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

Knowledge Discovery and Data Mining 1

(Data Mining Algorithms 1)

Winter Semester 2019/20

Agenda

Agenda

Supervised vs. Unsupervised Learning

Unsupervised Learning (clustering)

- The class labels of training data are unknown
- Given a set of measurements, observations, etc. with the aim of establishing the existence of classes or clusters in the data
  - Classes (=clusters) are to be determined

Supervised Learning (classification)

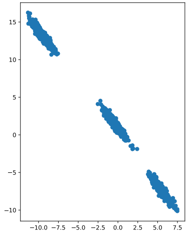
- Supervision: The training data (observations, measurements, etc.) are accompanied by labels indicating the class of the observations
  - Classes are known in advance (a priori)
- New data is classified based on information extracted from the training set

What is Clustering?

Clustering

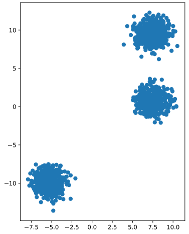
Grouping a set of data objects into clusters (=collections of data objects).

- *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

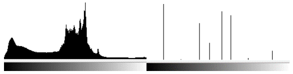


General Applications of Clustering

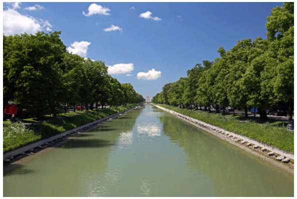
- Preprocessing – as a data reduction (instead of sampling)
  - Image data bases (color histograms for filter distances)
  - Stream clustering (handle endless data sets for offline clustering)
- Pattern Recognition and Image Processing
- Spatial Data Analysis:
  - create thematic maps in Geographic Information 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

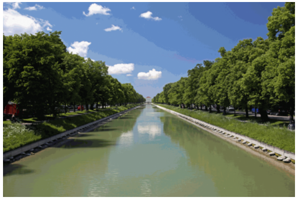
- Reassign color values to k distinct colors
- Cluster pixels using color difference, not spatial data



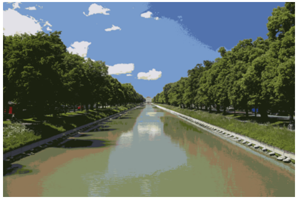
65536



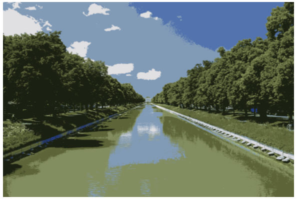
256



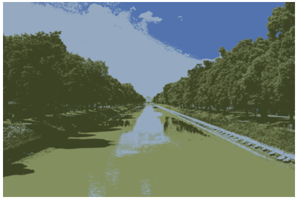
16



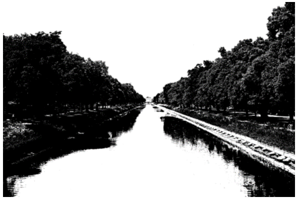
8



4

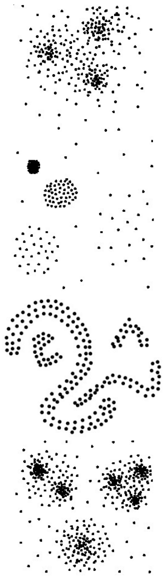


2



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

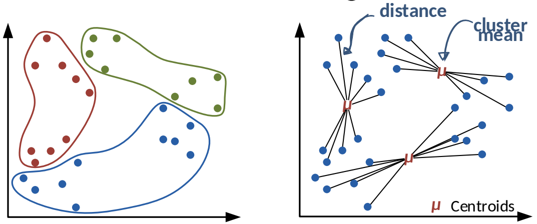
Idea[1]

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

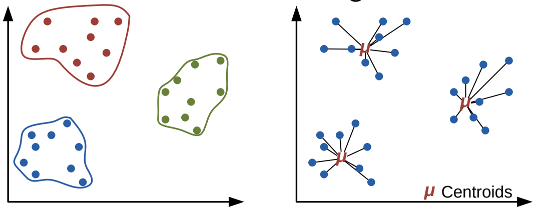
Objective

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

Poor clustering



Good clustering



[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

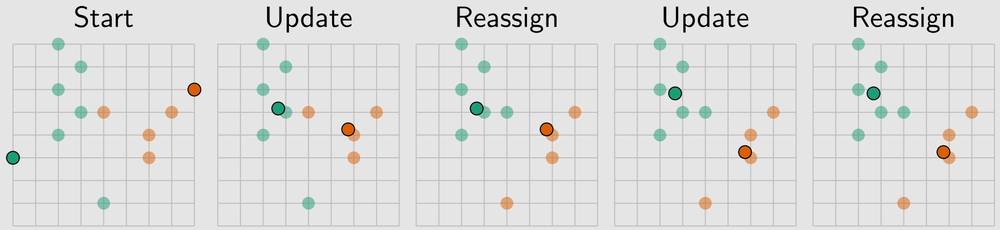
- Objects $p = (p_1, \ldots, p_d)$ are points in a d-dimensional vector space (the mean μ_S of a set of points S must be defined: $\mu_S = \frac{1}{|S|} \sum\limits_{p \in S} p$)
- Measure for the compactness of a *cluster* C_j (sum of squared distances): $SSE(C_j) = \sum\limits_{p \in C_j} ||p - \mu_{C_j}||_2^2$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



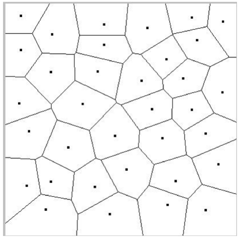
k-Means: Voronoi Model for Convex Cluster Regions

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)

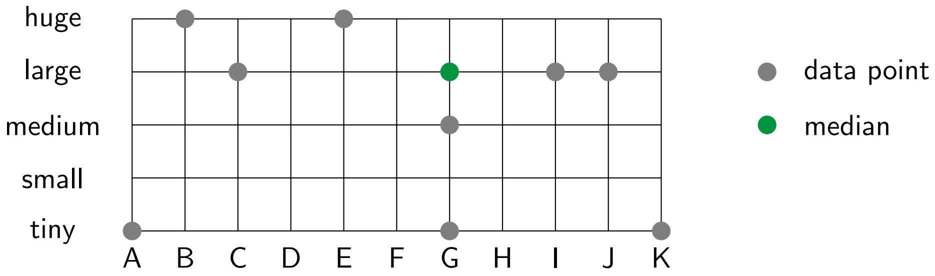
Alternatives for *Mean* representatives

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

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)

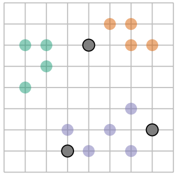
K-Means/Median/Mode/Medoid Clustering: Discussion

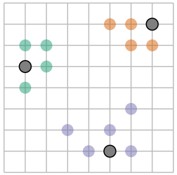
	k-**Means**	k-**Median**	k-**Mode**	k-**Medoid**
data	numerical (mean)	ordinal	categorical	metric
efficiency	high $\mathcal{O}(tkn)$			low $\mathcal{O}\left(tk(n-k)^2\right)$
sensitivity to outliers	high	low		

- 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 $k = 2, \dots, n-1$
  - 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]

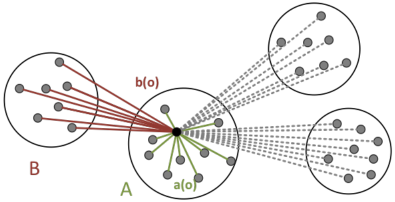
Quality measure for k-means or k-medoid clusterings that is not monotonic over k.

[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): $b(o)$



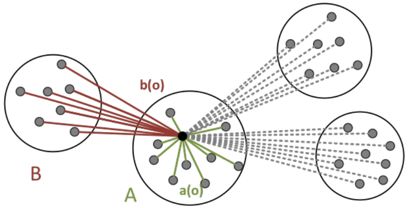
The Silhouette Coefficient

- $a(o)$ = "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

dataset with 10 clusters

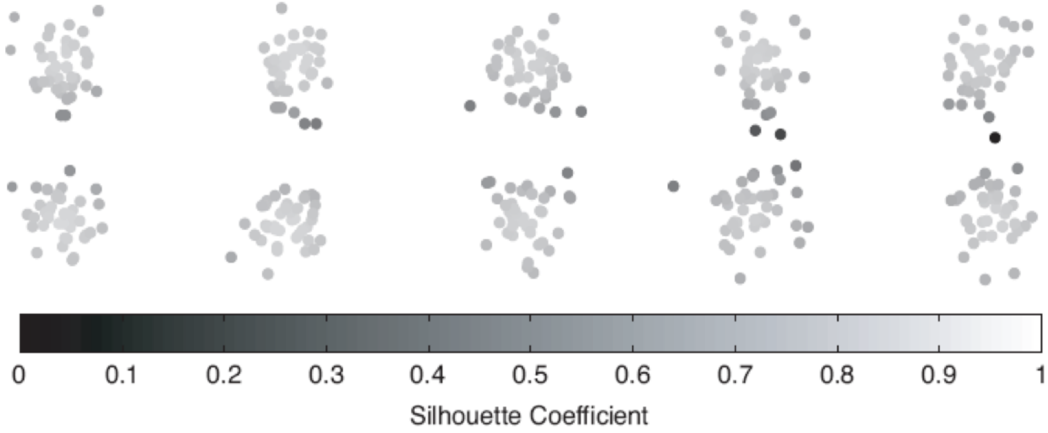
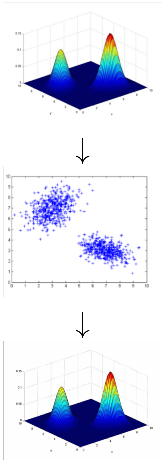


Image from Tan, Steinbach, Kumar: Introduction to Data Mining (Pearson, 2006)

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

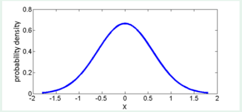
Excursus: Gaussian Mixture Distributions

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

Gaussian Distribution

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

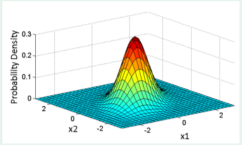
$$p(x \mid \mu, \sigma^2) = \mathcal{N}(x \mid \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(x-\mu)^2\right)$$



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

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

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



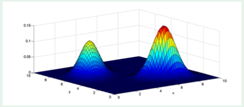
with *mean vector* $\mu \in \mathbb{R}^d$ and *covariance matrix* $\Sigma \in \mathbb{R}^{d \times d}$

Excursus: Gaussian Mixture Distributions

Gaussian mixture distribution with k components

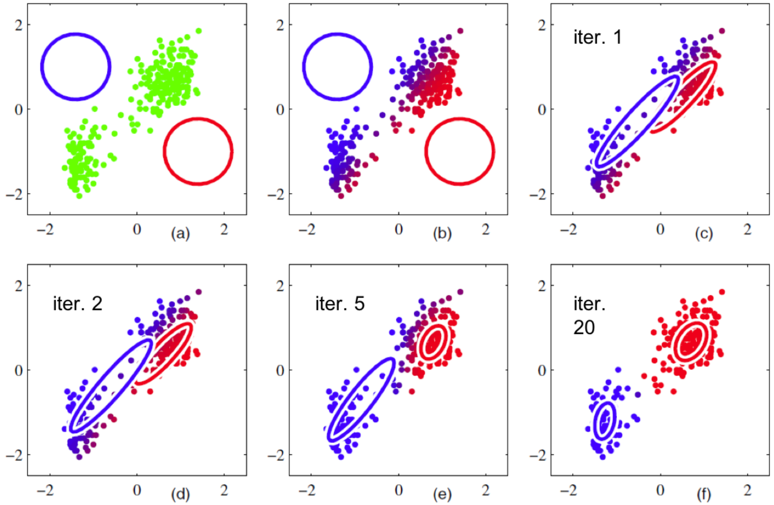
- For d-dimensional vector $x \in \mathbb{R}^d$:

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



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

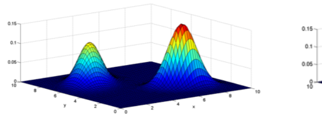
A clustering $\mathcal{M} = (C_1, \dots, C_k)$ is represented by a mixture distribution with parameters $\theta = (\pi_1, \mu_1, \Sigma_1, \dots, \pi_k, \mu_k, \Sigma_k)$:

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

Cluster

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

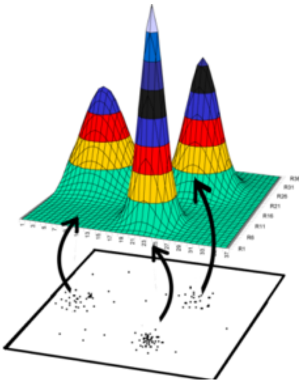
$$p(x \mid \mu_l, \Sigma_l) = \mathcal{N}(x \mid \mu_l, \Sigma_l)$$



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:

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

- The log-likelihood can be written as

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

- Maximization w.r.t. the means:

$$\frac{\partial \log p(X \mid \theta)}{\partial \mu_j} \overset{!}{=} 0$$

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

$$\vdots$$

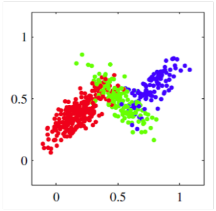
$$\Sigma_j^{new} = \frac{\sum_{i=1}^{n} \gamma_j^{new}(x_i)(x_i - \mu_j^{new})(x_i - \mu_j^{new})^T}{\sum_{i=1}^{n} \gamma_j^{new}(x_i)}$$

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

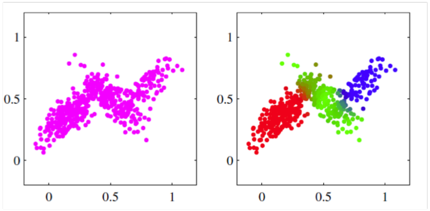
4. Evaluate the new log-likelihood $\log p(X \mid \theta^{new})$ and check for convergence of parameters or log-likelihood ($|\log p(X \mid \theta^{new}) - \log p(X \mid \theta)| \leq \epsilon$). If the convergence criterion is not satisfied, set $\theta = \theta^{new}$ and go to step 2.

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

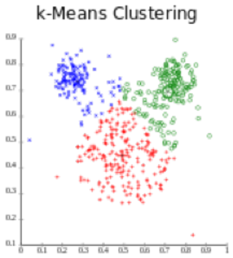
- Modification to obtain a partitioning variant: Assign each object to the cluster to which it belongs with the highest probability

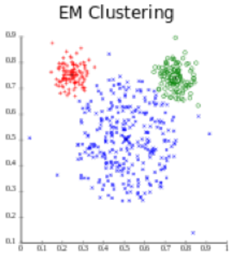
$$C(x_i) = \underset{l \in \{1,\ldots,k\}}{\operatorname{argmax}} \{\gamma_l(x_i)\}$$

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)

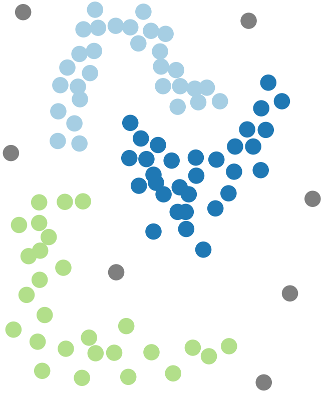
[1] G. McLachlan and D. Peel. Finite Mixture Models. Wiley, New York, 2000.

Agenda

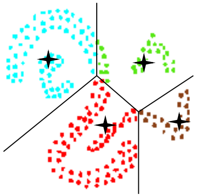
Density-Based Clustering

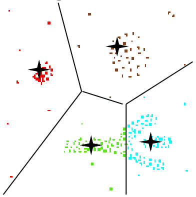
Basic Idea

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:

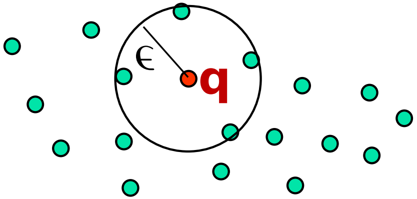
- ϵ-radius for the neighborhood of point q

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

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

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

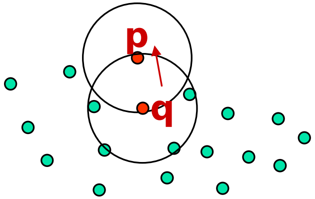
Density-Based Clustering: Basic Concept

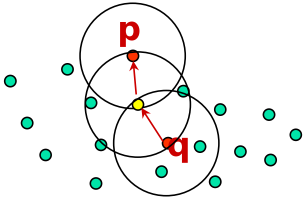


Core Point

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





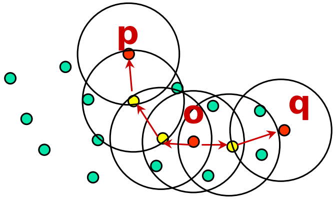
(Directly) Density-Reachable

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*

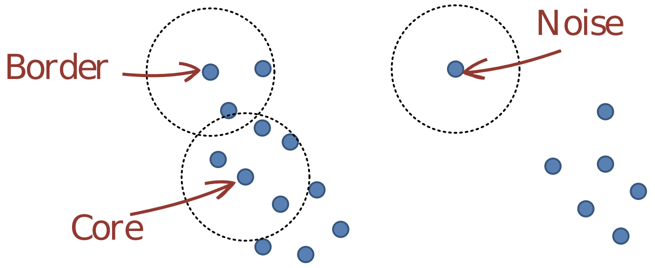
Density-Based Clustering: Basic Definitions

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[1]

```
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 $N$
```

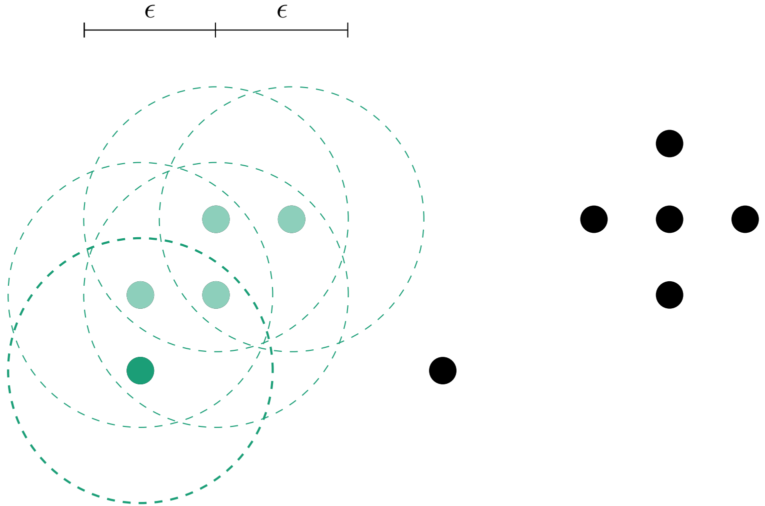
Note

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

[1] 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.

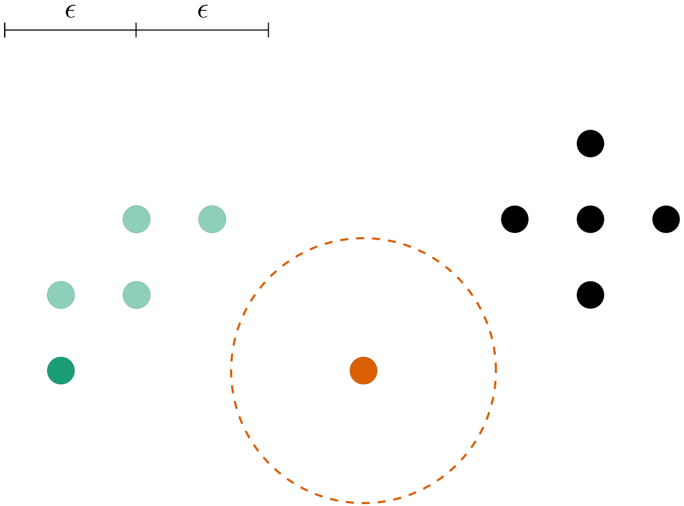
DBSCAN: Example

Parameters: $\epsilon = 1.75$, $minPts = 3$. Clusters: C_1



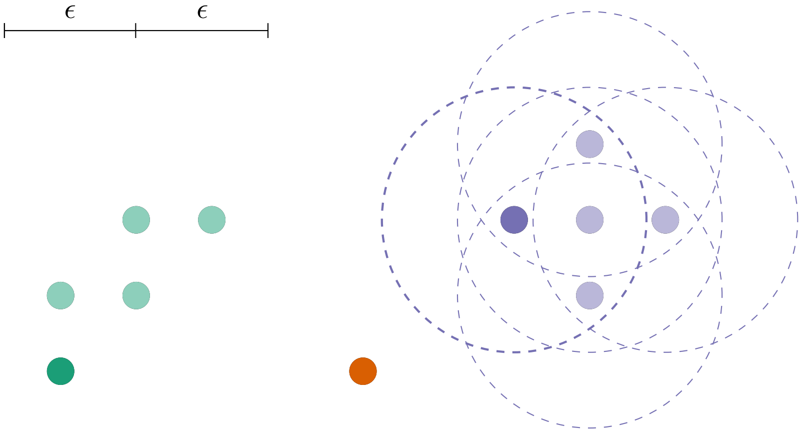
DBSCAN: Example

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



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*

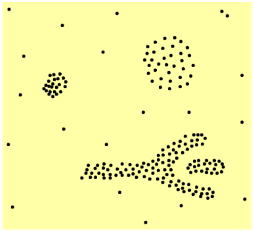
Idea

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

Problem

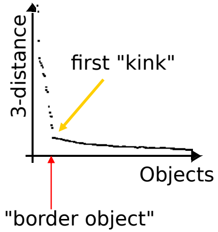
How to determine this?

Determining the Parameters ϵ and *MinPts*

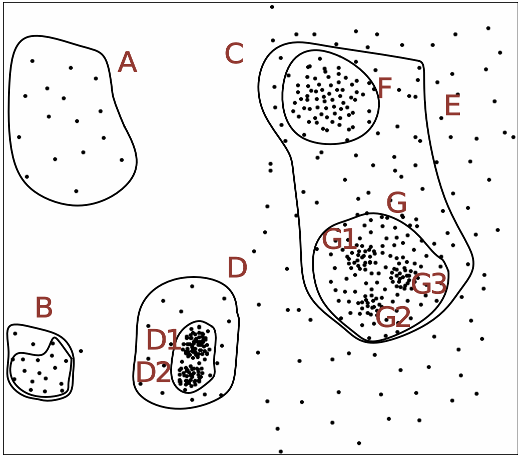


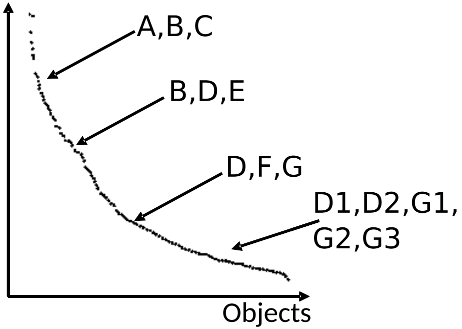
Heuristic

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.[2] 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:

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

SQL Query

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

[2] 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

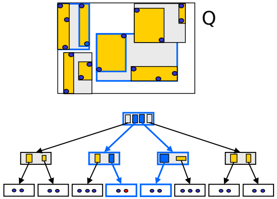
SELECT $*$ FROM $D\ q, D\ p$ WHERE $dist(q,p) \leq \epsilon$ GROUP BY $q.id$ HAVING $count(q.id) \geq \mu$

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

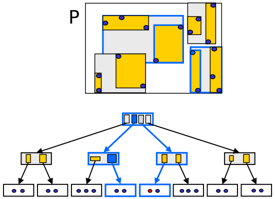
Efficient Similarity Join Processing

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.



$$Q \bowtie_{\epsilon} P$$



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

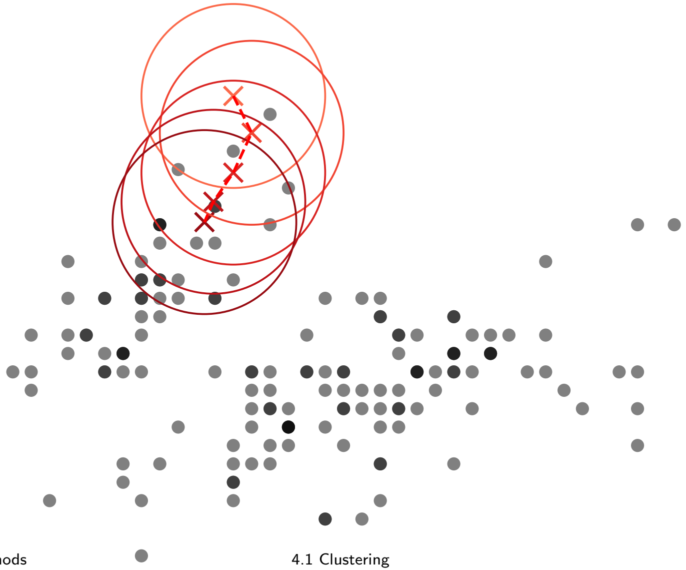
Find modes in the point density.

Algorithm[3]

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.

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

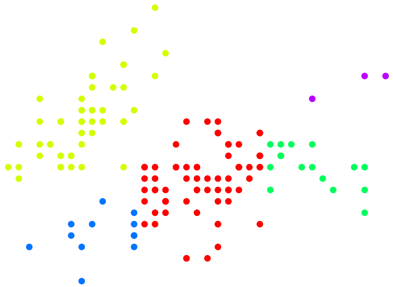
Iterative Mode Search: Example



Mean Shift: Core Algorithm

Algorithm[4]

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



[4] 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, with weights w_x, resp. calculated by some kernel κ:

$$m^{(i+1)} = \frac{\sum\limits_{x \in W(m^{(i)})} w_x \cdot x}{\sum\limits_{x \in W(m^{(i)})} w_x} \quad \rightarrow \quad m^{(i+1)} = \frac{\sum\limits_{x \in W(m^{(i)})} \kappa(x) \cdot x}{\sum\limits_{x \in W(m^{(i)})} \kappa(x)}$$

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: $\mathcal{O}(tn^2)$

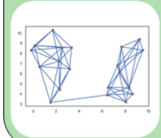
Advantages

- 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

General Steps for Spectral Clustering I

Construct Graph out of Data 1

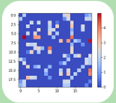


using:
- kNN
- ε-neighborhood
- fully-connected

graph

2

- (weighted) adjacency matrix W
- degree matrix D



- laplacian matrix L:

unnormalized (D − W)

normalized

3

problem to solve:

$$fLf^T = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2 = \min$$

Solution:

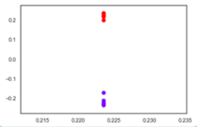
calculate eigenvalues λ ,
eigenvectors v of matrix L

General Steps for Spectral Clustering II

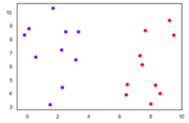
Choose usefull Number of Eigenvalues 4

- k smallest eigenvalues (k: #cluster)
- determine number by:
  - gap in eigenvalues
  - using eigenvectors (by matrix rotation and cost function)

Apply k-means on k Eigenvectors 5



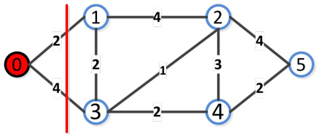
Map Results back to Original Data 6

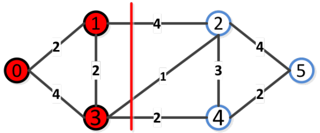


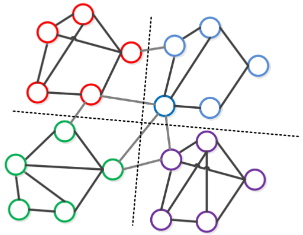
Clustering as Graph Partitioning

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 - Preliminaries

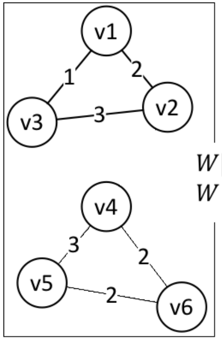
Given

Undirected graph G with weighted edges

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

Aim

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



$W[2,3] = 3$
$W[2,5] = 0$

2 connected components

Spectral Clustering : Preliminaries

Properties of L

1. For every vector $f \in \mathbb{R}^n$, we have: $fLf^T = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} W_{ij}(f_i - f_j)^2$
2. L is symmetric and positive semi-definite
3. The smallest eigenvalue of L is 0, with corresponding eigenvector $\mathbb{1}$
4. L has n non-negative, real-valued eigenvalues $0 = \lambda_0 \leq \lambda_1 \leq \ldots \leq \lambda_n$

Indicator vector

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

$$f_C{}^{(i)} = \begin{cases} 1 & \text{if } v_i \in C \\ 0 & \text{else} \end{cases}$$

Spectral Clustering: Graph Partitioning with Eigendecomposition

- 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?

Recap: Eigendecomposition

- Eigendecomposition on the Laplacian L:
  $LV = V\Lambda$, where the columns in V are the eigenvectors and Λ is a diagonal matrix with corresponding eigenvalues.
- Each element in $\Lambda : \lambda_i = v_i^T L v_i \geq 0$ (def. of positive semi-definite).

Spectral Clustering: k Connected Components

Observations: 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: Number of connected components

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

Spectral Clustering: Example for k connected components

- The graph consists of $k = 3$ independent connected components
- The k components yield a "perfect" clustering (no edges between clusters), minimizing $f_{C_i} L f_{C_i}^T = 0$ is given by the 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 $f_{C_3} = (0,0,0,0,0,0,1,1,1)$

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 $L = D - W$

$$L = \begin{pmatrix} L_1 & & \\ & L_2 & \\ & & L_3 \end{pmatrix}$$

- Because of the block form of L, we get $f_{C_i} L f_{C_i}^T = 0$ for each component C_i, i.e. the multiplicity of the eigenvalue 0 is 3 ($\lambda_0 = \lambda_1 = \lambda_2 = 0$).

Spectral Clustering: General Case

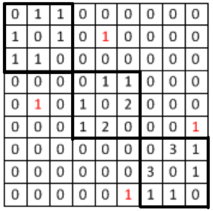
Observations: General Case

- All weights w_{ij} are non-negative, i.e. fLf^T can be minimized by making f_i be similar to f_j if the vertices v_i and v_j are connected
- Eigengap heuristic: Choose the number of clusters k such that all eigenvalues $\lambda_1, \dots, \lambda_k$ are small, but λ_{k+1} is relatively large.
  Motivations for that are:
  - k disconneted cluster have eigenvalue 0 and then there is a gap to $\lambda_{k+1} > 0$
  - The sizes of cuts are closely related to the size of the first eigenvalues

Spectral Clustering: General Case

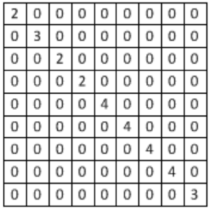
- In general: Multiplicity of eigenvalue 0 is 1 (i.e, only $\lambda_0 = 0$)
  - One large connected component $\rightarrow$ no perfect clustering possible
  - Determine the (linear independent) eigenvectors with the k *smallest eigenvalues*!

- Example: The 3 clusters are now connected by additional edges



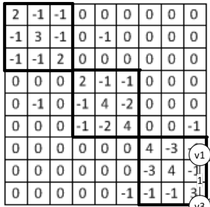
0	1	1	0	0	0	0	0	0
1	0	1	0	1	0	0	0	0
1	1	0	0	0	0	0	0	0
0	0	0	0	1	1	0	0	0
0	1	0	1	0	2	0	0	0
0	0	0	1	2	0	0	0	1
0	0	0	0	0	0	0	3	1
0	0	0	0	0	0	3	0	1
0	0	0	0	0	1	1	1	0

Adjacency matrix W

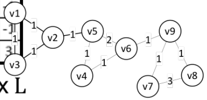


2	0	0	0	0	0	0	0	0
0	3	0	0	0	0	0	0	0
0	0	2	0	0	0	0	0	0
0	0	0	2	0	0	0	0	0
0	0	0	0	4	0	0	0	0
0	0	0	0	0	4	0	0	0
0	0	0	0	0	0	4	0	0
0	0	0	0	0	0	0	4	0
0	0	0	0	0	0	0	0	3

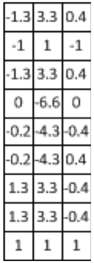
Degree matrix D



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	[illegible]
0	0	0	0	0	0	-3	4	-1
0	0	0	0	0	-1	-1	-1	3



Laplacian matrix L



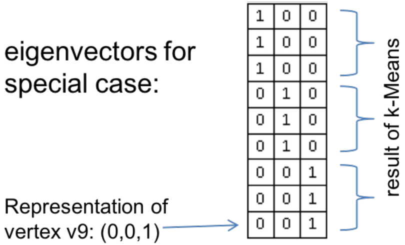
-1.3	3.3	0.4
-1	1	-1
-1.3	3.3	0.4
0	-6.6	0
-0.2	-4.3	-0.4
-0.2	-4.3	0.4
1.3	3.3	-0.4
1.3	3.3	-0.4
1	1	1

Eigenvectors $(v_i)_{i=1}^{3}$ of L

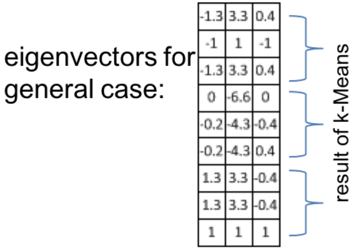
- Smallest eigenvalues of L, excluding non-trivial solutions $(\lambda_i, i \geq 1)$: $(0.23, 0.70, 3.43)$(Notice *eigengap* between λ_2 and λ_3)

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


eigenvectors for special case:
Representation of vertex v9: (0,0,1)
result of k-Means

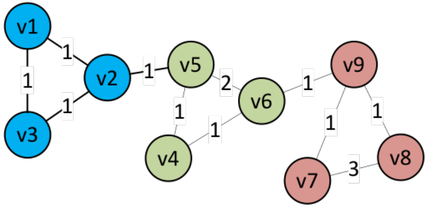
1	0	0
1	0	0
1	0	0
0	1	0
0	1	0
0	1	0
0	0	1
0	0	1
0	0	1

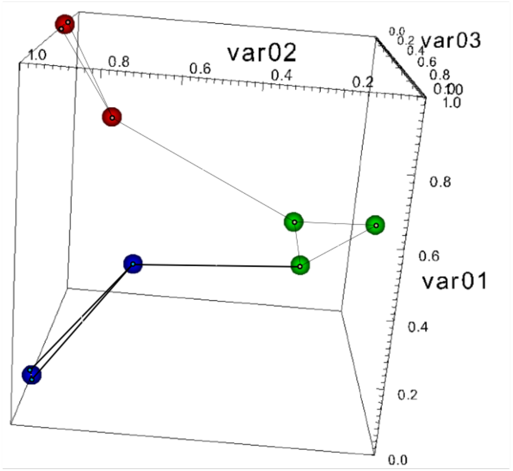


-1.3	3.3	0.4
-1	1	-1
-1.3	3.3	0.4
0	-6.6	0
-0.2	-4.3	-0.4
-0.2	-4.3	0.4
1.3	3.3	-0.4
1.3	3.3	-0.4
1	1	1

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

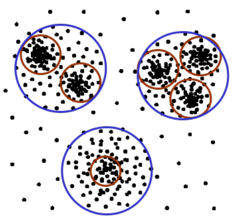
- Several variations of spectral clustering exist, using different Laplacian matrices which can be related to different graph cut problems [1]

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

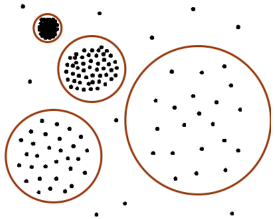
Agenda

From Partitioning to Hierarchical Clustering

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



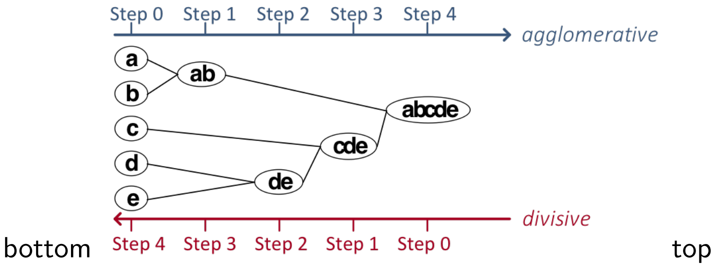
and/or



Need a hierarchical clustering algorithm in these situations

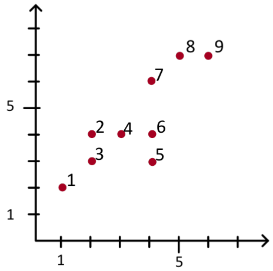
Hierarchical Clustering: Basic Notions

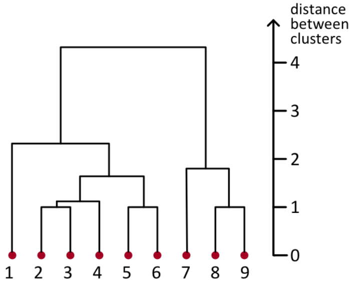
- Hierarchical decomposition of the data set (with respect to a given similarity measure) into a set of nested clusters
- Result represented by a so called *dendrogram* (greek $\delta\epsilon\nu\delta\rho o$ = tree)
  - Nodes in the dendrogram represent possible clusters
  - Dendrogram can be constructed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

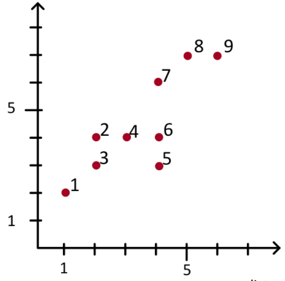


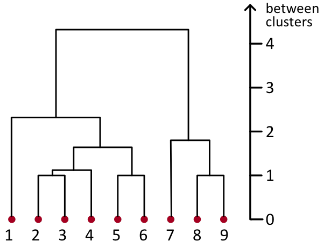


Agglomerative Hierarchical Clustering

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.

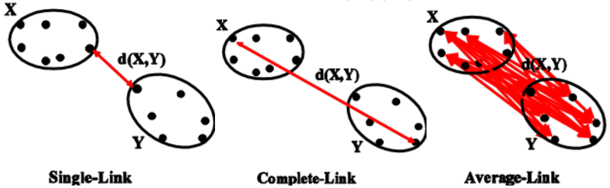




Single-Link Method and Variants

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



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

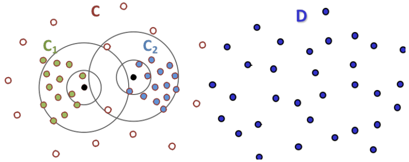
$$D(o') = \sum_{o_j \in C \setminus S} \frac{d(o', o_j)}{|C \setminus S|} - \sum_{o_i \in S} \frac{d(o', o_i)}{|S|}$$

Discussion Agglomerative vs. Divisive HC

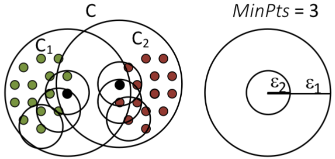
- Divisive and Agglomerative HC need $n-1$ steps
  - Agglomerative HC has to consider $\frac{n(n-1)}{2} = \binom{n}{2}$ combinations in the first step
  - Divisive HC potentially has $2^{n-1}-1$ many possibilities to split the data in its first step. Not every possibility has to be considered (DIANA)
- 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*

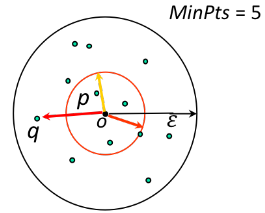
core-dist$_{\epsilon,MinPts}(o)$

- "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 $> \epsilon$: ∞

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



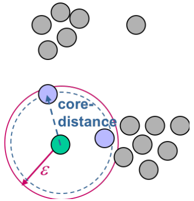
🟥 *core-distance(o)*
🟨 *reachability-distance(p,o)*
🟥 *reachability-distance(q,o)*

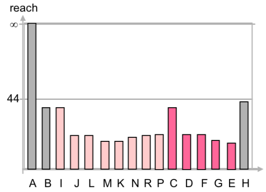
The Algorithm OPTICS

OPTICS[1]: Main Idea

"**O**rdering **P**oints **T**o **I**dentify the **C**lustering **S**tructure"

- Visit each point
  - Always make a shortest jump
- 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)

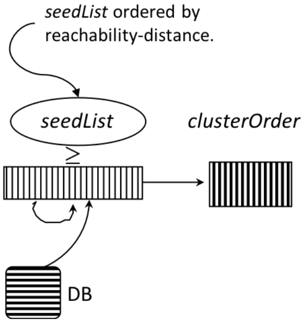




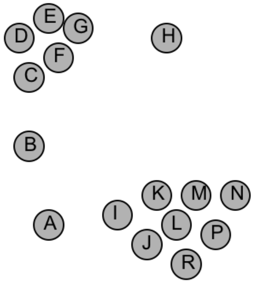
[1] Ankerst M., Breunig M., Kriegel H.-P., Sander J. "OPTICS: Ordering Points To Identify the Clustering Structure". SIGMOD (1999)

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$(p, o))$
```

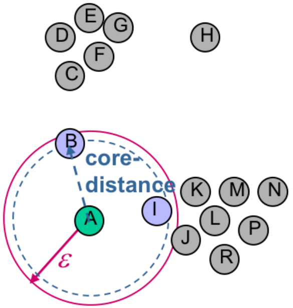


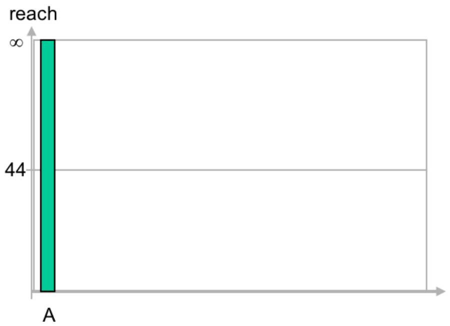
OPTICS: Example



seed list:

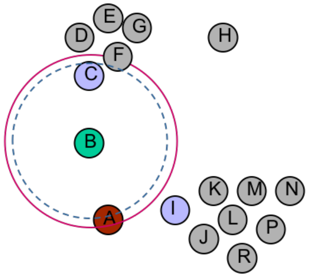
OPTICS: Example

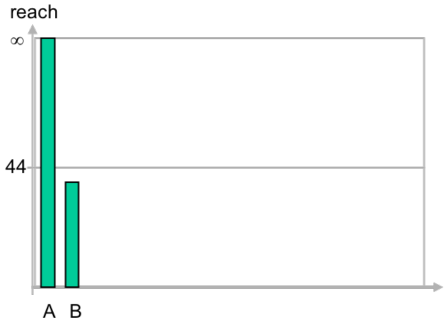




seed list: (B,40) (I, $\boxed{40}$)

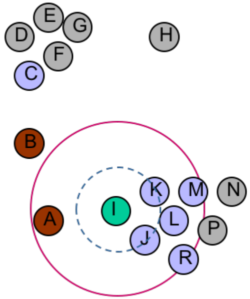
OPTICS: Example

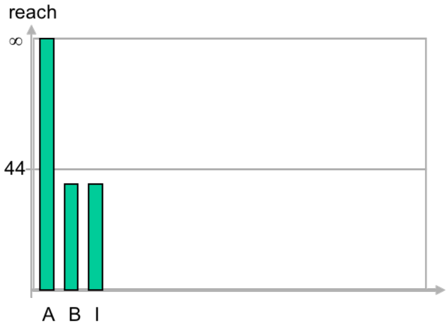




seed list: (I, 40) (C, 40)

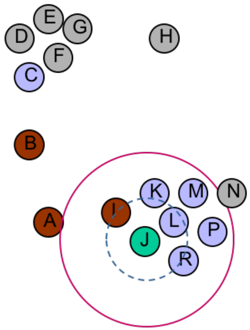
OPTICS: Example

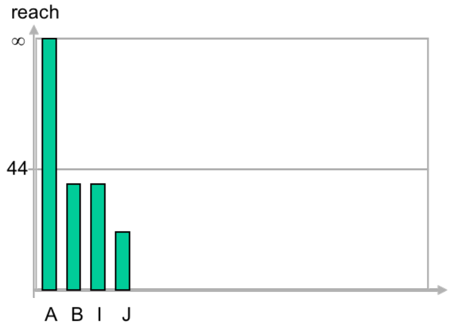




seed list: (J, 20) (K, 20) (L, 31) (C, 40) (M, 40) (R, 43)

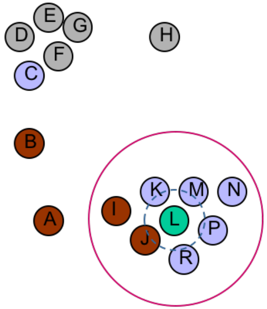
OPTICS: Example

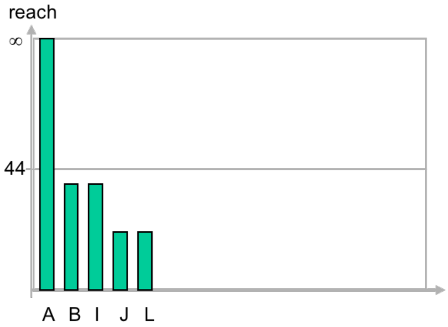




seed list: (L, 19) (K, 20) (R, 21) (M, 30) (P, 31) (C, 40)

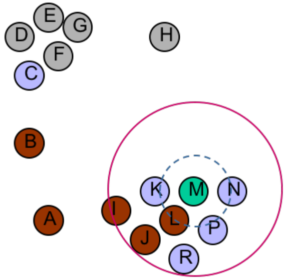
OPTICS: Example

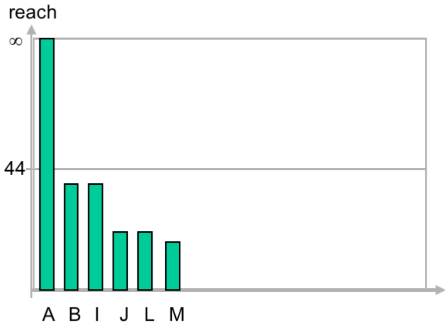




seed list: (M, 18) (K, 18) (R, 20) (P, 21) (N, 35) (C, 40)

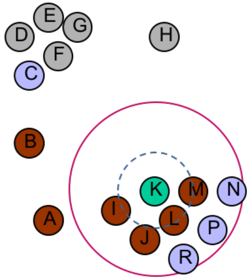
OPTICS: Example

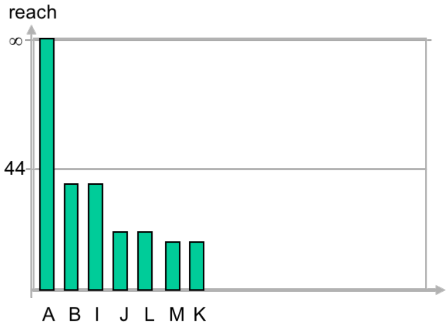




seed list: (K, 18) (N, 19) (R, 20) (P, 21) (C, 40)

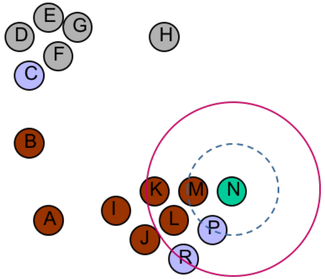
OPTICS: Example

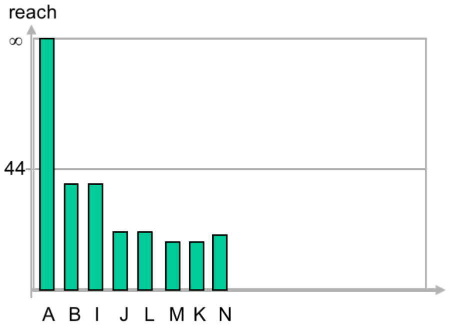




seed list: (N, 19) (R, 20) (P, 21) (C, 40)

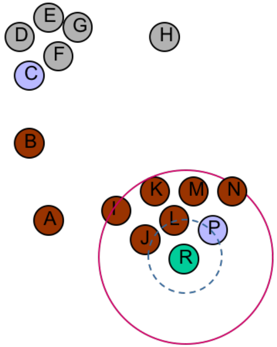
OPTICS: Example

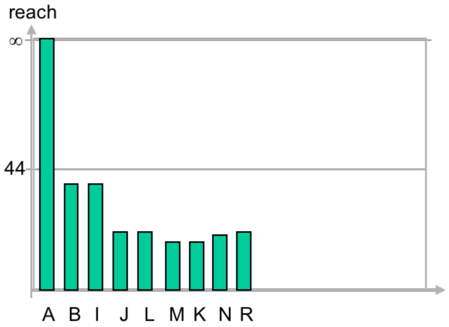




seed list: (R, 20) (P, 21) (C, 40)

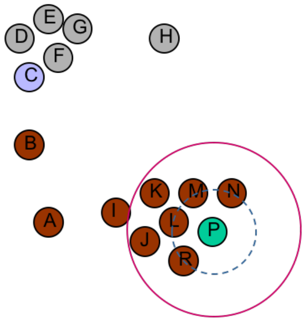
OPTICS: Example

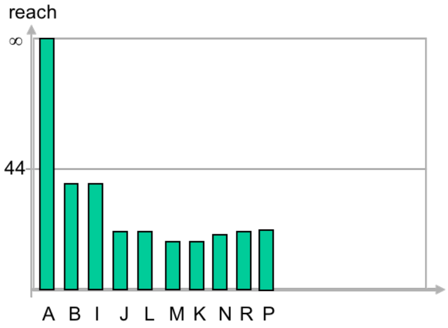




seed list: (P, 21) (C, 40)

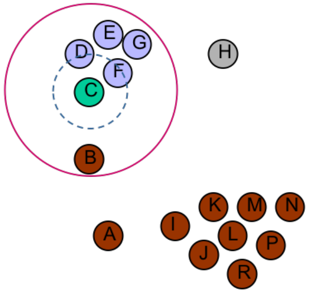
OPTICS: Example

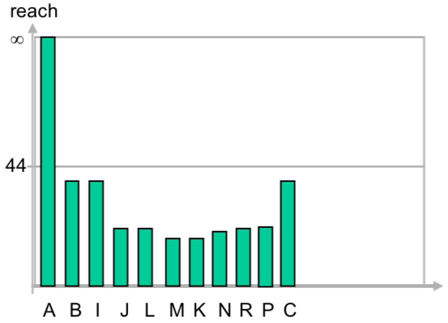




seed list: (C, 40)

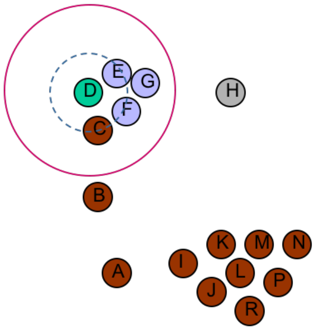
OPTICS: Example

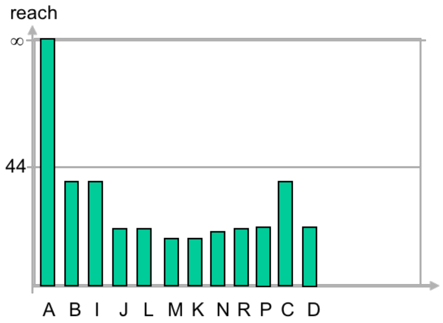




seed list: (D, 22) (F, 22) (E, 30) (G, 35)

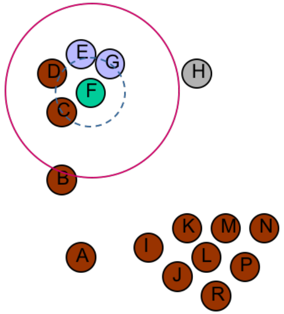
OPTICS: Example

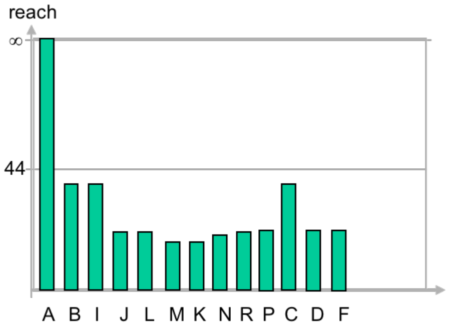




seed list: (F, 22) (E, 22) (G, 32)

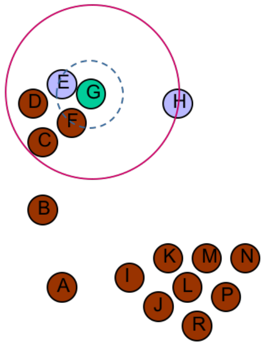
OPTICS: Example

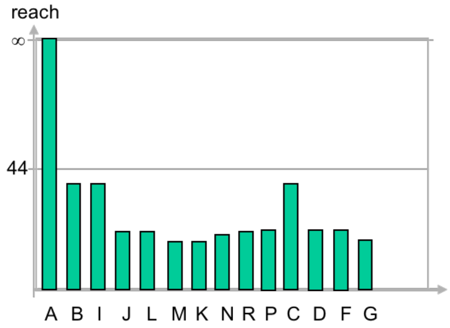




seed list: (G, 17) (E, 22)

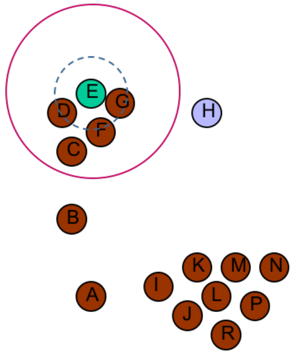
OPTICS: Example

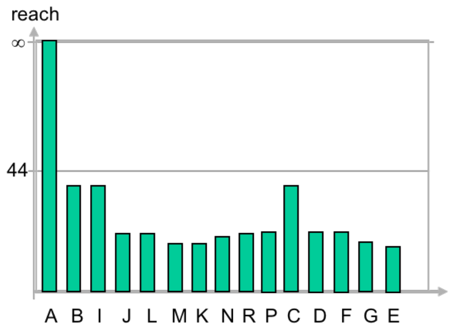




seed list: (E, 15) (H, 43)

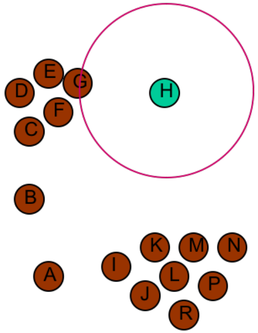
OPTICS: Example

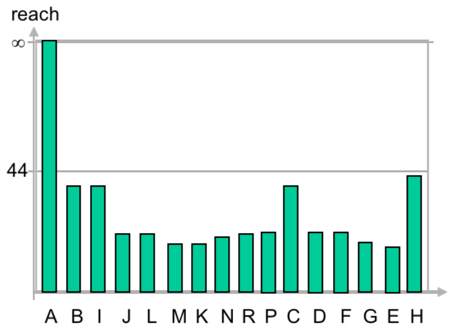




seed list: (H, 43)

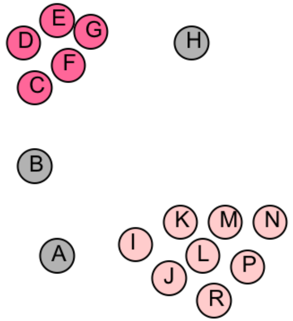
OPTICS: Example

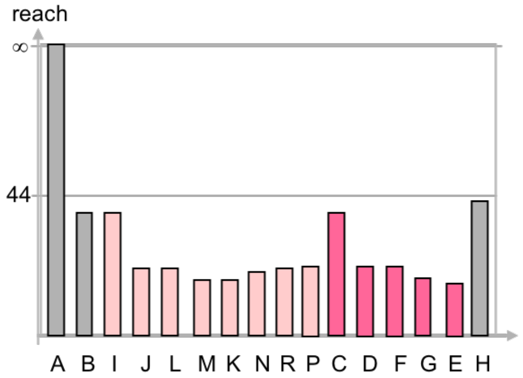




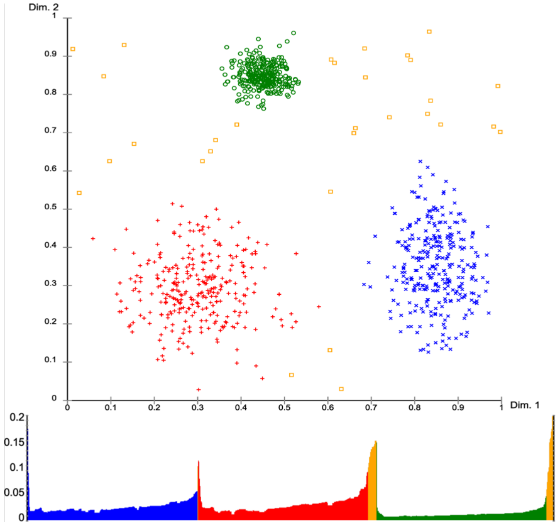
seed list: -

OPTICS: Example



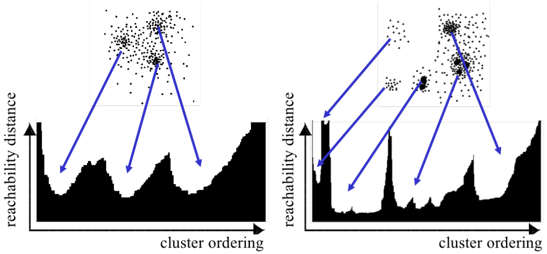


OPTICS: The Reachability Plot



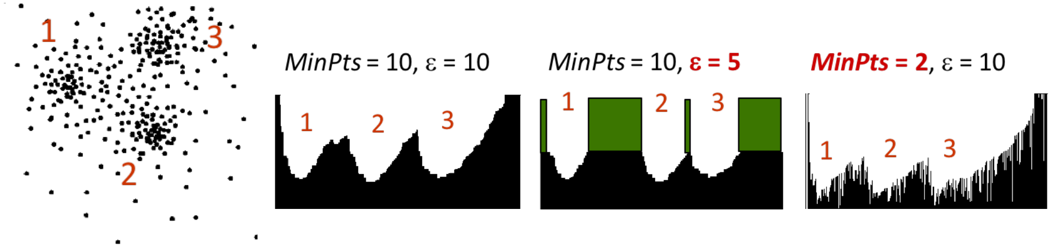
OPTICS: The Reachability Plot

- 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

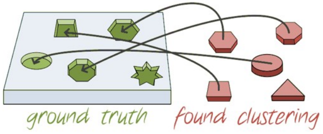
- May not scale well
  - Runtime for the standard methods: $\mathcal{O}(n^2 \log n^2)$
  - Runtime for OPTICS: without index support $\mathcal{O}(n^2)$
- User has to choose the final clustering

Agenda

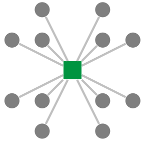
Evaluation of Clustering Results

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 $\mathcal{G} = \{G_1, \ldots, G_l\}$.

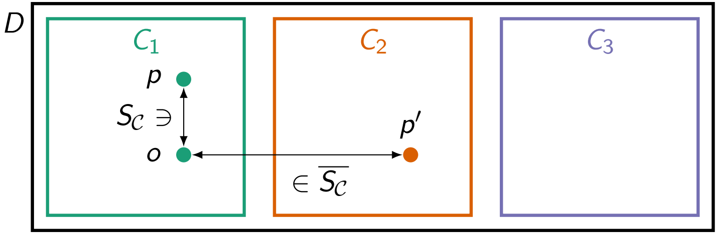
Problem

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 for Clustering



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

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

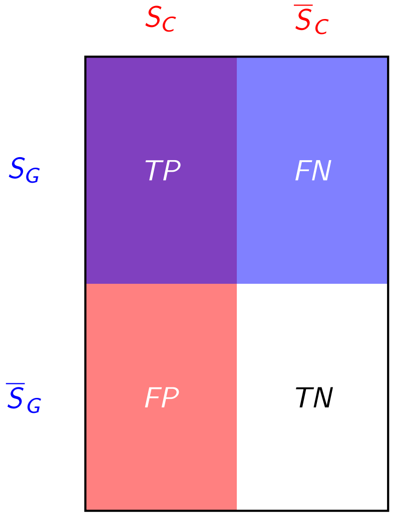
and analogously for $\mathcal{G}$.

Formalisation as Retrieval Problem for Clustering

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")

Note the difference to the definitions in classification!



External Measures - Retrieval Problem

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

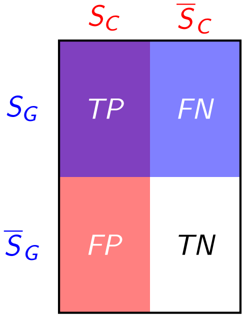
$$rec = \frac{TP}{TP + FN} = \frac{|S_{\mathcal{C}} \cap S_{\mathcal{G}}|}{|S_{\mathcal{G}}|}$$

- **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)

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



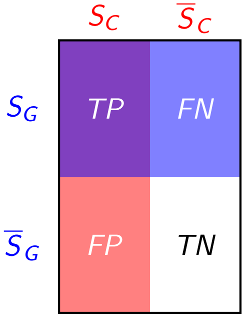
External Measures - Retrieval Problem

- **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 - Retrieval Problem

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

$$\begin{array}{c|ccc} & G_1 & \dots & G_l \\ \hline C_1 & |C_1 \cap G_1| & \dots & |C_1 \cap G_l| \\ \vdots & \vdots & \ddots & \\ C_k & |C_k \cap G_1| & & |C_k \cap G_l| \end{array}$$

- 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 - Information Theory

- **(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 - Information Theory

- **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.

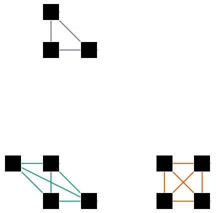
Cohesion

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

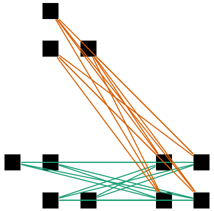
$$sep(C_i, C_j) = \frac{1}{|C_i||C_j|} \sum_{o \in C_i, p \in C_j} d(o, p)$$

Separation of one cluster: Minimum separation to another cluster:

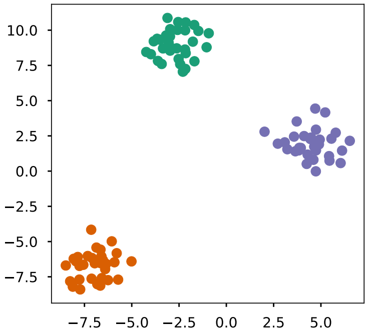
$$sep(C_i) = \min_{j \neq i} sep(C_i, C_j)$$

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

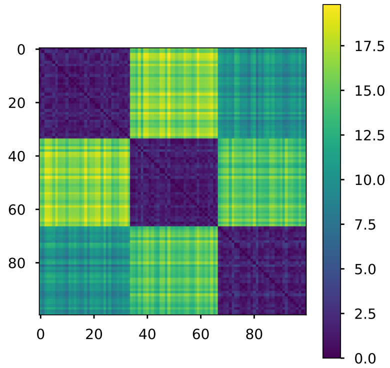
$$sep(\mathcal{C}) = \sum_{i=1}^{k} \frac{|C_i|}{n} sep(C_i)$$



Evaluating the Distance Matrix



dataset
(well separated)



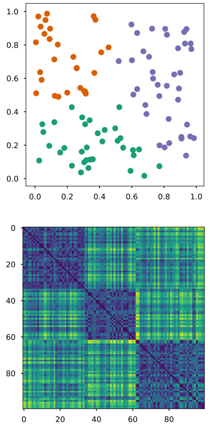
Distance matrix
(sorted by *k*-means cluster label)

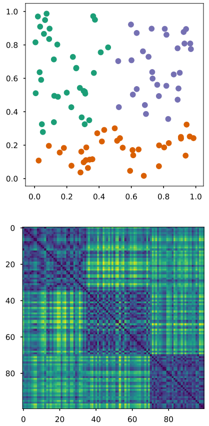
after: Tan, Steinbach, Kumar: Introduction to Data Mining (Pearson, 2006)

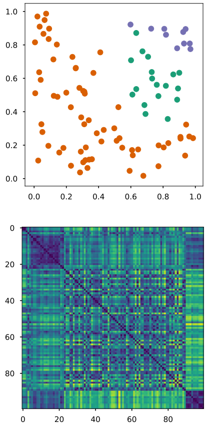
Evaluating the Distance Matrix

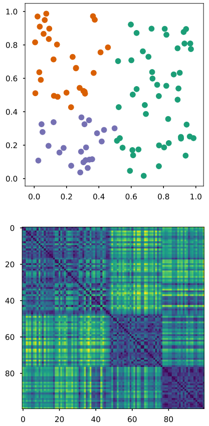
Distance matrices differ for different clustering approaches (here on random data)

k-means EM DBSCAN Complete Link







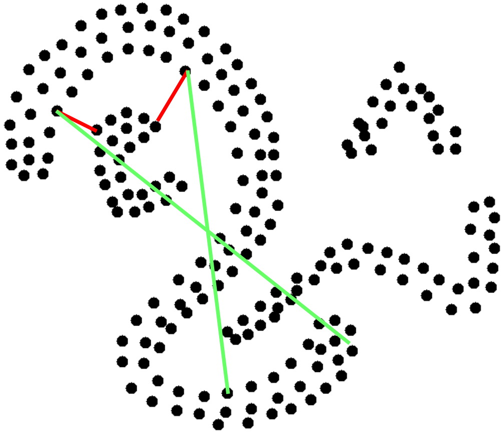


after: Tan, Steinbach, Kumar: Introduction to Data Mining (Pearson, 2006)

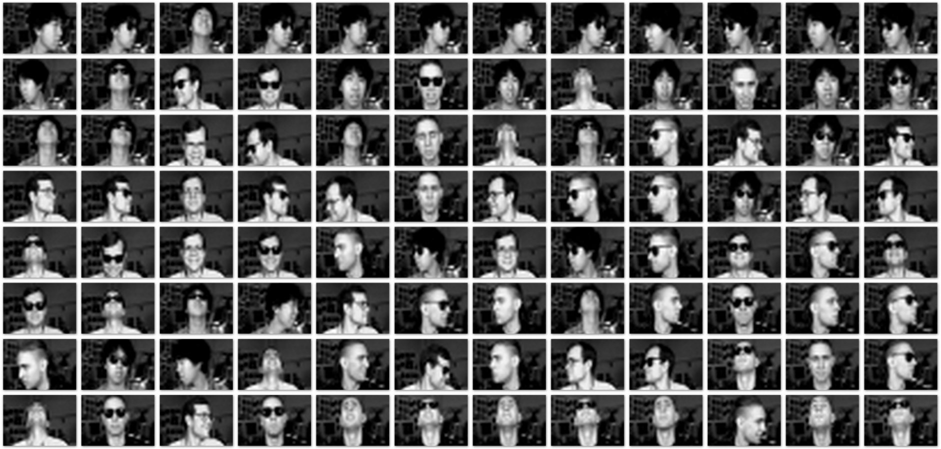
Cohesion and Separation

Problem

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

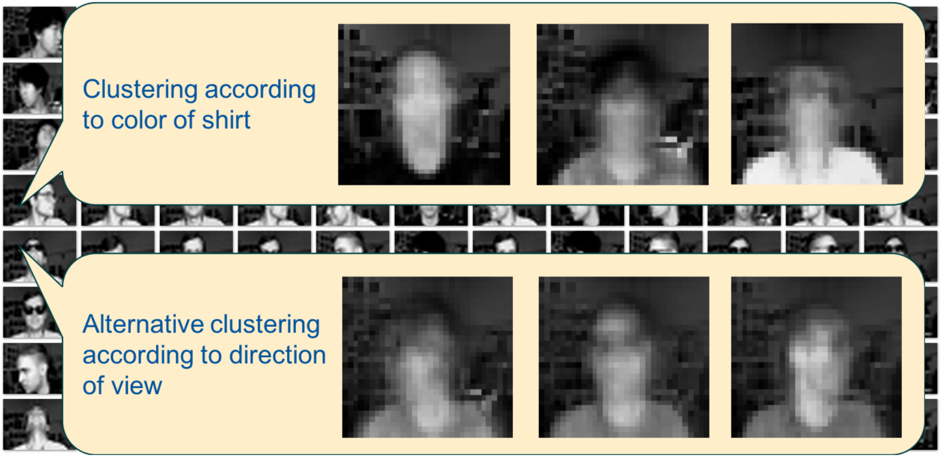


Ambiguity of Clusterings



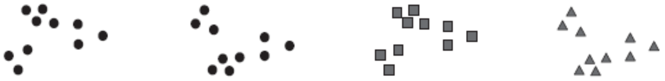
- Clustering according to: Color of shirt, direction of view, glasses, . . .

Ambiguity of Clusterings


Clustering according to color of shirt
Alternative clustering according to direction of view

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

Ambiguity of Clusterings


(a) Original points. (b) Two clusters.

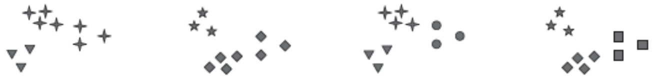

(c) Four clusters. (d) Six clusters.

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

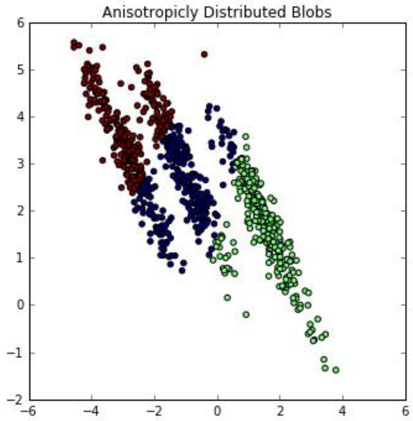
from: Tan, Steinbach, Kumar: Introduction to Data Mining (Pearson, 2006)

Ambiguity of Clusterings

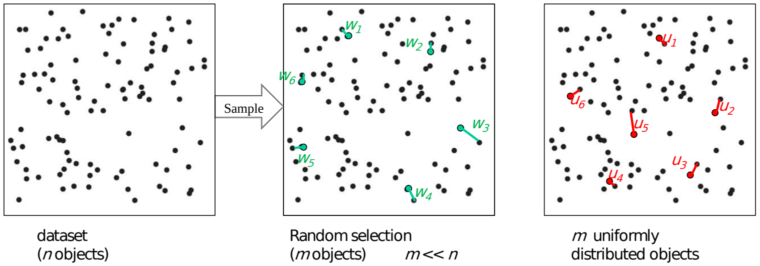
"Philosophical" Problem

"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



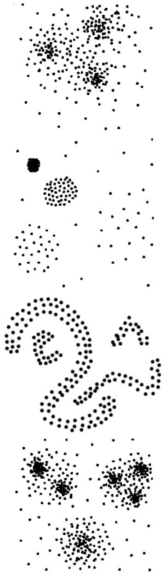
$$H = \frac{\sum_{i=1}^{m} u_i}{\sum_{i=1}^{m} u_i + \sum_{i=1}^{m} w_i}$$

- 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,

Recap: Observed Clustering Methods

- Partitioning Methods: Find k partitions, minimizing some objective function
- Probabilistic Model-Based Clustering (EM)
- Density-based Methods: Find clusters based on connectivity and density functions
- Mean-Shift: Find modes in the point density
- Spectral Clustering: Find global minimum cut
- Hierarchical Methods: Create a hierarchical decomposition of the set of objects

- Evaluation: External and internal measures



Agenda

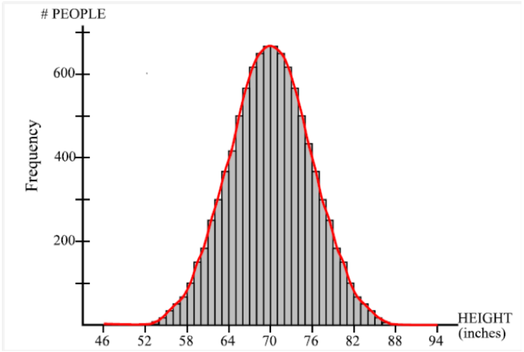
Agenda

Introduction

What is an outlier?

Hawkins (1980) "An outlier is an observation which deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism."

- Statistics-based intuition:
  - Normal data objects follow a "generating mechanism", e.g. some given statistical process
  - Abnormal objects deviate from this generating mechanism



Introduction

Applications

- Fraud detection
  - Purchasing behavior of a credit card owner usually changes when the card is stolen
  - Abnormal buying patterns can characterize credit card abuse
- Medicine
  - Whether a particular test result is abnormal may depend on other characteristics of the patients (e.g. gender, age, . . .)
  - Unusual symptoms or test results may indicate potential health problems of a patient
- Public health
  - The occurrence of a particular disease, e.g. tetanus, scattered across various hospitals of a city indicate problems with the corresponding vaccination program in that city
  - Whether an occurrence is abnormal depends on different aspects like frequency, spatial correlation, etc.

Introduction

Applications (cont'd)

- Sports statistics
  - In many sports, various parameters are recorded for players in order to evaluate the players' performances
  - Outstanding (in a positive as well as a negative sense) players may be identified as having abnormal parameter values
  - Sometimes, players show abnormal values only on a subset or a special combination of the recorded parameters
- Detecting measurement errors
  - Data derived from sensors (e.g. in a given scientific experiment) may contain measurement errors
  - Abnormal values could provide an indication of a measurement error
  - Removing such errors can be important in other data mining and data analysis tasks
  - *"One person's noise could be another person's signal."*

Introduction

Important Properties of Outlier Models

- Global vs. local approach
  - "Outlierness" regarding whole dataset (global) or regarding a subset of data (local)?
- Labeling vs. Scoring
  - Binary decision or outlier degree score?
- Assumptions about "Outlierness"
  - What are the characteristics of an outlier object?

- An object is a cluster-based outlier if it does not strongly belong to any cluster.

Agenda

Density-Based Approaches

General Idea

- Compare the density around a point with the density around its local neighbors.
- The relative density of a point compared to its neighbors is computed as an outlier score.
- Approaches also differ in how to estimate density.

Basic Assumption

- The density around a normal data object is similar to the density around its neighbors.
- The density around an outlier is considerably different to the density around its neighbors.

Density-Based Approaches

Problems

- Different definitions of density: e.g., #points within a specified distance ϵ from the given object
- The choice of ϵ is critical (too small $\implies$ normal points considered as outliers; too big $\implies$ outliers considered normal)
- A global notion of density is problematic (as it is in clustering); fails when data contain regions of different densities

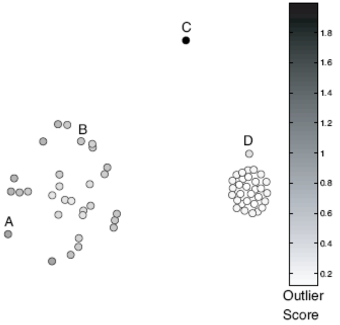


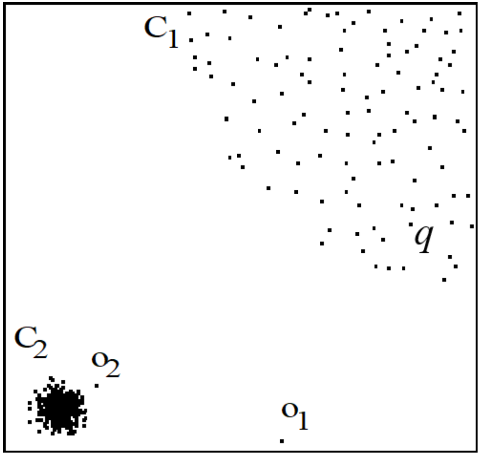
Figure 10.7. Outlier score based on the distance to the fifth nearest neighbor. Clusters of differing density.

D has a higher absolute density than A but compared to its neighborhood, Ds density is lower.

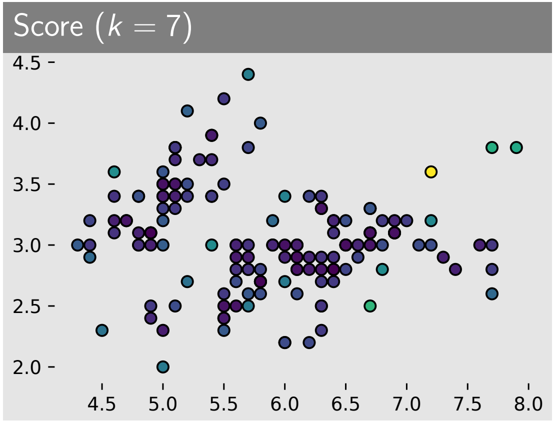
Density-Based Approaches

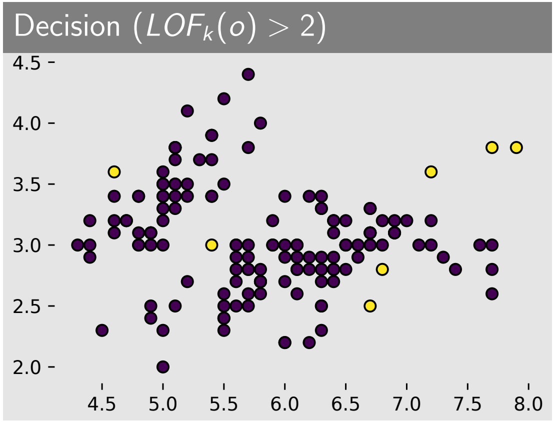
Failure Case of Distance-Based

- $D(\epsilon, \pi)$: parameters ϵ, π cannot be chosen s.t. o_2 is outlier, but none of the points in C_1 (e.g. q)
- kNN-distance: kNN-distance of objects in C_1 (e.g. q) larger than the kNN-distance of o_2.



Density-Based Approaches





Density-Based Approaches

Solution

Consider the relative density w.r.t. to the neighbourhood.

Model

- Local Density (*ld*) of point *p* (inverse of avg. distance of *k*NNs of *p*)

$$ld_k(p) = \left(\frac{1}{k} \sum_{o \in kNN(p)} dist(p, o) \right)^{-1}$$

- Local Outlier Factor (LOF) of *p* (avg. ratio of *ld*s of *k*NNs of *p* and *ld* of *p*)

$$LOF_k(p) = \frac{1}{k} \sum_{o \in kNN(p)} \frac{ld_k(o)}{ld_k(p)}$$

Density-Based Approaches

Extension **(Smoothing factor)**

- Reachability "distance"

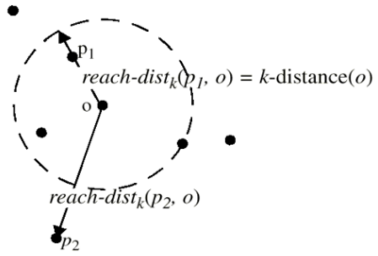
$$rd_k(p,o) = \max\{kdist(o), dist(p,o)\}$$

- Local reachability distance lrd_k

$$lrd_k(p) = \left(\frac{1}{k}\sum_{o\in kNN(p)} rd(p,o)\right)^{-1}$$

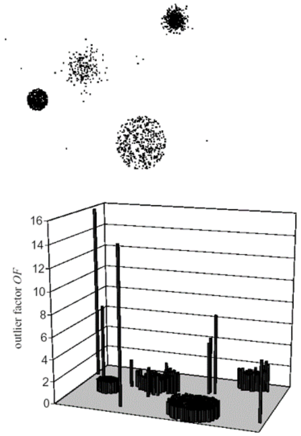
- Replace ld by lrd

$$LOF_k(p) = \frac{1}{k}\sum_{o\in kNN(p)} \frac{lrd_k(o)}{lrd_k(p)}$$



p_1
$reach\text{-}dist_k(p_1, o) = k\text{-distance}(o)$
o
$reach\text{-}dist_k(p_2, o)$
p_2

Density-Based Approaches



Discussion

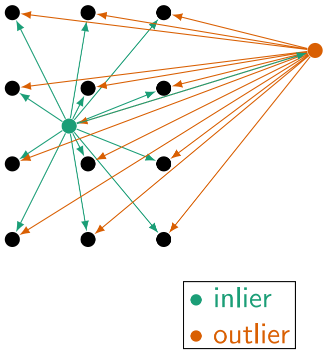
- $LOF \approx 1 \implies$ point in cluster
- $LOF \gg 1 \implies$ outlier.
- Choice of k defines the reference set

Agenda

Angle-Based Approach

General Idea

- Angles are more stable than distances in high dimensional spaces
- *o outlier* if most other objects are located in similar directions
- *o no outlier* if many other objects are located in varying directions



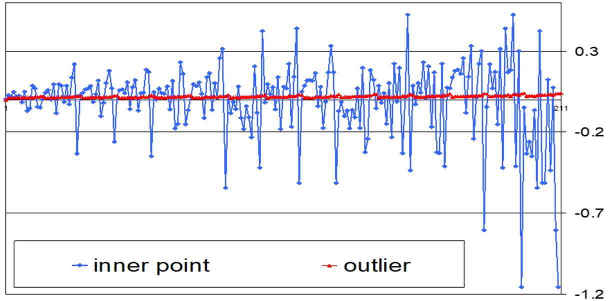
Basic Assumption

- Outliers are at the border of the data distribution
- Normal points are in the center of the data distribution

Angle-Based Approach

Model

- Consider for a given point p the angle between $\overrightarrow{px}$ and $\overrightarrow{py}$ for any two x, y from the database
- Measure the variance of the angle spectrum



Angle-Based Approach

Model (cont'd)

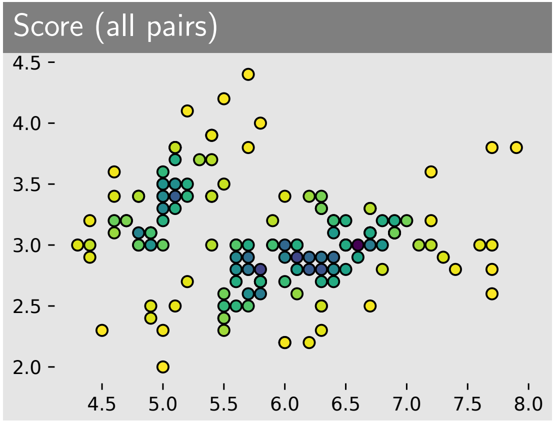
- Weighted by the corresponding distances (for lower dimensional data sets where angles are less reliable)
  Angle-based Outlier Detection[5]:

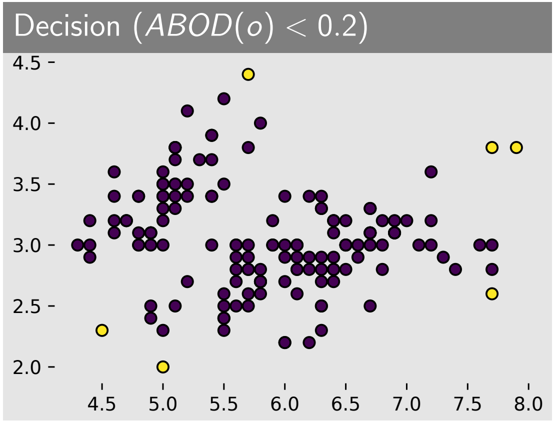
$$ABOD(p) = \mathrm{VAR}_{x,y \in D}\left(\frac{1}{\|\overrightarrow{xp}\|_2 \|\overrightarrow{yp}\|_2} \cos\left(\overrightarrow{xp}, \overrightarrow{yp}\right)\right) = \mathrm{VAR}_{x,y \in D}\left(\frac{\langle \overrightarrow{xp}, \overrightarrow{yp} \rangle}{\|\overrightarrow{xp}\|_2^2 \|\overrightarrow{yp}\|_2^2}\right)$$

- Small ABOD $\iff$ outlier

[5] Kriegel, Hans-Peter, Matthias Schubert, and Arthur Zimek. "Angle-based outlier detection in high-dimensional data." Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2008.

Angle-Based Approaches





Agenda

Tree-Based Approaches: Isolation Forest

General Idea

Outlierness = how easy it is to separate a point from the rest by random space splitting?

Basic Assumption

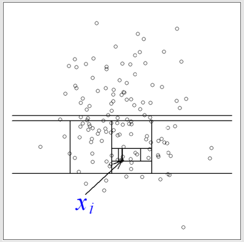
- Anomalies are the minority consisting of fewer instances
- Anomalies have attribute-values that are very different from those of normal instances

Tree-Based Approaches

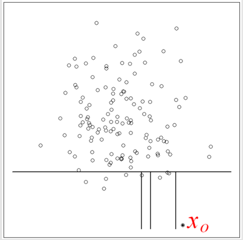
Isolation Tree - Training

1. Randomly select one dimension
2. Randomly select a split position in that dimension
3. Repeat until: a) only one point left or b) height reaches predefined threshold *h*

Normal point path length=10 splits



Outlier point path length=4 splits

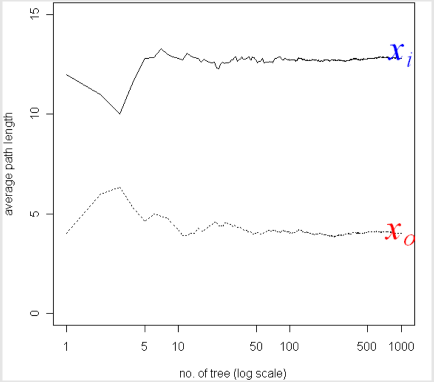


Tree-Based Approaches: Training

Isolation Forest - Training

1. Random sample ψ points, build an isolation tree
2. Repeat for t times $\Rightarrow$ a forest of t isolation trees

Average path lengths converge



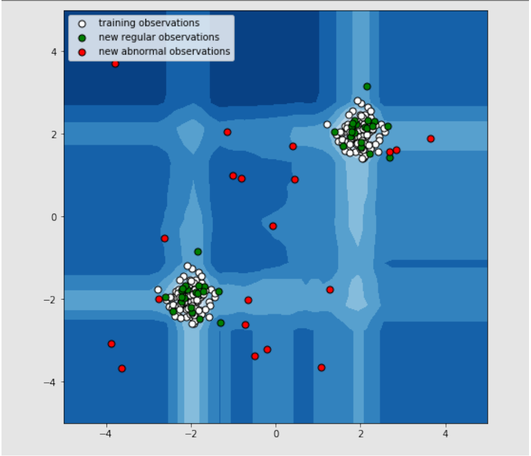
Tree-Based Approaches: Anomaly Score

- Let $h(x)$ be the path length of x on an isolation tree, and estimate $E(h(x))$ by the *average path length* among t isolation trees.
- Let $c(\psi) = 2H(\psi - 1) - 2(\psi - 1)/\psi$, which is the expected path length of unsuccessful search in BST of ψ points; $H(\cdot)$ is the harmonic number.
- Define the anomaly score of a point x as $s(x) = 2^{-\frac{E(h(x))}{c(\psi)}}$
- Observe $s(x) \in (0, 1)$
  - $E(h(x)) \to c(\psi)$ yields $s \to 0.5$,
  - $E(h(x)) \to 0$ yields $s \to 1$,
  - $E(h(x)) \to n - 1$ yields $s \to 0$.
- Usually, set $s = 0.5$ as threshold, i.e. the average of the expected path length

Tree-Based Approaches: Discussion

- Advantages:
  - Anomaly score between 0 and 1
  - Very efficient, especially on large dataset
  - A model (the forest) is learned from the training dataset
  - Easy for parallelization
  - Can be adapted to categorical data
- Disadvantages:
  - Only detects global outliers (of course, follow-up approaches are available)
  - Not efficient on high-dimensional data

iForest anomaly score contour



Recap - Outlier Detection

- Properties: global vs. local, labeling vs. scoring
- *Clustering-Based* Outliers: Identification as non-(cluster-members)
- *Statistical* Outliers: Assume probability distribution; outliers = unlikely to be generated by distribution
- *Distance-Based* Outliers: Distance to neighbors as outlier metric
- *Density-Based* Outliers: Relative density around the point as outlier metric
- *Angle-Based* Outliers: Angles between outliers and random point pairs vary only slightly

Agenda

Agenda

What is Frequent Pattern Mining?

Setting: Transaction Databases

A database of transactions, where each transaction comprises a set of items, e.g. one transaction is the basket of one customer in a grocery store.

Frequent Pattern Mining

Finding frequent patterns, associations, correlations, or causal structures among sets of items or objects in transaction databases, relational databases, and other information repositories.

Applications

Basket data analysis, cross-marketing, catalogue design, loss-leader analysis, clustering, classification, recommendation systems, etc.

What is Frequent Pattern Mining?

Task 1: Frequent Itemset Mining

Find all subsets of items that occur together in many transactions.

Example

Which items are bought together frequently?

$$
\begin{aligned}
D = \{&\{butter, bread, milk, sugar\},\\
&\{butter, flour, milk, sugar\},\\
&\{butter, eggs, milk, salt\},\\
&\{eggs\},\\
&\{butter, flour, milk, salt, sugar\}\}
\end{aligned}
$$

$\leadsto$ 80% of transactions contain the itemset {milk, butter}

What is Frequent Pattern Mining?

Task 2: Association Rule Mining

Find all rules that correlate the presence of one set of items with that of another set of items in the transaction database.

Example

98% of people buying tires and auto accessories also get automotive service done

Agenda

Mining Frequent Itemsets: Basic Notions

- **Items** $I = \{i_1, \ldots, i_m\}$: a set of literals (denoting items)
- **Itemset** X: Set of items $X \subseteq I$
- **Database** D: Set of *transactions* T, each transaction is a set of items $T \subseteq I$
- Transaction T contains an itemset X: $X \subseteq T$
- **Length** of an itemset X equals its cardinality $|X|$
- k-**itemset**: itemset of length k
- (Relative) **Support** of an itemset: $supp(X) = |\{T \in D \mid X \subseteq T\}| / |D|$
- X is **frequent** if $supp(X) \geq minSup$ for threshold $minSup$.

Task

Given a database D and a threshold $minSup$, find all frequent itemsets $X \subseteq I$.

Mining Frequent Itemsets: Basic Idea

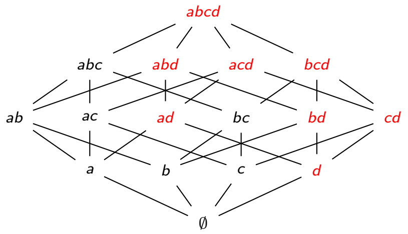
Naïve Algorithm

Count the frequency of all possible subsets of I in the database D.

Problem

Too expensive since there are 2^m such itemsets for m items (for $|I| = m$, $2^m =$ cardinality of the powerset of I).

Mining Frequent Patterns: Apriori Principle



Hasse diagram shows lattice structure of complete partial order of item subsets

- frequent
- non-frequent

Apriori Principle (anti-monotonicity)

- Any (non-empty) subset of a frequent itemset A is frequent:
$$\forall A' \subseteq A : supp(A) \geq minSup \implies supp(A') \geq minSup$$
- Any superset of a non-frequent itemset A is non-frequent:
$$\forall A'' \supseteq A : supp(A) < minSup \implies supp(A'') < minSup$$

Apriori Algorithm

Idea

- First count the 1-itemsets, then the 2-itemsets, then the 3-itemsets, and so on
- When counting $(k+1)$-itemsets, only consider those $(k+1)$-itemsets where all subsets of length k have been determined as frequent in the previous step

Apriori Algorithm

```
variable $C_k$: candidate itemsets of size $k$
variable $L_k$: frequent itemsets of size $k$
$L_1 = \{$frequent items$\}$
**for** $(k = 1;\ L_k \neq \emptyset;\ k{+}{+})$ **do**
Produce candidates.
    join $L_k$ with itself to produce $C_{k+1}$                      ▷ JOIN STEP
    discard $(k+1)$-itemsets from $C_{k+1}$ that ...                ▷ PRUNE STEP
        ...contain non-frequent $k$-itemsets as subsets
    $C_{k+1}$ = candidates generated from $L_k$
Prove candidates.
    **for** each transaction $T \in D$ **do**
        Increment the count of all candidates in $C_{k+1}$ ...
            ...that are contained in $T$
    $L_{k+1}$ = candidates in $C_{k+1}$ with *minSupp*
**return** $\bigcup_k L_k$
```

Apriori Algorithm: Generating Candidates – Join Step

Requirements for Candidate $(k+1)$-itemsets

- *Completeness*: Must contain all frequent $(k+1)$-itemsets (superset property $C_{k+1} \supseteq L_{k+1}$)
- *Selectiveness*: Significantly smaller than the set of all $(k+1)$-subsets

Suppose the itemsets are sorted by any order (e.g. lexicographic)

Step 1: Joining ($C_{k+1} = L_k \bowtie L_k$)

- Consider frequent k-itemsets p and q
- p and q are joined if they share the same first $(k-1)$ items.

Apriori Algorithm: Generating Candidates – Join Step

Example

- $k = 3 \; (\implies k + 1 = 4)$
- $p = (a, c, f) \in L_k$
- $q = (a, c, g) \in L_k$
- $r = (a, c, f, g) \in C_{k+1}$

SQL example

insert into C_{k+1}

select $p.i_1, p.i_2, \ldots, p.i_k, q.i_k$

from $L_k : p, L_k : q$

where $p.i_1 = q.i_1, \ldots, p.i_{k-1} = q.i_{k-1}, p.i_k < q.i_k$

Apriori Algorithm: Generating Candidates – Prune Step

Step 2: Pruning ($L_{k+1} = \{X \in C_{k+1} \mid supp(X) \geq minSup\}$)

- *Naïve*: Check support of every itemset in C_{k+1} $\rightsquigarrow$ inefficient for huge C_{k+1}
- *Better*: Apply Apriori principle first: Remove candidate $(k+1)$-itemsets which contain a non-frequent k-subset s, i.e., $s \notin L_k$

Pseudocode

```
for all $c \in C_{k+1}$ do
    for all $k$-subsets $s$ of $c$ do
        if $s \notin L_k$ then
            Delete $c$ from $C_{k+1}$
```

Apriori Algorithm: Generating Candidates – Prune Step

Example

- $L_3 = \{acf, acg, afg, afh, cfg\}$
- Candidates after join step: $\{acfg, afgh\}$
- In the pruning step: delete $afgh$ because $fgh \notin L_3$, i.e. fgh is not a frequent 3-itemset (also $agh \notin L_3$)
- $C_4 = \{acfg\} \rightsquigarrow$ check the support to generate L_4

Apriori Algorithm: Full example

Database

TID	items
0	acdf
1	bce
2	abce
3	aef

minSup = 0.5

Alphabetic Ordering

k	candidate	prune	count	threshold
1	a		3	a
	b		2	b
	c		3	c
	d		1	
	e		3	e
	f		2	f
2	ab		1	
	ac		2	ac
	ae		2	ae
	af		2	af
	bc		2	bc
	be		2	be
	bf		0	
	ce		2	ce
	cf		1	
	ef		1	
3	ace		1	
	acf	with cf		
	aef	with ef		
	bce		2	bce

Frequency-Ascending Ordering

k	candidate	prune	count	threshold
1	d		1	
	b		2	b
	f		2	f
	a		3	a
	c		3	c
	e		3	e
2	bf		0	
	ba		1	
	bc		2	bc
	be		2	be
	fa		2	fa
	fc		1	
	fe		1	
	ac		2	ac
	ae		2	ae
	ce		2	ce
3	bce		2	bce
	ace		1	

Counting Candidate Support

Motivation

Why is counting supports of candidates a problem?

- Huge number of candidates
- One transaction may contain many candidates

Solution

Store candidate itemsets in hash-tree

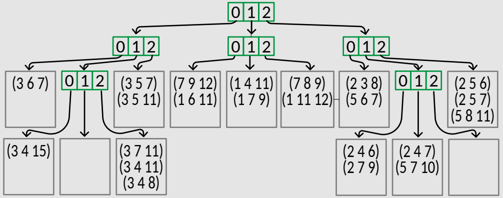
Counting Candidate Support: Hash Tree

Hash-Tree

- Leaves contain itemset lists with their support (e.g. counts)
- Interior nodes comprise hash tables
- *subset* function to find all candidates contained transaction

Example

3-itemsets; $h(i) = i \mod 3$



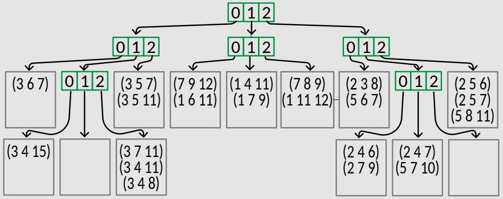
Hash-Tree: Construction

Search

- Start at the root (level 1)
- At level d: Apply hash function h to d-th item in the itemset

Example

3-itemsets; $h(i) = i \mod 3$



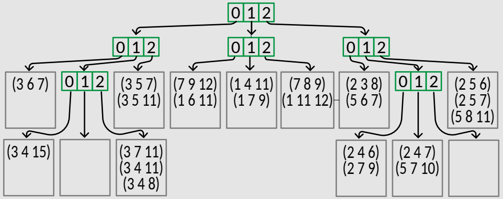
Hash-Tree: Construction

Insertion

- Search for the corresponding leaf node
- Insert the itemset into leaf; if an overflow occurs:
  - Transform the leaf node into an internal node
  - Distribute the entries to the new leaf nodes according to the hash function h

Example

3-itemsets; $h(i) = i \mod 3$



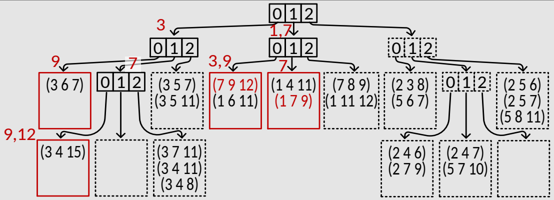
Hash-Tree: Counting

Search all candidates of length k in transaction $T = (t_1, \dots, t_n)$

- At root:
  - Compute hash values for all items $t_1, \dots, t_{n-k+1}$
  - Continue search in all resulting child nodes
- At internal node at level d (reached after hashing of item t_i):
  - Determine the hash values and continue the search for each item t_j with $i < j \leq n - k + d$
- At leaf node:
  - Check whether the itemsets in the leaf node are contained in transaction T

Example

3-itemsets;
$h(i) = i \mod 3$
Transaction:
$\{1, 3, 7, 9, 12\}$



Apriori – Performance Bottlenecks

Huge Candidate Sets

- 10^4 frequent 1-itemsets will generate 10^7 candidate 2-itemsets
- To discover a frequent pattern of size 100, one needs to generate $2^{100} \approx 10^{30}$ candidates.

Multiple Database Scans

- Needs n or $n+1$ scans, where n is the length of the longest pattern

Is it possible to mine the complete set of frequent itemsets without candidate generation?

Mining Frequent Patterns *Without Candidate Generation*

Idea

- Compress large database into compact tree structure; complete for frequent pattern mining, but avoiding several costly database scans (called *FP-tree*)
- Divide compressed database into *conditional databases* associated with one frequent item

FP-Tree Construction

minSup=2/12

Database

TID	Items
1	c
2	cd
3	cef
4	cef
5	bcd
6	bcd
7	bcdg
8	bde
9	bd
10	bh
11	bi
12	b

1. Scan DB once to identify frequent items (1-itemsets)
2. Scan DB again:
   - 2.1 Keep frequent items only; sort them within itemsets by descending frequency
   - 2.2 Does path with common prefix exist?
     Yes: Increment counter; append suffix;
     No: Create new branch

FP-Tree Construction

minSup=2/12

Database

TID	Items
1	c
2	cd
3	cef
4	cef
5	bcd
6	bcd
7	bcdg
8	bde
9	bd
10	bh
11	bi
12	b

Header Table 1

Item	Frequency
b	8
c	7
d	6
e	3
f	2

1. Scan DB once to identify frequent items (1-itemsets)
2. Scan DB again:
   2.1 Keep frequent items only; sort them within itemsets by descending frequency
   2.2 Does path with common prefix exist?
   Yes: Increment counter; append suffix;
   No: Create new branch

FP-Tree Construction

minSup=2/12

Database

TID	Items	Freq. Item
1	c	c
2	cd	cd
3	cef	cef
4	cef	cef
5	bcd	bcd
6	bcd	bcd
7	bcdg	bcd
8	bde	bde
9	bd	bd
10	bh	b
11	bi	b
12	b	b

2.1

Header Table 1

Item	Frequency
b	8
c	7
d	6
e	3
f	2

1. Scan DB once to identify frequent items (1-itemsets)
2. Scan DB again:
   - 2.1 Keep frequent items only; sort them within itemsets by descending frequency
   - 2.2 Does path with common prefix exist? *Yes*: Increment counter; append suffix; *No*: Create new branch

FP-Tree Construction

minSup=2/12

Database

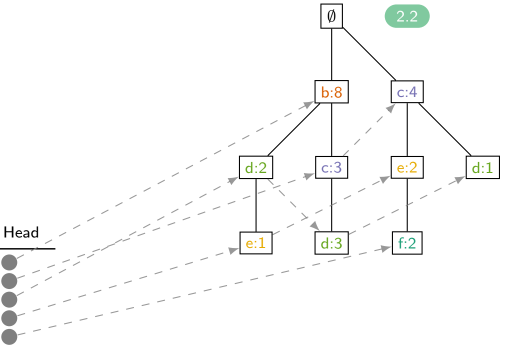
2.1

TID	Items	Freq. Item
1	c	c
2	cd	cd
3	cef	cef
4	cef	cef
5	bcd	bcd
6	bcd	bcd
7	bcdg	bcd
8	bde	bde
9	bd	bd
10	bh	b
11	bi	b
12	b	b

Header Table

1

Item	Frequency	Head
b	8	●
c	7	●
d	6	●
e	3	●
f	2	●



1. Scan DB once to identify frequent items (1-itemsets)
2. Scan DB again:
   2.1 Keep frequent items only; sort them within itemsets by descending frequency
   2.2 Does path with common prefix exist?
   Yes: Increment counter; append suffix;
   No: Create new branch

Benefits of the FP-Tree Structure

Completeness

- never breaks a long pattern of any transaction
- preserves complete information for frequent pattern mining

Compactness

- reduce irrelevant information – infrequent items are gone
- frequency descending ordering: more frequent items are more likely to be shared
- never be larger than the original database (if not count node-links and counts)
- Experiments demonstrate compression ratios over 100

Mining Frequent Patterns Using FP-Tree

General Idea: (Divide-and-Conquer)

Recursively grow frequent pattern path using the FP-tree

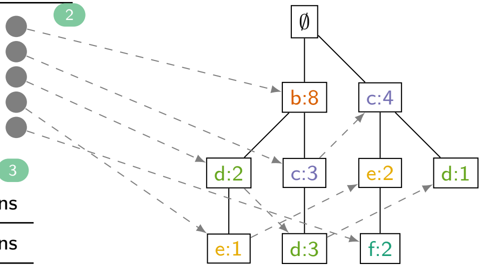
Method

1. Construct conditional pattern base for each node in the FP-tree
2. Construct conditional FP-tree from each conditional pattern-base
3. Recursively mine conditional FP-trees and grow frequent patterns obtained so far; If the conditional FP-tree contains a single path, simply enumerate all the patterns

Major Steps to Mine FP-Tree: Conditional Pattern Base

Header Table 1

Item	Frequency	Head
b	8	●
c	7	●
d	6	●
e	3	●
f	2	●



Conditional Pattern 3

Item	Cond. Patterns
Item	Cond. Patterns
b	∅
c	b:3, ∅
d	bc:3, b:2, c:1
e	c:2, bd:1
f	ce:2

1. Start from header table
2. Visit all nodes for this item (following links)
3. Accumulate all transformed prefix paths to form conditional pattern base (the frequency can be read from the node).

Properties of FP-Tree for Conditional Pattern Bases

Node-Link Property

For any frequent item a_i, all the possible frequent patterns that contain a_i can be obtained by following a_i's node-links, starting from a_i's head in the FP-tree header.

Prefix Path Property

To calculate the frequent patterns for a node a_i in a path P, only the prefix sub-path of a_i in P needs to be accumulated, and its frequency count should carry the same count as node a_i.

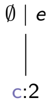
Major Steps to Mine FP-Tree: Conditional FP-Tree

Conditional Pattern

Item	Cond. Patterns
b	∅
c	b:3, ∅
d	bc:3, b:2, c:1
e	c:2, bd:1
f	ce:2

Example: *e*-conditional FP-Tree

Item	Frequency
c	2
b	**1**
d	**1**



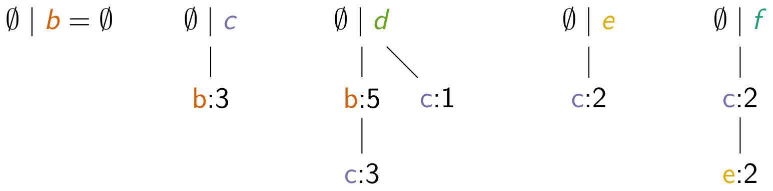
Construct conditional FP-tree from each conditional pattern-base

- The prefix paths of a suffix represent the conditional basis $\rightsquigarrow$ can be regarded as transactions of a database.
- For each pattern-base:
  - Accumulate the count for each item in the base
  - Re-sort items within sets by frequency
  - Construct the FP-tree for the frequent items of the pattern base

Major Steps to Mine FP-Tree: Conditional FP-Tree

- Build conditional FP-Trees for each item

Item	Cond. Patterns
b	∅
c	b:3, ∅
d	bc:3, b:2, c:1
e	c:2, bd:1
f	ce:2



Major Steps to Mine FP-Tree: Recursion

Base Case: Single Path

If the conditional FP-tree contains a single path, simply enumerate all the patterns (enumerate all combinations of sub-paths)

Example

$$\begin{tikzcd} \emptyset \mid f \arrow[d, no head] & \\ c{:}2 \arrow[d, no head] \arrow[r, phantom, "\rightsquigarrow"] & {} \\ e{:}2 & \end{tikzcd}$$

All frequent patterns concerning f:

f,

fc, fe

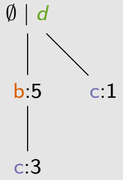
fce

Major Steps to Mine FP-Tree: Recursion

Recursive Case: Non-degenerated Tree

If the conditional FP-tree is not just a single path, create conditional pattern base for this smaller tree, and recurse.

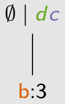
Example



Conditional Pattern Base

Item	Cond. Patterns
b	$\emptyset$
c	b:3, $\emptyset$

$\emptyset \mid d\,b = \emptyset$



Principles of Frequent Pattern Growth

Pattern Growth Property

Let X be a frequent itemset in D, B be X's conditional pattern base, and Y be an itemset in B. Then $X \cup Y$ is a frequent itemset in D if and only if Y is frequent in B.

Example

"abcdef" is a frequent pattern, if and only if

- "abcde" is a frequent pattern, and
- "f" is frequent in the set of transactions containing "abcde"

Why Is Frequent Pattern Growth Fast?

Performance study[1] shows: FP-growth is much faster than Apriori, and is also faster than tree-projection

Reasoning:

- No candidate generation, no candidate test (Apriori algorithm has to proceed breadth-first)
- Use compact data structure
- Eliminate repeated database scan
- Basic operation is counting and FP-tree building

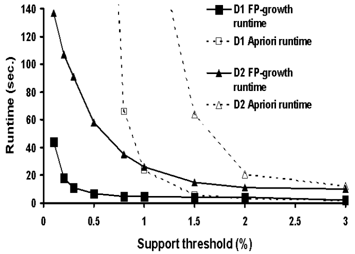


Image Source: [1]

[5]Han, Pei & Yin, *Mining frequent patterns without candidate generation*, SIGMOD'00

Maximal or Closed Frequent Itemsets

Challenge

Often, there is a huge number of frequent itemsets (especially if minSup is set too low), e.g. a frequent itemset of length 100 contains $2^{100} - 1$ many frequent subsets

Closed Frequent Itemset

Itemset X is *closed* in dataset D if for all $Y \supset X : supp(Y) < supp(X)$.

$\Rightarrow$ The set of closed frequent itemsets contains complete information regarding its corresponding frequent itemsets.

Maximal Frequent Itemset

Itemset X is *maximal* in dataset D if for all $Y \supset X : supp(Y) < minSup$.

$\Rightarrow$ The set of maximal itemsets does not contain the complete support information

$\Rightarrow$ More compact representation

Agenda

Simple Association Rules: Introduction

Example

Transaction database:

$$D = \{\{butter, bread, milk, sugar\}, \\ \{butter, flour, milk, sugar\}, \\ \{butter, eggs, milk, salt\}, \\ \{eggs\}, \\ \{butter, flour, milk, salt, sugar\}\}$$

Frequent itemsets:

items	support
{butter}	4
{milk}	4
{butter, milk}	4
{sugar}	3
{butter, sugar}	3
{milk, sugar}	3
{butter, milk, sugar}	3

Question of interest

- If milk and sugar are bought, will the customer always buy butter as well?
  milk, sugar $\Rightarrow$ butter?
- In this case, what would be the probability of buying butter?

Simple Association Rules: Basic Notions

Let *Items, Itemset, Database, Transaction, Transaction Length, k-itemset, (relative) Support, Frequent Itemset* be defined as before. Additionally:

- The items in transactions and itemsets are **sorted** lexicographically: itemset $X = (x_1, \dots, x_k)$, where $x_1 \leq, \dots, \leq x_k$
- **Association rule**: An association rule is an implication of the form $X \Rightarrow Y$ where $X, Y \subseteq I$ are two itemsets with $X \cap Y = \emptyset$
- Note: simply enumerating all possible association rules is not reasonable!

What are the interesting association rules w.r.t. D?

Interestingness of Association Rules

Goal

Quantify the interestingness of an association rule with respect to a transaction database D.

Support

- Frequency (probability) of the entire rule with respect to D:

$$supp(X \Rightarrow Y) = P(X \cup Y) = \frac{|\{T \in D \mid X \cup Y \subseteq T\}|}{|D|} = supp(X \cup Y)$$

- "Probability that a transaction in D contains the itemset."

Interestingness of Association Rules

Confidence

- Indicates the strength of implication in the rule:

$$conf(X \Rightarrow Y) = \frac{supp(X \cup Y)}{supp(X)} \overset{(*)}{=} \frac{P(X \cap Y)}{P(X)} = P(Y \mid X)$$

  (*) Note that the support of the union of the items in X and Y, i.e. $supp(X \cup Y)$ can be interpreted by the joint probability $P(X \cap Y)$

- $P(Y \mid X) =$ conditional probability that a transaction in D containing the itemset X also contains itemset Y

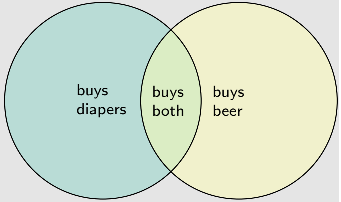
Interestingness of Association Rules

Rule form

"Body $\Rightarrow$ Head [support, confidence]"

Association rule examples

- buys diapers $\Rightarrow$ buys beer [0.5 %, 60%]
- major in CS $\wedge$ takes DB $\Rightarrow$ avg. grade A [1%, 75%]



Mining of Association Rules

Task of mining association rules

Given a database D, determine all association rules having a $supp \geq minSup$ and a $conf \geq minConf$ (so-called *strong association rules*).

Key steps of mining association rules

1. Find frequent itemsets, i.e., itemsets that have $supp \geq minSup$ (e.g. Apriori, FP-growth)
2. Use the frequent itemsets to generate association rules
   - For each itemset X and every nonempty subset $Y \subset X$ generate rule $Y \Rightarrow (X \setminus Y)$ if $minSup$ and $minConf$ are fulfilled
   - We have $2^{|X|} - 2$ many association rule candidates for each itemset X

Mining of Association Rules

Example

- Frequent itemsets:

1-itemset	**count**	**2-itemset**	**count**	**3-itemset**	**count**
{ a }	3	{ a,b }	3	{ a,b,c }	2
{ b }	4	{ a,c }	2		
{ c }	5	{ b,c }	4		

- Rule candidates
  - From 1-itemsets: None
  - From 2-itemsets: $a \Rightarrow b$; $b \Rightarrow a$; $a \Rightarrow c$; $c \Rightarrow a$; $b \Rightarrow c$; $c \Rightarrow b$
  - From 3-itemsets: $a, b \Rightarrow c$; $a, c \Rightarrow b$; $c, b \Rightarrow a$; $a \Rightarrow b, c$; $b \Rightarrow a, c$; $c \Rightarrow a, b$

Generating Rules from Frequent Itemsets

Rule generation

- For each frequent itemset X:
  - For each nonempty subset Y of X, form a rule $Y \Rightarrow (X \setminus Y)$
  - Delete those rules that do not have minimum confidence
- Note:
  - Support always exceeds *minSup*
  - The support values of the frequent itemsets suffice to calculate the confidence
- Exploit anti-monotonicity for generating candidates for strong association rules!
  - $Y \Rightarrow Z$ not strong $\implies$ for all $A \subseteq D : Y \Rightarrow Z \cup A$ not strong
  - $Y \Rightarrow Z$ not strong $\implies$ for all $Y' \subseteq Y$: $(Y \setminus Y') \Rightarrow (Z \cup Y')$ not strong

Generating Rules from Frequent Itemsets

Example: $minConf = 60\%$

$conf(a \Rightarrow b) = 3/3 = 1$ ✓

$conf(b \Rightarrow a) = 3/4$ ✓

$conf(a \Rightarrow c) = 2/3$ ✓

$conf(c \Rightarrow a) = 2/5$ ✗

$conf(b \Rightarrow c) = 4/4 = 1$ ✓

$conf(c \Rightarrow b) = 4/5$ ✓

$conf(a, b \Rightarrow c) = 2/3$ ✓

$conf(a, c \Rightarrow b) = 2/2 = 1$ ✓

$conf(b, c \Rightarrow a) = 2/4 = .5$ ✗

$conf(a \Rightarrow b, c) = 2/3$ ✓

$conf(b \Rightarrow a, c) = 2/4$ ✗ (pruned wrt. $b, c \Rightarrow a$)

$conf(c \Rightarrow a, b) = 2/5$ ✗ (pruned wrt. $b, c \Rightarrow a$)

itemset	count
{ a }	3
{ b }	4
{ c }	5
{ a,b }	3
{ a,c }	2
{ b,c }	4
{ a,b,c }	2

Interestingness Measurements

Objective measures

Two popular measures:

- Support
- Confidence

Subjective measures [Silberschatz & Tuzhilin, KDD95]

A rule (pattern) is interesting if it is

- *unexpected* (surprising to the user) and/or
- *actionable* (the user can do something with it)

Criticism to Support and Confidence

Example 1 [Aggarwal & Yu, PODS98]

- Among 5000 students
  - 3000 play basketball (=60%)
  - 3750 eat cereal (=75%)
  - 2000 both play basket ball and eat cereal (=40%)
- Rule "play basketball ⇒ eat cereal [40%, 66.7%]" is misleading because the overall percentage of students eating cereal is 75% which is higher than 66.7%
- Rule "play basketball ⇒ not eat cereal [20%, 33.3%]" is far more accurate, although with lower support and confidence
- Observation: "play basketball" and "eat cereal" are negatively correlated

Not all strong association rules are interesting and some can be misleading.

- Augment the support and confidence values with interestingness measures such as the correlation: "A ⇒ B [$supp$, $conf$, $corr$]"

Other Interestingness Measures: Correlation

Correlation

Correlation (sometimes called *Lift*) is a simple measure between two items A and B:

$$corr_{A,B} = \frac{P(A \cap B)}{P(A)P(B)} = \frac{P(B \mid A)}{P(B)} = \frac{conf(A \Rightarrow B)}{supp(B)}$$

- The two rules $A \Rightarrow B$ and $B \Rightarrow A$ have the same correlation coefficient
- Takes both $P(A)$ and $P(B)$ in consideration
- $corr_{A,B} > 1$: The two items A and B are positively correlated
- $corr_{A,B} = 1$: There is no correlation between the two items A and B
- $corr_{A,B} < 1$: The two items A and B are negatively correlated

Other Interestingness Measures: Correlation

Example 2

T	item		
	X	**Y**	**Z**
	1	1	0
	1	1	1
	1	0	1
	1	0	1
	0	0	1
	0	0	1
	0	0	1
	0	0	1

rule	support	confidence	correlation
$X \Rightarrow Y$	25%	50%	2
$X \Rightarrow Z$	37.5%	75%	0.89
$Y \Rightarrow Z$	12.5%	50%	0.57

- X and Y: positively correlated
- X and Z: negatively related
- Support and confidence of $X \Rightarrow Z$ dominates
- But: items X and Z are negatively correlated
- Items X and Y are positively correlated

Hierarchical Association Rules: Motivation

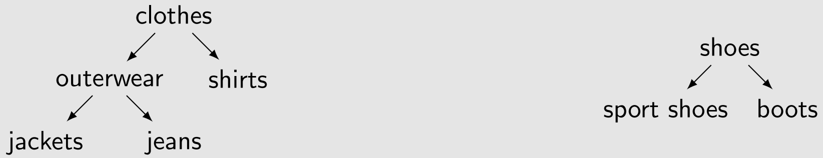
Problem

- High minSup: apriori finds only few rules
- Low minSup: apriori finds unmanagably many rules

Solution

Exploit item taxonomies (generalizations, is-a hierarchies) which exist in many applications

Example



Hierarchical Association Rules

New Task

Find all generalized association rules between generalized items, i.e. Body and Head of a rule may have items of any level of the hierarchy

Generalized Association Rule

$X \Rightarrow Y$ with $X, Y \subset I, X \cap Y = \emptyset$ and no item in Y is an ancestor of any item in X

Example

- Jeans $\Rightarrow$ Boots; supp $<$ minSup
- Jackets $\Rightarrow$ Boots; supp $<$ minSup
- Outerwear $\Rightarrow$ Boots; supp $>$ minSup

Hierarchical Association Rules: Characteristics

$$\begin{tikzcd}
& & Y \arrow[dll] \arrow[dl] \arrow[d] \arrow[dr] \arrow[drr] & & \\
X_1 & \cdots & X_i & \cdots & X_k
\end{tikzcd}$$

Characteristics

Let $Y = \biguplus_{i=1}^{k} X_i$ be a generalisation.

- For all $1 \leq i \leq k$ it holds $supp(Y \Rightarrow Z) \geq supp(X_i \Rightarrow Z)$
- In general, $supp(Y \Rightarrow Z) = \sum_{i=1}^{k} supp(X_i \Rightarrow Z)$ does not hold (a transaction might contain elements from multiple low-level concepts, e.g. boots *and* sport shoes).

Mining Multi-Level Associations

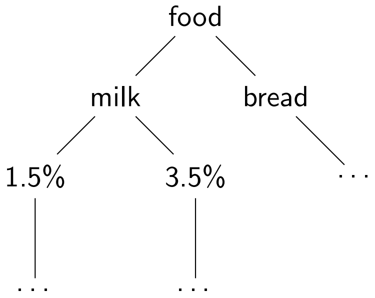
Top-Down, Progressive-Deepening Approach

1. First find high-level strong rules, e.g. milk $\Rightarrow$ bread [20%, 60%]
2. Then find their lower-level "weaker" rules, e.g. low-fat milk $\Rightarrow$ wheat bread [6%, 50%].

Support Threshold Variants

Different minSup threshold across multi-levels lead to different algorithms:

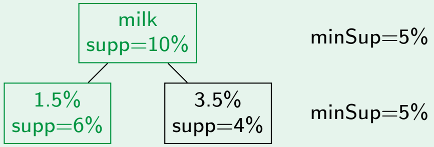
- adopting the same minSup across multi-levels
- adopting reduced minSup at lower levels



Minimum Support for Multiple Levels

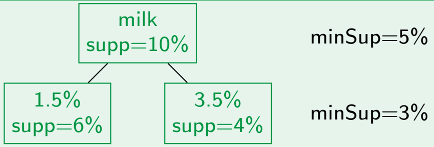
Uniform Support

- Search procedure is simplified (monotonicity)
- User only specifies one threshold



Reduced Support (Variable Support)

- Takes into account lower frequency of items in lower levels



Multilevel Association Mining using Reduced Support

Level-by-level independent method

Examine each node in the hierarchy, regardless of the frequency of its parent node.

Level-cross-filtering by single item

Examine a node only if its parent node at the preceding level is frequent.

Level-cross-filtering by k-itemset

Examine a k-itemset at a given level only if its parent k-itemset at the preceding level is frequent.

Multi-level Association: Redundancy Filtering

Some rules may be redundant due to ”ancestor” relationships between items.

Example

- R_1: milk ⇒ wheat bread [8%, 70%]
- R_2: 1.5% milk ⇒ wheat bread [2%, 72%]

We say that rule 1 is an ancestor of rule 2.

Redundancy

A rule is redundant if its support is close to the ”expected” value, based on the rule’s ancestor.

’

Interestingness of Hierarchical Association Rules: Notions

Let $X, X', Y, Y' \subseteq I$ be itemsets.

- X' is ancestor of X iff there exists ancestors $x'_1, \ldots, x'_k$ of $x_1, \ldots, x_k \in X$ and $x_{k+1}, \ldots, x_n$ with $n = |X|$ such that $X' = \{x'_1, \ldots, x'_k, x_{k+1}, \ldots, x_n\}$
- Let X' and Y' be ancestors of X and Y. Then we call the rules $X' \Rightarrow Y'$, $X \Rightarrow Y'$, and $X' \Rightarrow Y$ ancestors of the rule $X \Rightarrow Y$.
- The rule $X' \Rightarrow Y'$ is a direct ancestor of rule $X \Rightarrow Y$ in a set of rules if:
  1. Rule $X' \Rightarrow Y'$ is an ancestor of rule $X \Rightarrow Y$, and
  2. There is no rule $X'' \Rightarrow Y''$ being ancestor of $X \Rightarrow Y$ and $X' \Rightarrow Y'$ is an ancestor of $X'' \Rightarrow Y''$

R-Interestingness

R-Interestingness

A hierarchical association rule $X \Rightarrow Y$ is called R-interesting if:

- There are no direct ancestors of $X \Rightarrow Y$ or
- The actual support is larger than R times the expected support or
- The actual confidence is larger than R times the expected confidence

Example in tutorial

R-Interestingness: Expected Support

Given the rule for $X \Rightarrow Y$ and its ancestor rule $X' \Rightarrow Y'$ the expected support of $X \Rightarrow Y$ is defined as:

$$\mathbb{E}_{Z'}[P(Z)] = P(Z') \cdot \prod_{i=1}^{j} \frac{P(y_i)}{P(y_i)'}$$

where $Z = X \cup Y = \{z_1, \ldots, z_n\}$, $Z' = X' \cup Y' = \{z'_1, \ldots, z'_j, z_{j+1}, \ldots, z_n\}$ and each $z'_i \in Z'$ is an ancestor of $z_i \in Z$.

R. Srikant, R. Agrawal: Mining Generalized Association Rules. In VLDB, 1995.

R-Interestingness: Expected Confidence

Given the rule for $X \Rightarrow Y$ and its ancestor rule $X' \Rightarrow Y'$, then the expected confidence of $X \Rightarrow Y$ is defined as:

$$\mathbb{E}_{X' \Rightarrow Y'}[P(Y|X)] = P(Y' \mid X') \cdot \prod_{i=1}^{j} \frac{P(y_i)}{P(y_i)'}$$

where $Y = \{y_1, \ldots, y_n\}$ and $Y' = \{y'_1, \ldots, y'_j, y_{j+1}, \ldots, y_n\}$ and each $y'_i \in Y'$ is an ancestor of $y_i \in Y$.

R. Srikant, R. Agrawal: Mining Generalized Association Rules. In VLDB, 1995.

Summary Frequent Itemset & Association Rule Mining

- Frequent Itemsets
  - Mining: Apriori algorithm, hash trees, FP-tree
  - support, confidence
- Simple Association Rules
  - Mining: (Apriori)
  - Interestingness measures: support, confidence, correlation
- Hierarchical Association Rules
  - Mining: Top-Down Progressive Deepening
  - Multilevel support thresholds, redundancy, *R*-interestingness
- Further Topics (not covered)
  - Quantitative Association Rules (for numerical attributes)
  - Multi-dimensional association rule mining

Agenda

Motivation

Motivation

- So far we only considered sets of items. In many applications the order of the items is the crucial information.
- The ordering encodes e.g. temporal aspects, patterns in natural language.
- In an ordered sequence, items are allowed to occur more than one time.

Applications

Bioinformatics (DNA/protein sequences), Web mining, text mining (NLP), sensor data mining, process mining, . . .

Sequential Pattern Mining: Basic Notions I

We now consider transactions having an order of the items. Define:

- **Alphabet** Σ is a set of symbols or characters (denoting items)
  e.g. $\Sigma = \{A, B, C, D, E\}$
- **Sequence** $S = s_1 s_2 \dots s_k$ is an ordered list of a length $|S| = k$ items where $s_i \in \Sigma$ is an item at position i also denoted as $S[i]$.
  e.g. $S = CAB, \quad s_3 = B$
- A **k-sequence** is a sequence of length k
  e.g. $S = CAB$ is a 3-sequence
- **Consecutive subsequence** $R = r_1 r_2 \dots r_m$ of $S = s_1 s_2 \dots s_n$ is also a sequence in Σ s.t. $r_1 r_2 \dots r_m = s_j s_{j+1} \dots s_{j+m-1}$, with $1 \leq j \leq n - m + 1$. We say S contains R and denote this by $R \subseteq S$
  e.g. $R_1 = AB \subseteq S = CAB$

Sequential Pattern Mining: Basic Notions II

- In a more general **subsequence** R of S we allow for gaps between the items of R, i.e. the items of the subsequence $R \subseteq S$ must have the same order of the ones in S but there can be some other items between them

  e.g. $R_2 = CB$ is a subsequence of $S = CAB$
- A **prefix** of a sequence S is any consecutive subsequence of the form $S[1:i] = s_1 s_2 \dots s_i$ with $0 \leq i \leq n$, $S[1:0]$ is the empty prefix

  e.g. $R_3 = C, R_4 = CA, R_5 = CAB$ are prefixes of $S = CAB$
- A **suffix** of a sequence S is any consecutive subsequence of the form $S[i:n] = s_i s_{i+1} \dots s_n$ with $1 \leq i \leq n+1$, $S[n+1:n]$ is the empty suffix.

  e.g. $R_4 = AB$ is a suffix of $S = CAB$
- **(Relative) support** of a sequence R in D: $supp(R) = |\{S \in D \mid R \subseteq S\}|/|D|$

Sequential Pattern Mining: Basic Notions III

- S is *frequent (or sequential)* if $supp(S) \geq minSup$ for threshold $minSup$.
- A frequent sequence is *maximal* if it is not a subsequence of any other frequent sequence
- A frequent sequence is *closed* if it is not a subsequence of any other frequent sequence with the same support

Sequential Pattern Mining

Task

Find all frequent subsequences occuring in many transactions.

Difficulty

The number of possible patterns is even larger than for frequent itemset mining!

Example

There are $|\Sigma|^k$ different k-sequences, where $k > |\Sigma|$ is possible and often encountered, e.g. when dealing with DNA sequences where the alphabet only comprises four symbols.

Sequential Pattern Mining Algorithms

Breadth-First Search Based

- GSP (Generalized Sequential Pattern) algorithm[6]
- SPADE[7]
- ...

Depth-First Search Based

- PrefixSpan[8]
- SPAM[9]
- ...

[6] Sirkant & Aggarwal: *Mining sequential patterns: Generalizations and performance improvements*. EDBT 1996

[7] Zaki M J. *SPADE: An efficient algorithm for mining frequent sequences*. Machine learning, 2001, 42(1-2): 31-60.

[8] Pei at. al.: *Mining sequential patterns by pattern-growth: PrefixSpan approach*. TKDE 2004

[9] Ayres, Jay, et al: *Sequential pattern mining using a bitmap representation*. SIGKDD 2002.

GSP (Generalized Sequential Pattern) algorithm

- Breadth-first search: Generate frequent sequences ascending by length
- Given the set of frequent sequences at level k, generate *all* possible sequence extensions or candidates at level $k + 1$
- Uses the Apriori principle (anti-monotonicity)
- Next compute the support of each candidate and prune the ones with $supp(c) < minSup$
- Stop the search when no more frequent extensions are possible

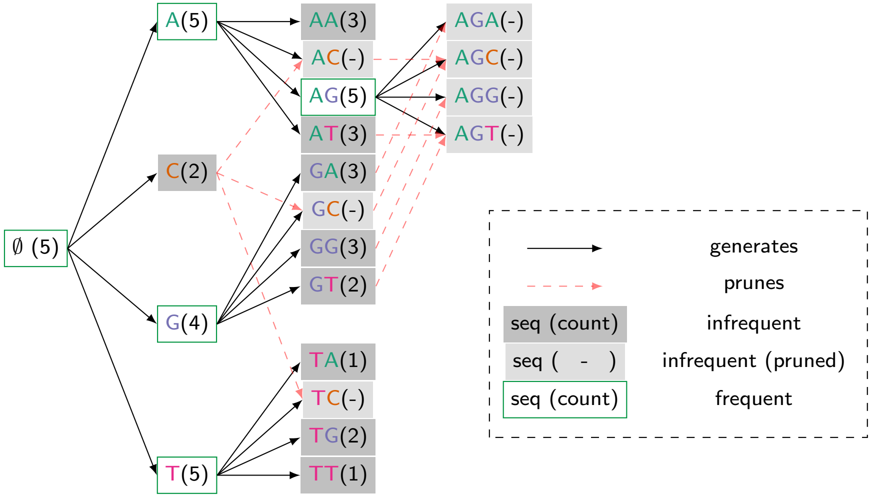
Projection-Based Sequence Mining: PrefixSpan: Representation

- The sequence search space can be organized in a prefix search tree
- The root (level 0) contains the empty sequence with each item $x \in \Sigma$ as one of its children
- A node labelled with sequence: $S = s_1 s_2 \dots s_k$ at level k has children of the form $S' = s_1 s_2 \dots s_k s_{k+1}$ at level $k+1$ (i.e. S is a prefix of S' or S' is an extension of S)

Prefix Search Tree: Example

ID	Sequence
S_1	CAGAAGT
S_2	TGACAG
S_3	GAG
S_4	AGTT
S_5	ATAG

$minSup = .8$



Projected Database

- For a database D and an item $s \in \Sigma$, the projected database w.r.t. s is denoted D_s and is found as follows: For each sequence $S_i \in D$ do
  - Find the first occurrence of s in S_i, say at position p
  - $suff_{S_i,s} \leftarrow suffix(S_i)$ starting at position $p+1$
  - Remove infrequent items from $suff_{S_i,s}$
  - $D_s = D_s \cup suff_{S_i,s}$

Example

$minSup = .8$ (i.e. 4 transactions)

ID	Sequence	D_A	D_G	D_T
S_1	CAGAAGT	GAAGT	AAGT	$\emptyset$
S_2	TGACAG	AG	AAG	GAAG
S_3	GAG	G	AG	-
S_4	AGTT	GTT	TT	T
S_5	ATAG	TAG	$\emptyset$	AG

Projection-Based Sequence Mining: PrefixSpan Algorithm

- The *PrefixSpan* algorithm computes the support for only the individual items in the projected databased D_s
- Then performs recursive projections on the frequent items in a depth-first manner

```
1: Initialization: $D_R \leftarrow D, R \leftarrow \emptyset, \mathcal{F} \leftarrow \emptyset$
2: procedure PrefixSpan($D_R, R, minSup, \mathcal{F}$)
3:     for all $s \in \Sigma$ such that $supp(s, D_R) \geq minSup$ do
4:         $R_s \leftarrow R + s$                          ▷ append $s$ to the end of $R$
5:         $\mathcal{F} \leftarrow \mathcal{F} \cup \{(R_s, sup(s, D_R))\}$        ▷ calculate support of $s$ for each $R_s$ within $D_R$
6:         $D_s \leftarrow \emptyset$
7:         for all $S_i \in D_R$ do
8:             $S_i' \leftarrow$ projection of $S_i$ w.r.t. item $s$
9:             Remove all infrequent symbols from $S_i'$
10:            if $S' \neq \emptyset$ then
11:                $D_s \leftarrow D_s \cup S_i'$
12:        if $D_s \neq \emptyset$ then
13:            PrefixSpan($D_s, R_s, minSup, \mathcal{F}$)
```

PrefixSpan: Example

minSup = 0.8 (i.e. 4 transactions)

$D_\emptyset$

ID	Sequence
S_1	CAGAAGT
S_2	TGACAG
S_3	GAG
S_4	AGTT
S_5	ATAG

A(5)~~C(2)~~G(5)T(4)

D_G

ID	Sequence
S_1	AAGT
S_2	AAG
S_3	AG
S_4	TT
S_5	$\emptyset$

~~A(3)G(3)T(2)~~

D_T

ID	Sequence
S_1	$\emptyset$
S_2	GAAG
-	-
S_4	T
S_5	AG

~~A(2)G(2)T(1)~~

D_A

ID	Sequence
S_1	GAAGT
S_2	AG
S_3	G
S_4	GTT
S_5	TAG

~~A(3)~~G(5)~~T(3)~~

D_{AG}

ID	Sequence
S_1	G
S_2	$\emptyset$
S_3	$\emptyset$
S_4	$\emptyset$
S_5	$\emptyset$

~~G(1)~~

Hence, the frequent sequences are: $\emptyset$, A, G, T, AG

Interval-based Sequential Pattern Mining

Interval-Based Representation

- Deals with the more common interval-based items s (or events).
- Each event has a starting t_s^+ and an ending time point t_s^-, where $t_s^+ < t_s^-$

Application

Health data analysis, Stock market data analysis, etc.

Relationships

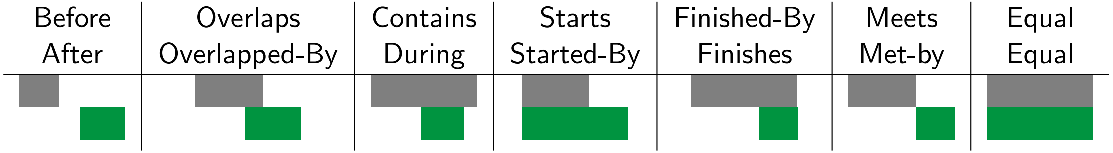
Predefined relationships between items are more complex.

- Point-based relationships: before, after, same time.
- Interval-based relationships: Allen's relations[10], End point representation[11], etc.

[10] Allen: Maintaining knowledge about temporal intervals. In Communications of the ACM 1983

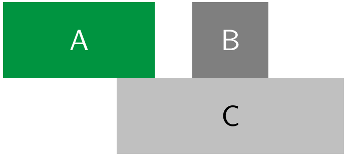
[11] Wu, Shin-Yi, and Yen-Liang Chen: Mining nonambiguous temporal patterns for interval-based events. TKDE 2007

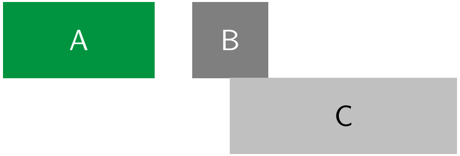
Allen's Relations



Problem

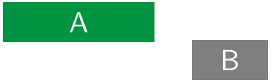
- Allen's relationships only describe the relation between two intervals.
- Describing the relationship between k intervals unambiguously requires $\mathcal{O}(k^2)$ comparisons.



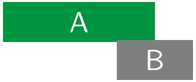


Interval-based Sequential Pattern Mining

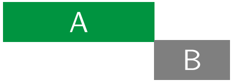
- *TPrefixSpan*[12] converts interval-based sequences into point-based sequences:



$$\{A^+\},\{A^-\},\{B^+\},\{B^-\}$$



$$\{A^+\},\{B^+\},\{A^-\},\{B^-\}$$

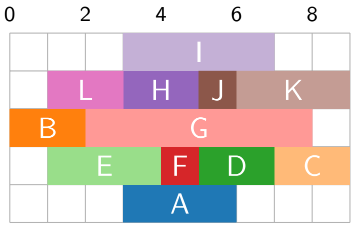


$$\{A^+\},\{A^-,B^+\},\{B^-\}$$

- Similar prefix projection mining approach as PrefixSpan algorithm.
- Validation checking is necessary in each expanding iteration to make sure that the appended time point can form an interval with a time point in the prefix.

[12] Wu, Shin-Yi, and Yen-Liang Chen: Mining nonambiguous temporal patterns for interval-based events. TKDE 2007

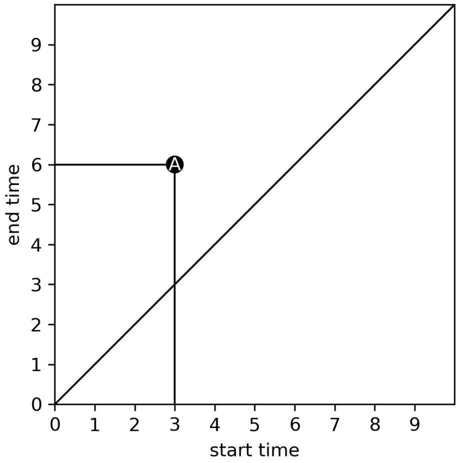
Allen's Relations with *Point Transformation*: Example



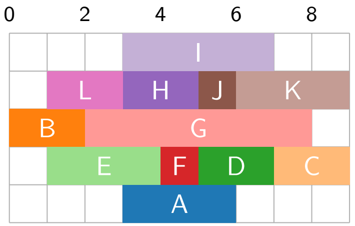
A is the interval starting at time 3 and ending at time 6.

→ Point Transformation maps it in the 2-dim space with $A = (3, 6)$.

A is the reference point in this example!

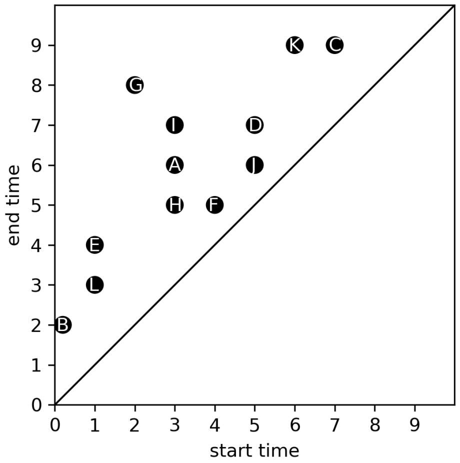


Allen's Relations with *Point Transformation*: Example

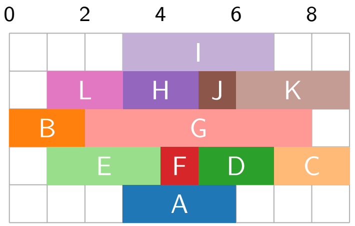


Before: BA
After: CA
Overlaps: DA
Overlapped-By: EA
During: FA
Contains: GA

Started-By: HA
Starts: IA
Finished-By: JA
Finishes: AJ
Met-By: KA
Meets: LA
Equal: AA

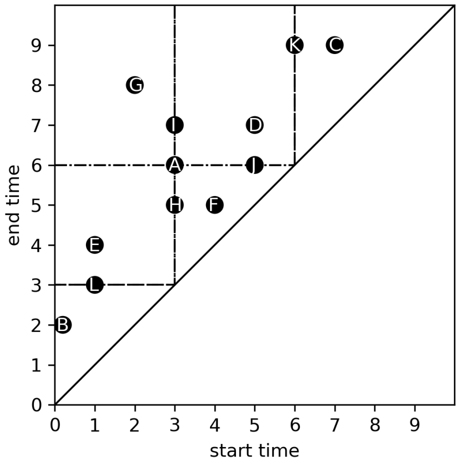


Allen's Relations with *Point Transformation*: Example

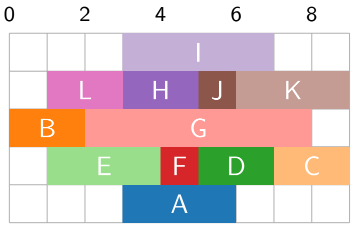


Before: BA
After: CA
Overlaps: DA
Overlapped-By: EA
During: FA
Contains: GA

Started-By: HA
Starts: IA
Finished-By: JA
Finishes: AJ
Met-By: KA
Meets: LA
Equal: AA

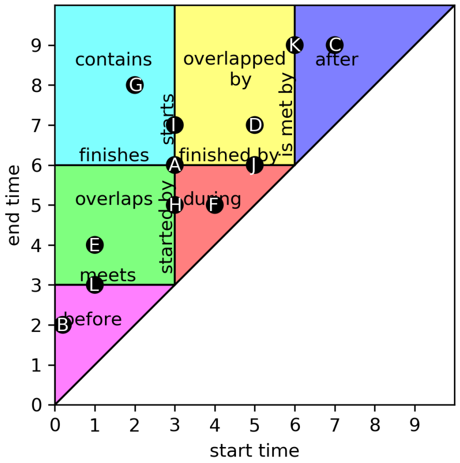


Allen's Relations with *Point Transformation*: Example

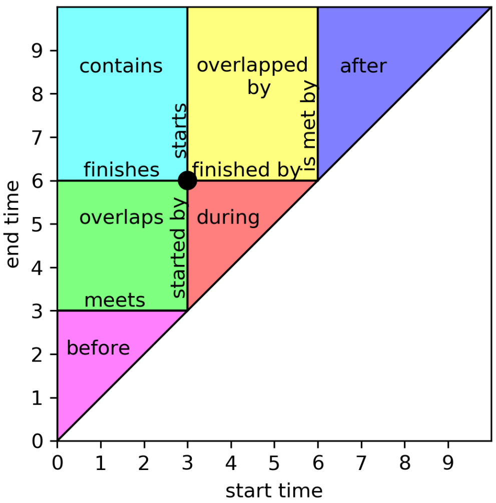


Before: BA
After: CA
Overlaps: DA
Overlapped-By: EA
During: FA
Contains: GA

Started-By: HA
Starts: IA
Finished-By: JA
Finishes: AJ
Met-By: KA
Meets: LA
Equal: AA



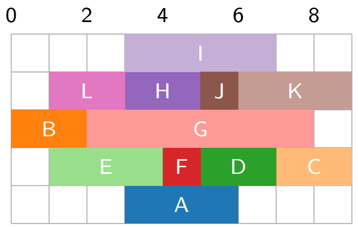
Allen's Relations with *Point Transformation*: Example

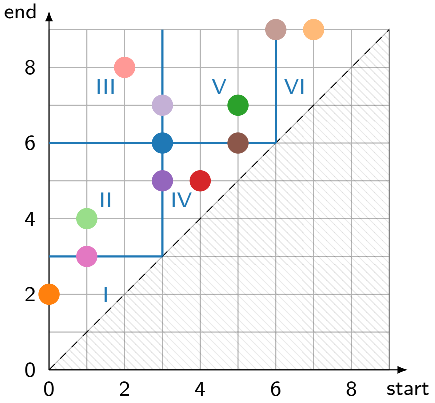


An Open Issue: Considering Timing Information

Idea

Learn pattern from data by clustering, e.g. QTempIntMiner[13], Event Space Miner[14], PIVOTMiner[15]





[13] Guyet, T., & Quiniou, R.: *Mining temporal patterns with quantitative intervals*. ICDMW 2008

[14] Ruan, G., Zhang, H., & Plale, B.: *Parallel and quantitative sequential pattern mining for large-scale interval-based temporal data*. IEEE Big Data 2014

[15] Hassani M., Lu Y. & Seidl T.: *A Geometric Approach for Mining Sequential Patterns in Interval-Based Data Streams*. FUZZ-IEEE 2016