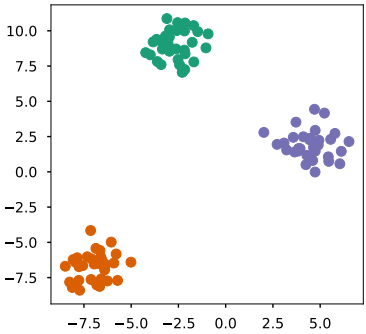
$$sep(C_i) = \min_{j \neq i} sep(C_i, C_j)$$

Separation of clustering is equal to weighted mean of the clusters' separations.

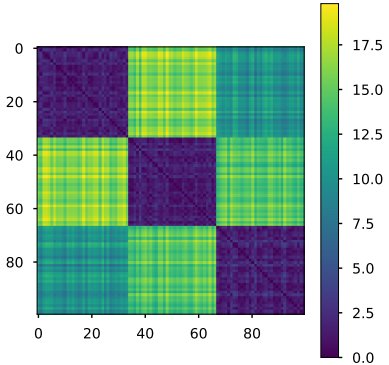
$$sep(\mathcal{C}) = \sum_{i=1}^{k} \frac{|C_i|}{n} sep(C_i)$$



Evaluating the Distance Matrix



dataset
(well separated)

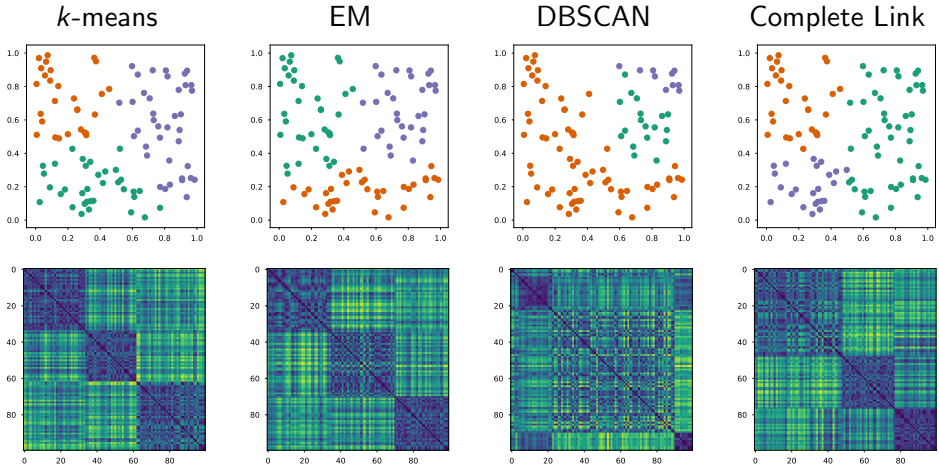


Distance matrix
(sorted by k-means cluster label)

after: Tan, Steinbach, Kumar: Introduction to Data Mining (Pearson, 2006)

Evaluating the Distance Matrix

Distance matrices differ for different clustering approaches (here on random data)

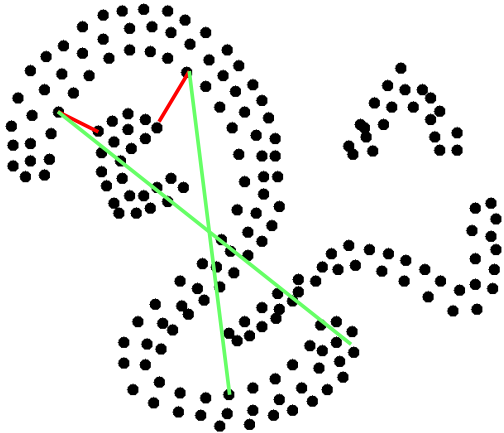


after: Tan, Steinbach, Kumar: Introduction to Data Mining (Pearson, 2006)

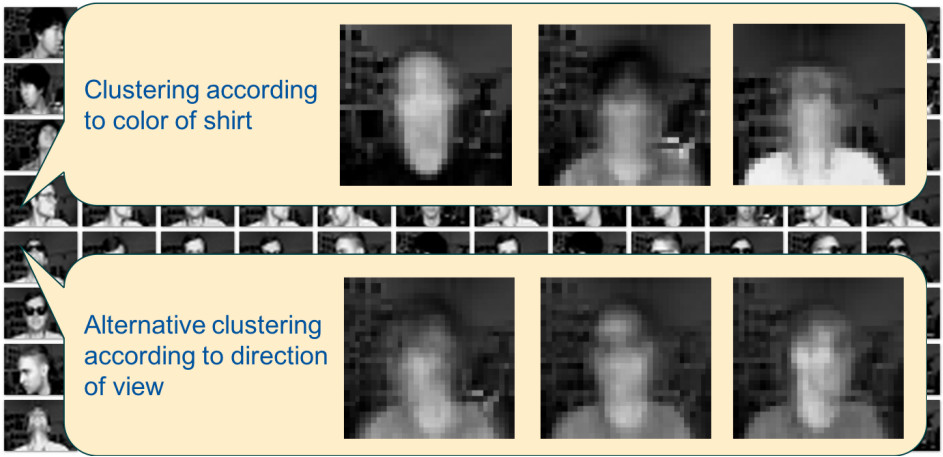
Cohesion and Separation

Problem

Suitable for convex cluster, but not for stretched clusters (cf. silhouette coefficient).



Ambiguity of Clusterings



- Clustering according to: Color of shirt, direction of view, glasses, ...

Ambiguity of Clusterings

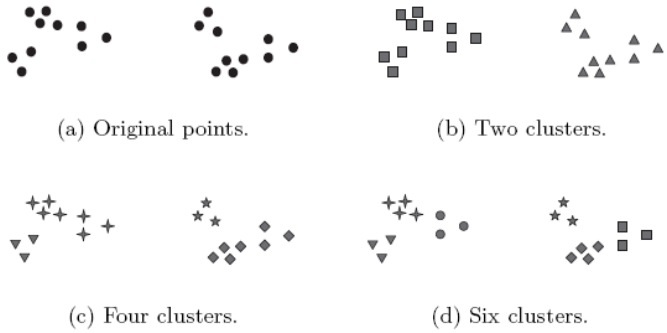


Figure 8.1. Different ways of clustering the same set of points.

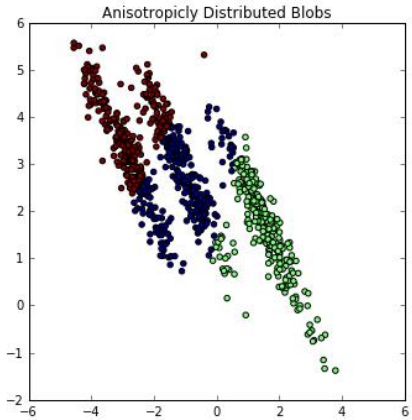
from: Tan, Steinbach, Kumar: Introduction to Data Mining (Pearson, 2006)

Ambiguity of Clusterings

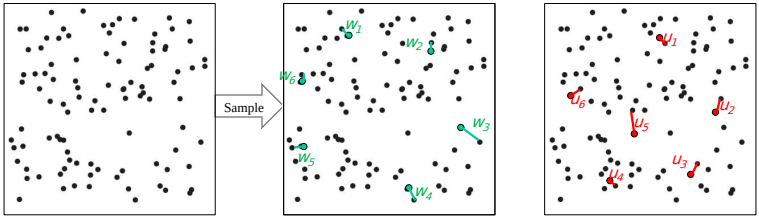
"Philosophical" Problem

"What is a correct clustering?"

- Most approaches find clusters in every dataset, even in uniformly distributed objects
- Are there clusters?
  - Apply clustering algorithm
  - Check for reasonability of clusters
- Problem: No clusters found $\neq$ no clusters existing
  - Maybe clusters exists only in certain models, but can not be found by used clustering approach



Hopkins Statistics

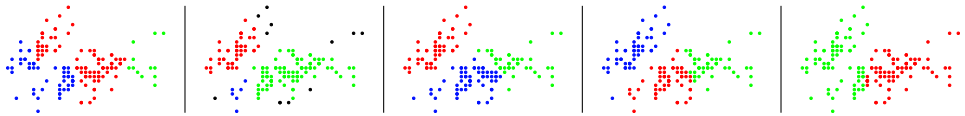


$$H = \frac{\sum_{i=1}^{m} u_i}{\sum_{i=1}^{m} u_i + \sum_{i=1}^{m} w_i}$$

- w_i: distance of selected objects to the next neighbor in dataset
- u_i: distances of uniformly distributed objects to next neighbor in dataset
- $0 \leq H \leq 1$;
  - $H \approx 0$: very regular data (e.g. grid);
  - $H \approx 0.5$: uniformly distributed data;
  - $H \approx 1$: strongly clustered

Agenda

Ensemble Clustering



Problem

- Many differing clustering models
- Different parameter choices, usually highly influences the result

What is a "good" clustering?

Idea

Find a consensus solution (also ensemble clustering) that consolidates multiple clustering solutions.

Ensemble Clustering: Benefits

- *Knowledge Reuse*: Possibility to integrate the knowledge of multiple known, good clusterings
- *Improved Quality*: Often ensemble clustering leads to "better" results than its individual base solutions.
- *Improved Robustness*: Combining several clustering approaches with differing data modeling assumptions leads to an increased robustness across a wide range of datasets.
- *Model Selection*: Novel approach for determining the final number of clusters
- *Distributed Clustering*: if data is inherently distributed (either feature-wise or object-wise) and each clusterer has only access to a subset of objects and/or features, ensemble methods can be used to compute a unifying result

Ensemble Clustering: Basic Notions

Given

A set of L clusterings $\mathfrak{C} = \mathcal{C}_1, \ldots, \mathcal{C}_L$ for dataset $D = \{x_1, \ldots, x_n\} \in \mathbb{R}^d$.

Goal

Find a consensus clustering $\mathcal{C}^*$.

How to define a consensus clustering?

Two categories:

- Approaches based on pairwise similarity: Find a consensus clustering $\mathcal{C}^*$ for which the similarity function $\Phi(\mathfrak{C}, \mathcal{C}^*) = \sum_{\mathcal{C} \in \mathfrak{C}} \phi(\mathcal{C}, \mathcal{C}^*)$ (ϕ is basically an external measure)
- Probabilistic approaches: Assume that the L labels for the objects $x_i \in D$ follow a certain distribution.

Similarity-Based Approaches

Goal

Find a consensus clustering $\mathcal{C}^*$ for which the similarity function $\Phi(\mathfrak{C}, \mathcal{C}^*) = \sum_{\mathcal{C} \in \mathfrak{C}} \phi(\mathcal{C}, \mathcal{C}^*)$ is maximal.

Choices for ϕ

- Pair counting-based measures: Rand Index (RI), Adjusted RI, Probabilistic RI
- Information theoretic measures: Mutual Information (I), Normalized Mutual Information (NMI), Variation of Information (VI)

Problem

Minimising the objective for the above mentioned choices of ϕ in intractable.

Similarity-Based Approaches

Solutions

- Methods based on the co-association matrix (related to RI)
- Methods using cluster labels without co-association matrix (often related to NMI)
  - Mostly graph partitioning
  - Cumulative voting

Ensemble Clustering: Co-Association Matrix

Co-Association Matrix

The *co-association matrix* $S^{\mathfrak{C}} \in \mathbb{R}^{n \times n}$ represents the label similarity of object pairs:

$$S_{ij}^{\mathfrak{C}} = \sum_{\mathcal{C} \in \mathfrak{C}} \mathbb{I}[x_i \in \mathcal{C} \wedge x_j \in \mathcal{C}]$$

where $\mathbb{I}$ is the indicator function with $\mathbb{I}[\mathit{False}] = 0$, and $\mathbb{I}[\mathit{True}] = 1$.

Example

$D = \{1, 2, 3, 4, 5\}$ (i.e. $n = 5$),
$\mathfrak{C} = \{\mathcal{C}_1, \mathcal{C}_2\}$,
$\mathcal{C}_1 = \{\{1, 2, 3\}, \{4, 5\}\}$,
$\mathcal{C}_2 = \{\{1, 2\}, \{3, 4, 5\}\}$.

$$S = \begin{pmatrix} 2 & 2 & 1 & 0 & 0 \\ 2 & 2 & 1 & 0 & 0 \\ 1 & 1 & 2 & 1 & 1 \\ 0 & 0 & 1 & 2 & 2 \\ 0 & 0 & 1 & 2 & 2 \end{pmatrix}$$

Ensemble Clustering: Co-Association Matrix

Usage of Co-Association Matrix

- Use $S^{\mathfrak{C}}$ as similarity matrix to apply traditional clustering approach.
- By interpreting $S^{\mathfrak{C}}$ as weighted adjacency matrix, graph partitioning methods can be applied.

Co-Association Matrix and Rand Index

In [16] a connection of consensus clustering based on the co-association matrix and the optimization of the pairwise similarity based on the Rand Index has been proven:

$$C_{best} = \underset{\mathcal{C}^*}{\operatorname{argmax}} \sum_{\mathcal{C} \in \mathfrak{C}} RI(\mathcal{C}, \mathcal{C}^*)$$

[16] B. Mirkin: *Mathematical Classification and Clustering*. Kluwer, 1996.

Information-Theoretic Approaches

Setting

Find a consensus clustering $\mathcal{C}^*$ for which the similarity function $\Phi(\mathfrak{C}, \mathcal{C}^*) = \sum_{\mathcal{C} \in \mathfrak{C}} \phi(\mathcal{C}, \mathcal{C}^*)$ is maximal, with ϕ chosen as (Normalised) Mutual Information.

Problem

Usually a hard optimization problem!

Solution 1

Use meaningful optimization approaches (e.g. gradient descent) or heuristics to approximate the best clustering solution (e.g. [17])

[17] A. Strehl, J. Ghosh: *Cluster ensembles - a knowledge reuse framework for combining multiple partitions.* Journal of Machine Learning Research, 3, 2002, pp. 583-617.

Information-Theoretic Approaches

Solution 2

- Use a similar but solvable objective, e.g. [18]:
- Use as objective

$$C_{best} = \underset{\mathcal{C}^*}{\operatorname{argmax}} \sum_{\mathcal{C} \in \mathfrak{C}} I^s(\mathcal{C}, \mathcal{C}^*)$$

where I^s is the mutual information based on the generalized entropy of degree s:

$$H^s(X) = (2^{1-s} - 1)^{-1} \sum_{x_i \in X} (p_i^s - 1)$$

For $s = 2$, $I^s(\mathcal{C}, \mathcal{C}^*)$ is equal to the category utility function whose maximization is proven to be equivalent to the minimization of the square-error clustering criterion. $\implies$ Apply a simple label transformation and use e.g. K-Means

[18] A. Topchy, A.K. Jain, W. Punch. *Combining multiple weak clusterings*. In ICDM, pages 331-339, 2003

Probabilistic Approach

Assumptions

- All clusterings $\mathcal{C} \in \mathfrak{C}$ are partitionings of the dataset D.
- There are K^* consensus clusters.
- With $\mathcal{C}(x)$ denoting the cluster label assigned to x in clustering $\mathcal{C}$, the following dataset Y given by

$$Y = \{y_i \in \mathbb{N}_0^L \mid x_i \in D, \forall 1 \leq j \leq L : (y_i)_j = \mathcal{C}_i(x_i)\}$$

(labels of base clusterings) follows a multivariate mixture distribution:

$$p(Y \mid \Theta) = \prod_{i=1}^{n} \sum_{k=1}^{K^*} \alpha_k p_k(y_i \mid \theta_k) \overset{cond.ind.}{=} \prod_{i=1}^{n} \sum_{k=1}^{K^*} \alpha_k \prod_{j=1}^{L} p_{kl}(y_{ij} \mid \theta_{kl})$$

with $p_{kl}(y_{ij} \mid \theta_{kl}) \sim M(1, (p_{kl1}, \ldots, p_{kl|\mathcal{C}_l|}))$, i.e. $p_{kl}(y_{ij} \mid \theta_{kl}) = \prod_{k'=1}^{|\mathcal{C}_l|} p_{klk'}^{\mathbb{I}(y_{nl}=k')}$

Probabilistic Approach

Goal

Find the parameters $\Theta = (\alpha_1, \theta_1, \ldots, \alpha_{K*}, \theta_{K*})$ such that the likelihood $p(Y \mid \Theta)$ is maximized.

Solution [19]

Optimize the parameters via the EM approach

[19] Topchy, Jain, Punch: *A mixture model for clustering ensembles*. In ICDM, pp. 379-390, 2004.

Agenda