Agenda

1. Introduction

2. Basics

3. Supervised Methods

4. Unsupervised Methods
   4.1 Clustering
      - Introduction
      - Partitioning Methods
      - Probabilistic Model-Based Methods
      - Density-Based Methods
      - Mean-Shift
      - Spectral Clustering
      - Hierarchical Methods
      - Evaluation

4.2 Outlier Detection

4.3 Frequent Pattern Mining
Agenda

1. Introduction

2. Basics

3. Supervised Methods

4. Unsupervised Methods
   4.1 Clustering
      - Introduction
      - Partitioning Methods
      - Probabilistic Model-Based Methods
      - Density-Based Methods
      - Mean-Shift
      - Spectral Clustering
      - Hierarchical Methods
      - Evaluation

4.2 Outlier Detection

4.3 Frequent Pattern Mining
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

4. Unsupervised Methods 4.1 Clustering
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

1. Introduction

2. Basics

3. Supervised Methods

4. Unsupervised Methods
   4.1 Clustering
      - Introduction
      - Partitioning Methods
      - Probabilistic Model-Based Methods
      - Density-Based Methods
      - Mean-Shift
      - Spectral Clustering
      - Hierarchical Methods
      - Evaluation

4.2 Outlier Detection

4.3 Frequent Pattern Mining
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**

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

**Objective**

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

---

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

---

4. Unsupervised Methods

4.1 Clustering
Objects $p = (p_1, \ldots, p_d)$ are points in a $d$-dimensional vector space (the mean $\mu_S$ of a set of points $S$ must be defined: $\mu_S = \frac{1}{|S|} \sum_{p \in S} p$)

Measure for the compactness of a cluster $C_j$ (sum of squared distances):

$$SSE(C_j) = \sum_{p \in C_j} ||p - \mu_{C_j}||^2$$

Measure for the compactness of a clustering $C$:

$$SSE(C) = \sum_{C_j \in C} SSE(C_j) = \sum_{p \in D} ||p - \mu_{C(p)}||^2$$

Optimal Partitioning: $\arg\min_C SSE(C)$

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**

<table>
<thead>
<tr>
<th>Start</th>
<th>Update</th>
<th>Reassign</th>
<th>Update</th>
<th>Reassign</th>
</tr>
</thead>
</table>
**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

4. Unsupervised Methods

4.1 Clustering
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).
If there is an ordering on the data use median instead of mean.
Compute median separately per dimension (\(\leadsto\) efficient computation)
### K-Means/Median/Mode/Medoid Clustering: Discussion

<table>
<thead>
<tr>
<th></th>
<th>$k$-Means</th>
<th>$k$-Median</th>
<th>$k$-Mode</th>
<th>$k$-Medoid</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Data</strong></td>
<td>numerical (mean)</td>
<td>ordinal</td>
<td>categorical</td>
<td>metric</td>
</tr>
<tr>
<td><strong>Efficiency</strong></td>
<td>high $\mathcal{O}(tkn)$</td>
<td></td>
<td>low $\mathcal{O}(tk(n-k)^2)$</td>
<td></td>
</tr>
<tr>
<td><strong>Sensitivity to Outliers</strong></td>
<td>high</td>
<td>low</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **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 $O(\log k)$ (standard $k$-means can generate arbitrarily bad clusterings)

- In general: Repeat with different initial centers and choose result with lowest clustering error

---

Choice of the Parameter $k$

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

### Silhouette-Coefficient

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

---

1Rousseeuw, 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

- \( a(o) = "\text{Avg. distance between } o \text{ and objects in its cluster } A." \)
  \[
a(o) = \frac{1}{|C(o)|} \sum_{p \in C(o)} d(o, p)
\]

- \( b(o): "\text{Smallest avg. distance between } o \text{ and objects in other cluster."} \)
  \[
b(o) = \min_{C_i \neq C(o)} \left\{ \frac{1}{|C_i|} \sum_{p \in C_i} d(o, p) \right\}
\]
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 $C = (C_1, \ldots, C_k)$ is defined as

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

where $D$ denotes the whole dataset
The Silhouette Coefficient

"Reading" the silhouette coefficient: Let $a(o) \neq 0$

- $b(o) \gg a(o) \iff s(o) \approx 1$: good assignment of $o$ to its cluster $A$
- $b(o) \approx a(o) \iff s(o) \approx 0$: $o$ is in-between $A$ and $B$
- $b(o) \ll a(o) \iff s(o) \approx -1$: bad, on average $o$ is closer to members of $B$

Silhouette coefficient $s(C)$ of a clustering: Average silhouette of all objects

- $0.7 < s(C) \leq 1.0$: strong structure
- $0.5 < s(C) \leq 0.7$: medium structure
- $0.25 < s(C) \leq 0.5$: weak structure
- $s(C) \leq 0.25$: no structure
Silhouette Coefficient: Example

dataset with 10 clusters

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

1. Introduction

2. Basics

3. Supervised Methods

4. Unsupervised Methods
   4.1 Clustering
      - Introduction
      - Partitioning Methods
      - Probabilistic Model-Based Methods
      - Density-Based Methods
      - Mean-Shift
      - Spectral Clustering
      - Hierarchical Methods
      - Evaluation
   4.2 Outlier Detection
   4.3 Frequent Pattern Mining
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 → 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

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

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

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 $\theta$ is given as:

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

**Goal**

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

$$\theta_{ML} = \arg\max_{\theta} \{ p(X | \theta) \}$$
EM: Maximum Likelihood Estimation

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

\[ \theta_{ML} = \arg \max_{\theta} \{ p(X \mid \theta) \} = \arg \max_{\theta} \{ \log p(X \mid \theta) \} \]

- The log-likelihood can be written as

\[ \log p(X \mid \theta) = \log \prod_{i=1}^{n} \sum_{l=1}^{k} \pi_l \cdot p(x_i \mid \mu_l, \Sigma_l) \]

\[ = \sum_{i=1}^{n} \log \sum_{l=1}^{k} \pi_l \cdot p(x_i \mid \mu_l, \Sigma_l) \]

- Maximization w.r.t. the means:

\[ \frac{\partial \log p(X \mid \theta)}{\partial \mu_j} = 0 \]
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 $\theta_{ML}$:

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

- 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 $\mu_j$ and $\gamma_j(x_i)$ independently
Iterative Optimization

1. Initialize means $\mu_j$, covariances $\Sigma_j$, and mixing coefficients $\pi_j$ and evaluate the initial log-likelihood.

2. **E-step**: Evaluate the responsibilities using the current parameter values:

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

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

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

...
EM: Iterative Optimization

Iterative Optimization

\[
\Sigma_{\text{new}} = \frac{\sum_{i=1}^{n} \gamma_{j}^{\text{new}}(x_i)(x_i - \mu_{j}^{\text{new}})(x_i - \mu_{j}^{\text{new}})^T}{\sum_{i=1}^{n} \gamma_{j}^{\text{new}}(x_i)}
\]

\[
\pi_{j}^{\text{new}} = \frac{\sum_{i=1}^{n} \gamma_{j}^{\text{new}}(x_i)}{\sum_{k=1}^{k} \sum_{i=1}^{n} \gamma_{i}^{\text{new}}(x_i)}
\]

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

4. Unsupervised Methods
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.

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

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

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: $O(tnk)$
  - $t$ 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 $\theta_k$ for $k \in \{k_{\text{min}}, \ldots, k_{\text{max}}\}$ and select the model according to some quality measure $\text{qual}$:

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

- Silhouette Coefficient (as for $k$-Means) only works for partitioning approaches
- The likelihood is nondecreasing in $k$
Deterministic or stochastic model selection methods which try to balance the goodness of fit with simplicity.

- **Deterministic:**
  
  \[
  \text{qual}(\theta_k) = \log p(X \mid \theta_k) - P(k)
  \]

  where \(P(k)\) is an increasing function penalizing higher values of \(k\)

- **Stochastic:** Based on Markov Chain Monte Carlo (MCMC)
1. Introduction

2. Basics

3. Supervised Methods

4. Unsupervised Methods
   4.1 Clustering
   - Introduction
   - Partitioning Methods
   - Probabilistic Model-Based Methods
     - Density-Based Methods
   - Mean-Shift
   - Spectral Clustering
   - Hierarchical Methods
   - Evaluation

4.2 Outlier Detection

4.3 Frequent Pattern Mining
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:

- $\epsilon$-radius for the neighborhood of point $q$

$$N_\epsilon(q) = \{p \in D \mid dist(p, q) \leq \epsilon\}$$  \hspace{1cm} (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. $\epsilon$, $MinPts$ if $|N_\epsilon(q)| \geq minPts$
Density-Based Clustering: Basic Definitions

(Directly) Density-Reachable

$p$ directly density-reachable from $q$ w.r.t. $\epsilon$, $MinPts$ if:

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

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. $\epsilon$, $MinPts$ if there is a point $o$ such that both, $p$ and $q$ are density-reachable from $o$ w.r.t. $\epsilon$, $MinPts$
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

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 Spatial Clustering of Applications with Noise\footnote{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.}

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

\textbf{Note}

Density-reachable objects are collected by performing successive $\epsilon$-neighborhood queries.
DBSCAN: Example

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

Parameters: $\epsilon = 1.75$, $\text{minPts} = 3$. Clusters: $C_1$; Noise: $N$
DBSCAN: Example

Parameters: $\epsilon = 1.75$, $minPts = 3$. Clusters: $C_1$, $C_2$; Noise: $N$
## Determining the Parameters $\epsilon$ and $MinPts$

<table>
<thead>
<tr>
<th>Recap</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cluster: Point density higher than specified by $\epsilon$ and $MinPts$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Idea</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use the point density of the least dense cluster in the data set as parameters.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Problem</th>
</tr>
</thead>
<tbody>
<tr>
<td>How to determine this?</td>
</tr>
</tbody>
</table>
Determining the Parameters $\epsilon$ 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: $\epsilon$ is set to $MinPts$-distance($o$).
Determining the Parameters $\epsilon$ and $MinPts$: Problematic Example

4. Unsupervised Methods

4.1 Clustering
Standard DBSCAN evaluation is based on recursive database traversal. Böhm et al.\textsuperscript{2} observed that DBSCAN, among other clustering algorithms, may be efficiently built on top of similarity join operations.

**$\epsilon$-Similarity Join**

An $\epsilon$-similarity join yields all pairs of $\epsilon$-similar objects from two data sets $Q$, $P$:

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

**SQL Query**

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

---

Database Support for Density-Based Clustering

**$\epsilon$-Similarity Self-Join**

An $\epsilon$-similarity self join yields all pairs of $\epsilon$-similar objects from a database $D$.

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

**SQL Query**

```sql
SELECT * FROM D q, D p WHERE dist(q, p) \leq \epsilon
```
Database Support for Density-Based Clustering

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

\[
ddr_{\epsilon,\mu} = \{(q, p) \in D \times D \mid q \text{ is } \epsilon, \mu\text{-core-point} \land p \in N_\epsilon(q)\}
\]
\[
= \{(q, p) \in D \times D \mid \text{dist}(q, p) \leq \epsilon \land \exists_{\geq \mu} p' \in D : \text{dist}(q, p') \leq \epsilon\}
\]
\[
= \{(q, p) \in D \times D \mid (q, p) \in D \bowtie_\epsilon D \land \exists_{\geq \mu} p'(q, p') \in D \bowtie_\epsilon D\}
\]
\[
= \sigma_{|\pi_q(D \bowtie_\epsilon D)|_{\geq \mu}}(D \bowtie_\epsilon D) =: D \bowtie_{\epsilon,\mu} D
\]

**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: $N_{\epsilon}$-query: $O(n)$, DBSCAN: $O(n^2)$.
- Can be supported by spatial index structures ($\sim N_{\epsilon}$-query: $O(\log n)$)

## Disadvantages

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

2. Basics

3. Supervised Methods

4. Unsupervised Methods
   4.1 Clustering
      - Introduction
      - Partitioning Methods
      - Probabilistic Model-Based Methods
      - Density-Based Methods
      - Mean-Shift
      - Spectral Clustering
      - Hierarchical Methods
      - Evaluation
   4.2 Outlier Detection
   4.3 Frequent Pattern Mining
Iterative Mode Search

Idea
Find modes in the point density.

Algorithm

1. Select a window size $\epsilon$, 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.

---

Iterative Mode Search: Example

4. Unsupervised Methods

4.1 Clustering
Mean Shift: Core Algorithm

Algorithm

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

---

Mean Shift: Extensions

Weighted Mean

Use different weights for the points in the window, with weights \( w_x \), resp. calculated by some kernel \( \kappa \):

\[
m^{(i+1)} = \frac{\sum_{x \in W(m^{(i)})} w_x \cdot x}{\sum_{x \in W(m^{(i)})} w_x} \quad \rightarrow \quad m^{(i+1)} = \frac{\sum_{x \in W(m^{(i)})} \kappa(x) \cdot x}{\sum_{x \in W(m^{(i)})} \kappa(x)}
\]

Binning

First quantise data points to grid. Apply iterative mode seeking only once per bin.
Mean Shift: Discussion

Disadvantages

- Relatively high complexity: $N_\epsilon$-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: $\epsilon$
- Support by spatial index: $N_\epsilon$-query (=windowing): $\mathcal{O}(\log n)$. Algorithm: $\mathcal{O}(tn \log n)$
4. Unsupervised Methods

4.1 Clustering

- Introduction
- Partitioning Methods
- Probabilistic Model-Based Methods
- Density-Based Methods
- Mean-Shift
- **Spectral Clustering**
- Hierarchical Methods
- Evaluation

4.2 Outlier Detection

4.3 Frequent Pattern Mining
General Steps for Spectral Clustering

1. Construct Graph out of Data using:
   - kNN
   - \(\epsilon\)-neighborhood
   - fully-connected graph

2. \((\text{weighted})\) adjacency matrix \(W\)
   - degree matrix \(D\)
   - Laplacian matrix \(L\):
     - unnormalized \((D - W)\)
     - normalized

3. Problem to solve:
   \[
   f^T L f = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij} (f_i - f_j)^2 = \min
   \]
   Solution:
   calculate eigenvalues \(\lambda\),
   eigenvectors \(v\) of matrix \(L\)
Choose useful Number of Eigenvalues
- $k$ smallest eigenvalues ($k$: #cluster)
- determine number by:
  - gap in eigenvalues
  - using eigenvectors
  (by matrix rotation and cost function)

Apply k-means on $k$ Eigenvectors

Map Results back to Original Data
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: $\epsilon$-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 $D_{ii} = \sum_{j=1}^{n} W_{ij}$; 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.
Spectral Clustering: Preliminaries

Properties of $L$

1. For every vector $f \in \mathbb{R}^n$, we have: $f L f^T = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} W_{ij} (f_i - f_j)^2$

2. $L$ is symmetric and positive semi-definite

3. The smallest eigenvalue of $L$ is 0, with corresponding eigenvector $\mathbf{1}$

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 $\Lambda$ 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 $f_C L f_C^T = 0$ 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 $f_{C_3} = (0, 0, 0, 0, 0, 1, 1, 1, 1)$.

$$L = \begin{pmatrix} L_1 & 0 \\ 0 & L_2 \\ 0 & 0 & L_3 \end{pmatrix}$$

- Because of the block form of $L$, we get $f_C L f_C^T = 0$ 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. $f^T L f^T$ 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 $\lambda_1, \ldots, \lambda_k$ are small, but $\lambda_{k+1}$ is relatively large. Motivations for that are:
  - $k$ disconnected clusters 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

Smallest eigenvalues of $L$, excluding non-trivial solutions ($\lambda_i, i \geq 1$):
(0.23, 0.70, 3.43) (Notice eigengap between $\lambda_2$ and $\lambda_3$)
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

4. Unsupervised Methods

4.1 Clustering

<table>
<thead>
<tr>
<th>eigenvectors for special case:</th>
<th>eigenvectors for general case:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 0 0</td>
<td>-1.3 3.3 0.4</td>
</tr>
<tr>
<td>1 0 0</td>
<td>-1.1 1 -1</td>
</tr>
<tr>
<td>1 0 0</td>
<td>-1.3 3.3 0.4</td>
</tr>
<tr>
<td>1 0 0</td>
<td>-0.2 -4.3 0.4</td>
</tr>
<tr>
<td>0 1 0</td>
<td>0 -6.6 0</td>
</tr>
<tr>
<td>0 1 0</td>
<td>1.3 3.3 -0.4</td>
</tr>
<tr>
<td>0 1 0</td>
<td>1.3 3.3 -0.4</td>
</tr>
<tr>
<td>0 1 0</td>
<td>1 1 1</td>
</tr>
</tbody>
</table>

Representation of vertex $v_9$: (0,0,1)
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 $O(n^3)$ (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

---

Agenda

1. Introduction

2. Basics

3. Supervised Methods

4. Unsupervised Methods
   4.1 Clustering
     - Introduction
     - Partitioning Methods
     - Probabilistic Model-Based Methods
     - Density-Based Methods
     - Mean-Shift
     - Spectral Clustering
     - Hierarchical Methods
     - Evaluation

4.2 Outlier Detection

4.3 Frequent Pattern Mining
From Partitioning to Hierarchical Clustering

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

- hierarchical cluster structure
- largely differing densities and sizes

Need a hierarchical clustering algorithm in these situations
Hierarchical Clustering: Basic Notions

- Hierarchical decomposition of the data set (with respect to a given similarity measure) into a set of nested clusters
- Result represented by a so called *dendrogram* (greek $\delta\epsilon\nu\delta\rho\omicron = \text{tree}$)
  - 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.
Agglomerative hierarchical clustering requires a distance function for clusters

Given: a distance function \(\text{dist}(p, q)\) 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:**
  \[
  \text{dist}_{sl}(X, Y) = \min_{x \in X, y \in Y} \text{dist}(x, y)
  \]

- **Complete-Link:**
  \[
  \text{dist}_{cl}(X, Y) = \max_{x \in X, y \in Y} \text{dist}(x, y)
  \]

- **Average-Link:**
  \[
  \text{dist}_{al}(X, Y) = \frac{1}{|X| \cdot |Y|} \sum_{x \in X, y \in Y} \text{dist}(x, y)
  \]
Divisive Hierarchical Clustering

**General Approach: Top Down**

- Initially, all objects form one cluster
- Repeat until all clusters are singletons
  - Choose a cluster to split → *how?*
  - Replace the chosen cluster with the sub-clusters and split into two → *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 \( \frac{n(n-1)}{2} \) 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

\[ \text{MinPts} = 3 \]
Core Distance and Reachability Distance

Parameters: "generating" distance $\epsilon$, fixed value $MinPts$

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

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

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

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

\[
\text{reach-dist}(p, o) = \begin{cases} 
\text{dist}(p, o), & \text{dist}(p, o) \geq \text{core-dist}(o) \\
\text{core-dist}(o), & \text{dist}(p, o) < \text{core-dist}(o) \\
\infty, & \text{dist}(p, o) > \epsilon
\end{cases}
\]
The Algorithm OPTICS

OPTICS\textsuperscript{1}: Main Idea

"\textbf{Ordering Points To Identify the Clustering Structure}"

- Visit each point
  - Always make a shortest jump
- Maintain two data structures
  - \textit{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
  - \textit{clusterOrder}: Resulting cluster order is constructed sequentially (order of objects + reachability-distances)

\hspace{1cm}\textsuperscript{1}Ankerst M., Breunig M., Kriegel H.-P., Sander J. "OPTICS: Ordering Points To Identify the Clustering Structure". SIGMOD (1999)
The Algorithm OPTICS

1: \textit{seedList} = ∅
2: \textbf{while} there are unprocessed objects in DB \textbf{do}
3: \hspace{1em} \textbf{if} seedList = ∅ \textbf{then}
4: \hspace{2em} insert arbitrary unprocessed object into \textit{clusterOrder} with reach-dist = ∞
5: \hspace{1em} \textbf{else}
6: \hspace{2em} remove first object from seedList and insert into \textit{clusterOrder} with its current reach-dist
7: \hspace{1em} // Let o be the last object inserted into clusterOrder
8: \hspace{1em} mark o as processed
9: \hspace{1em} \textbf{for} p \in \textit{range}(o, ϵ) \textbf{do}
10: \hspace{2em} // Insert/update p in seedList
11: \hspace{2em} compute \textit{reach-dist}(p, o)
12: \hspace{2em} seedList.update(p, \textit{reach-dist}(p, o))
OPTICS: Example

seed list:
OPTICS: Example

seed list: (B, 40) (I, 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)**
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

4. Unsupervised Methods 4.1 Clustering
OPTICS: The Reachability Plot
OPTICS: The Reachability Plot

- Plot the points together with their reachability-distances. Use the order in which they were 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"

\[ \text{MinPts} = 10, \ \varepsilon = 10 \quad \text{MinPts} = 10, \ \varepsilon = 5 \quad \text{MinPts} = 2, \ \varepsilon = 10 \]
### 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: $O(n^2 \log n^2)$
  - Runtime for OPTICS: without index support $O(n^2)$
- User has to choose the final clustering
Agenda

1. Introduction
2. Basics
3. Supervised Methods

4. Unsupervised Methods
   4.1 Clustering
      - Introduction
      - Partitioning Methods
      - Probabilistic Model-Based Methods
      - Density-Based Methods
      - Mean-Shift
      - Spectral Clustering
      - Hierarchical Methods
      - Evaluation
   4.2 Outlier Detection
   4.3 Frequent Pattern Mining
## Evaluation of Clustering Results

<table>
<thead>
<tr>
<th>Type</th>
<th>Positive</th>
<th>Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expert’s Opinion</td>
<td>may reveal new insight into the data</td>
<td>very expensive, results are not comparable</td>
</tr>
<tr>
<td>External Measures</td>
<td>objective evaluation</td>
<td>needs ”ground truth”</td>
</tr>
<tr>
<td>Internal Measures</td>
<td>no additional information needed</td>
<td>approaches optimizing the evaluation criteria will always be preferred</td>
</tr>
</tbody>
</table>

**Expert’s Opinion**

- **External Measure**
  - ground truth
  - found clustering

**Internal Measure**

- External Measure
- Internal Measure
External Measures

Notation

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

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 $G$ and if they have the same in $C$. 
With \( P = \{(o, p) \in D \times D \mid o \neq p\} \) define:

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

and analogously for \( \mathcal{G} \).
Formalisation as Retrieval Problem for Clustering

Define

- \( TP = |S_C \cap S_G| \)
  (same cluster in both, "true positives")

- \( FP = |S_C \cap S_G| \)
  (same cluster in \( C \), different cluster in \( G \), "false positives")

- \( TN = |\overline{S_C} \cap S_G| \)
  (different cluster in both, "true negatives")

- \( FN = |\overline{S_C} \cap S_G| \)
  (different cluster in \( C \), same cluster in \( G \), "false negatives")

Note the difference to the definitions in classification!
External Measures - Retrieval Problem

- **Recall** \((0 \leq rec \leq 1, \text{larger is better})\)
  
  \[
  rec = \frac{TP}{TP + FN} = \frac{|S_C \cap S_G|}{|S_G|}
  \]

- **Precision** \((0 \leq prec \leq 1, \text{larger is better})\)
  
  \[
  prec = \frac{TP}{TP + FP} = \frac{|S_C \cap S_G|}{|S_C|}
  \]

- **\(F_1\)-Measure** \((0 \leq F_1 \leq 1, \text{larger is better})\)
  
  \[
  F_1 = \frac{2 \cdot rec \cdot prec}{rec + prec} = \frac{2|S_C \cap S_G|}{|S_C| + |S_G|}
  \]
External Measures - Retrieval Problem

- **Rand Index** ($0 \leq RI \leq 1$, larger is better):
  
  $$RI(C \mid G) = \frac{TP + TN}{TP + TN + FP + FN} = \frac{|S_C \cap S_G| + |\overline{S_C} \cap \overline{S_G}|}{|P|}$$

- **Adjusted Rand Index (ARI)**: Compares $RI(C, G)$ against expected ($R, G$) of random cluster assignment $\mathcal{R}$.

- **Jaccard Coefficient** ($0 \leq JC \leq 1$, larger is better):
  
  $$JC = \frac{TP}{TP + FP + FN} = \frac{|S_C \cap S_G|}{|P| - |\overline{S_C} \cap \overline{S_G}|}$$
Confusion Matrix / Contingency Table \( N \in \mathbb{N}^{k \times l} \) with \( N_{ij} = |C_i \cap G_j| \)

\[
\begin{array}{c|ccc}
 & G_1 & \ldots & G_l \\
\hline
C_1 & |C_1 \cap G_1| & \ldots & |C_1 \cap G_l| \\
\vdots & \vdots & \ddots & \vdots \\
C_k & |C_k \cap G_1| & & |C_k \cap G_l| \\
\end{array}
\]

- 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:**

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

- **Mutual Entropy:**

\[
H(C \mid G) = - \sum_{C_i \in C} p(C_i) \sum_{G_j \in G} p(G_j \mid C_i) \log p(G_j \mid C_i)
\]

\[
= - \sum_{C_i \in C} \frac{|C_i|}{|D|} \sum_{G_j \in G} \frac{|C_i \cap G_j|}{|C_i|} \log \frac{|C_i \cap G_j|}{|C_i|}
\]

\[
= - \sum_{i=1}^{k} \frac{N_i}{N} \sum_{j=1}^{l} \frac{N_{ij}}{N_i} \log \frac{N_{ij}}{N_i}
\]
External Measures - Information Theory

- **Mutual Information**: 
  \[ I(C, G) = H(C) - H(C | G) = H(G) - H(G | C) \]

- **Normalized Mutual Information (NMI)** (0 ≤ NMI ≤ 1, larger is better): 
  \[ NMI(C, G) = \frac{I(C, G)}{\sqrt{H(C)H(G)}} \]

- **Adjusted Mutual Information (AMI)**: Compares \( MI(C, G) \) against expected \( MI(R, G) \) of random cluster assignment \( R \).
Internal Measures: Cohesion

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

Cohesion
Average distance between objects of the same cluster.

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

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

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

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

\[ k\text{-means} \quad \text{EM} \quad \text{DBSCAN} \quad \text{Complete Link} \]

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.
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

1. Introduction

2. Basics

3. Supervised Methods

4. Unsupervised Methods
   4.1 Clustering
   4.2 Outlier Detection
      - Introduction
      - Density-based Outliers
      - Angle-based Outliers
      - Tree-based Outliers
   4.3 Frequent Pattern Mining
1. Introduction

2. Basics

3. Supervised Methods

4. Unsupervised Methods
   4.1 Clustering
   4.2 Outlier Detection
      - Introduction
      - Density-based Outliers
      - Angle-based Outliers
      - Tree-based Outliers
   4.3 Frequent Pattern Mining
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.
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

1. Introduction

2. Basics

3. Supervised Methods

4. Unsupervised Methods
   4.1 Clustering
   4.2 Outlier Detection
      - Introduction
      - Density-based Outliers
      - Angle-based Outliers
      - Tree-based Outliers
   4.3 Frequent Pattern Mining
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 $\epsilon$ from the given object
- The choice of $\epsilon$ is critical (too small $\Rightarrow$ normal points considered as outliers; too big $\Rightarrow$ 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.](image)

$D$ has a higher absolute density than $A$ but compared to its neighborhood, $D$'s density is lower.
Density-Based Approaches

Failure Case of Distance-Based

- \(D(\epsilon, \pi)\): parameters \(\epsilon, \pi\) 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

Score ($k = 7$)

Decision ($LOF_k(o) > 2$)

4. Unsupervised Methods

4.2 Outlier Detection
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 $p$)

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

- Local Outlier Factor (LOF) of $p$ (avg. ratio of $lds$ 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 \( lrd \)

\[ LOF_k(p) = \frac{1}{k} \sum_{o \in kNN(p)} \frac{lrd_k(o)}{lrd_k(p)} \]
Density-Based Approaches

Discussion

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

1. Introduction

2. Basics

3. Supervised Methods

4. Unsupervised Methods
   4.1 Clustering
   4.2 Outlier Detection
      - Introduction
      - Density-based Outliers
      - Angle-based Outliers
      - Tree-based Outliers
   4.3 Frequent Pattern Mining
Angle-Based Approach

General Idea

▶ Angles are more stable than distances in high dimensional spaces
▶ *outlier* if most other objects are located in similar directions
▶ *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 $\vec{px}$ and $\vec{py}$ 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$^5$
  
  $$ABOD(p) = \text{VAR}_{x,y \in D} \left( \frac{1}{\|\vec{x}p\|_2 \|\vec{y}p\|_2} \cos (\vec{x}p, \vec{y}p) \right) = \text{VAR}_{x,y \in D} \left( \frac{\langle \vec{x}p, \vec{y}p \rangle}{\|\vec{x}p\|_2^2 \|\vec{y}p\|_2^2} \right)$$

- Small ABOD $\iff$ outlier

---

4. Unsupervised Methods

4.2 Outlier Detection

Angle-Based Approaches

Score (all pairs)

Decision ($ABOD(\sigma) < 0.2$)
Agenda

1. Introduction
2. Basics
3. Supervised Methods
4. Unsupervised Methods
   4.1 Clustering
   4.2 Outlier Detection
       - Introduction
       - Density-based Outliers
       - Angle-based Outliers
       - Tree-based Outliers
   4.3 Frequent Pattern Mining
### 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 $\psi$ points, build an isolation tree
2. Repeat for $t$ times $\Rightarrow$ a forest of $t$ isolation trees
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 $\psi$ 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
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
1. Introduction

2. Basics

3. Supervised Methods

4. Unsupervised Methods
   4.1 Clustering
   4.2 Outlier Detection
   4.3 Frequent Pattern Mining
      Introduction
      Frequent Itemset Mining
      Association Rule Mining
      Sequential Pattern Mining

Introduction
Frequent Itemset Mining
Association Rule Mining
Sequential Pattern Mining
1. Introduction

2. Basics

3. Supervised Methods

4. Unsupervised Methods
   4.1 Clustering
   4.2 Outlier Detection
   4.3 Frequent Pattern Mining
      Introduction
      Frequent Itemset Mining
      Association Rule Mining
      Sequential Pattern Mining
# 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?

\[ D = \{ \begin{align*} &\{ \text{butter, bread, milk, sugar} \}, \\
&\{ \text{butter, flour, milk, sugar} \}, \\
&\{ \text{butter, eggs, milk, salt} \}, \\
&\{ \text{eggs} \}, \\
&\{ \text{butter, flour, milk, salt, sugar} \} \} \end{align*} \]

\[ \Rightarrow \text{80\% of transactions contain the itemset } \{\text{milk, butter}\} \]
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

1. Introduction
2. Basics
3. Supervised Methods
4. Unsupervised Methods
   4.1 Clustering
   4.2 Outlier Detection
   4.3 Frequent Pattern Mining

- Introduction
- Frequent Itemset Mining
- Association Rule Mining
- Sequential Pattern Mining
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 $minSup$.

**Goal**

Given a database $D$ and a threshold $minSup$, find all frequent itemsets $X \in Pot(I)$. 

4. Unsupervised Methods 4.3 Frequent Pattern Mining
### 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 $|I| = m$, $2^m = \text{cardinality of the powerset of } I$).
Mining Frequent Patterns: Apriori Principle

Apriori Principle (anti-monotonicity)

- Any non-empty subset of a frequent itemset is frequent, too!
  \[ A \subseteq I : \text{supp}(A) \geq \text{minSup} \implies \forall \emptyset \neq A' \subset A : \text{supp}(A') \geq \text{minSup} \]

- Any superset of a non-frequent itemset is non-frequent, too!
  \[ A \subseteq I : \text{supp}(A) < \text{minSup} \implies \forall A' \supset A : \text{supp}(A') < \text{minSup} \]
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 $k$
$L_1 = \{\text{frequent items}\}$
for $(k = 1; L_k \neq \emptyset; k++)$ do

\[ \begin{align*}
&\text{join } L_k \text{ with itself to produce } C_{k+1} \\
&\text{discard } (k+1)\text{-itemsets from } C_{k+1} \text{ that } \ldots \\
&\ldots \text{contain non-frequent } k\text{-itemsets as subsets}
\end{align*} \]  \hspace{1cm} \triangleright \text{JOIN STEP} \\
\hspace{1cm} \triangleright \text{PRUNE STEP}

$C_{k+1} = \text{candidates generated from } L_k$

\[ \begin{align*}
&\text{for each transaction } T \in D \text{ do} \\
&\text{Increment the count of all candidates in } C_{k+1} \ldots \\
&\ldots \text{that are contained in } T
\end{align*} \]

$L_{k+1} = \text{candidates in } C_{k+1} \text{ with minSupp}$

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**

```sql
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 \( (L_{k+1} = \{X \in C_{k+1} \mid \text{supp}(X) \geq \text{minSup}\}) \)

- Naïve: Check support of every itemset in \( C_{k+1} \) \( \sim \) 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

\[
\begin{align*}
\text{for all } & c \in C_{k+1} \text{ do} \\
& \text{for all } k\text{-subsets } s \text{ of } c \text{ do} \\
& \quad \text{if } s \notin L_k \text{ then} \\
& \quad \quad \text{Delete } c \text{ from } C_{k+1}
\end{align*}
\]
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 $agh \notin L_3$)
- $C_4 = \{acfg\} \leadsto$ check the support to generate $L_4$
### Apriori Algorithm: Full example

#### Database

<table>
<thead>
<tr>
<th>TID</th>
<th>items</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>acdf</td>
</tr>
<tr>
<td>1</td>
<td>bce</td>
</tr>
<tr>
<td>2</td>
<td>abce</td>
</tr>
<tr>
<td>3</td>
<td>aef</td>
</tr>
</tbody>
</table>

\[
\text{minSup} = 0.5
\]

#### Alphabetic Ordering

<table>
<thead>
<tr>
<th>k</th>
<th>candidate</th>
<th>prune</th>
<th>count threshold</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>a</td>
<td>3</td>
<td>a</td>
</tr>
<tr>
<td></td>
<td>b</td>
<td>2</td>
<td>b</td>
</tr>
<tr>
<td></td>
<td>c</td>
<td>3</td>
<td>c</td>
</tr>
<tr>
<td></td>
<td>d</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>e</td>
<td>3</td>
<td>e</td>
</tr>
<tr>
<td></td>
<td>f</td>
<td>2</td>
<td>f</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>k</th>
<th>candidate</th>
<th>prune</th>
<th>count threshold</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>ab</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ac</td>
<td>2</td>
<td>ac</td>
</tr>
<tr>
<td></td>
<td>ae</td>
<td>2</td>
<td>ae</td>
</tr>
<tr>
<td></td>
<td>af</td>
<td>2</td>
<td>af</td>
</tr>
<tr>
<td></td>
<td>bc</td>
<td>2</td>
<td>bc</td>
</tr>
<tr>
<td></td>
<td>be</td>
<td>2</td>
<td>be</td>
</tr>
<tr>
<td></td>
<td>bf</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ce</td>
<td>2</td>
<td>ce</td>
</tr>
<tr>
<td></td>
<td>cf</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ef</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>k</th>
<th>candidate</th>
<th>prune</th>
<th>count threshold</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>ace</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>acf with cf</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>aef with ef</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>bce</td>
<td>2</td>
<td>bce</td>
</tr>
</tbody>
</table>

#### Frequency-Ascending Ordering

<table>
<thead>
<tr>
<th>k</th>
<th>candidate</th>
<th>prune</th>
<th>count threshold</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>d</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>b</td>
<td>2</td>
<td>b</td>
</tr>
<tr>
<td></td>
<td>f</td>
<td>2</td>
<td>f</td>
</tr>
<tr>
<td></td>
<td>a</td>
<td>3</td>
<td>a</td>
</tr>
<tr>
<td></td>
<td>c</td>
<td>3</td>
<td>c</td>
</tr>
<tr>
<td></td>
<td>e</td>
<td>3</td>
<td>e</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>k</th>
<th>candidate</th>
<th>prune</th>
<th>count threshold</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>bf</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ba</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>bc</td>
<td>2</td>
<td>bc</td>
</tr>
<tr>
<td></td>
<td>be</td>
<td>2</td>
<td>be</td>
</tr>
<tr>
<td></td>
<td>fa</td>
<td>2</td>
<td>fa</td>
</tr>
<tr>
<td></td>
<td>fc</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>fe</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ac</td>
<td>2</td>
<td>ac</td>
</tr>
<tr>
<td></td>
<td>ae</td>
<td>2</td>
<td>ae</td>
</tr>
<tr>
<td></td>
<td>ce</td>
<td>2</td>
<td>ce</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>k</th>
<th>candidate</th>
<th>prune</th>
<th>count threshold</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>bce</td>
<td>2</td>
<td>bce</td>
</tr>
<tr>
<td></td>
<td>ace</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
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; $h(i) = i \mod 3$

### Example

4. Unsupervised Methods

4.3 Frequent Pattern Mining
Hash-Tree: Counting

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

- At root:
  - Compute hash values for all items $t_1, \ldots, t_{n-k+1}$
  - 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:
$\{1, 3, 7, 9, 12\}$
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

<table>
<thead>
<tr>
<th>TID</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>c</td>
</tr>
<tr>
<td>2</td>
<td>cd</td>
</tr>
<tr>
<td>3</td>
<td>cef</td>
</tr>
<tr>
<td>4</td>
<td>cef</td>
</tr>
<tr>
<td>5</td>
<td>bcd</td>
</tr>
<tr>
<td>6</td>
<td>bcd</td>
</tr>
<tr>
<td>7</td>
<td>bcdg</td>
</tr>
<tr>
<td>8</td>
<td>bde</td>
</tr>
<tr>
<td>9</td>
<td>bd</td>
</tr>
<tr>
<td>10</td>
<td>bh</td>
</tr>
<tr>
<td>11</td>
<td>bi</td>
</tr>
<tr>
<td>12</td>
<td>b</td>
</tr>
</tbody>
</table>

1. Scan DB once, find frequent 1-itemsets (single items); Order frequent items in frequency descending order

2. Scan DB again:
   2.1 Keep only freq. items; sort by descending freq.
   2.2 Does path with common prefix exist?
      Yes: Increment counter; append suffix;
      No: Create new branch
FP-Tree Construction

minSup=2/12

<table>
<thead>
<tr>
<th>TID</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>c</td>
</tr>
<tr>
<td>2</td>
<td>cd</td>
</tr>
<tr>
<td>3</td>
<td>cef</td>
</tr>
<tr>
<td>4</td>
<td>cef</td>
</tr>
<tr>
<td>5</td>
<td>bcd</td>
</tr>
<tr>
<td>6</td>
<td>bcd</td>
</tr>
<tr>
<td>7</td>
<td>bcdg</td>
</tr>
<tr>
<td>8</td>
<td>bde</td>
</tr>
<tr>
<td>9</td>
<td>bd</td>
</tr>
<tr>
<td>10</td>
<td>bh</td>
</tr>
<tr>
<td>11</td>
<td>bi</td>
</tr>
<tr>
<td>12</td>
<td>b</td>
</tr>
</tbody>
</table>

1. Scan DB once, find frequent 1-itemsets (single items); Order frequent items in frequency descending order

2. Scan DB again:
   2.1 Keep only freq. items; sort by descending freq.
   2.2 Does path with common prefix exist?
      Yes: Increment counter; append suffix;
      No: Create new branch
FP-Tree Construction

minSup=2/12

<table>
<thead>
<tr>
<th>TID</th>
<th>Items</th>
<th>Freq. Item</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>c</td>
<td>c</td>
</tr>
<tr>
<td>2</td>
<td>cd</td>
<td>cd</td>
</tr>
<tr>
<td>3</td>
<td>cef</td>
<td>cef</td>
</tr>
<tr>
<td>4</td>
<td>cef</td>
<td>cef</td>
</tr>
<tr>
<td>5</td>
<td>bcd</td>
<td>bcd</td>
</tr>
<tr>
<td>6</td>
<td>bcd</td>
<td>bcd</td>
</tr>
<tr>
<td>7</td>
<td>bcdg</td>
<td>bcd</td>
</tr>
<tr>
<td>8</td>
<td>bde</td>
<td>bde</td>
</tr>
<tr>
<td>9</td>
<td>bd</td>
<td>bd</td>
</tr>
<tr>
<td>10</td>
<td>bh</td>
<td>b</td>
</tr>
<tr>
<td>11</td>
<td>bi</td>
<td>b</td>
</tr>
<tr>
<td>12</td>
<td>b</td>
<td>b</td>
</tr>
</tbody>
</table>

1. Scan DB once, find frequent 1-itemsets (single items); Order frequent items in frequency descending order

2. Scan DB again:
   
   2.1 Keep only freq. items; sort by descending freq.
   
   2.2 Does path with common prefix exist?
   
   Yes: Increment counter; append suffix;
   No: Create new branch
1. Scan DB once, find frequent 1-itemsets (single items); Order frequent items in frequency descending order

2. Scan DB again:
   2.1 Keep only freq. items; sort by descending freq.
   2.2 Does path with common prefix exist?
      Yes: Increment counter; append suffix;
      No: Create new branch

---

4. Unsupervised Methods
4.3 Frequent Pattern Mining

---

**minSup=2/12**

<table>
<thead>
<tr>
<th>TID</th>
<th>Items</th>
<th>Freq. Item</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>c</td>
<td>c</td>
</tr>
<tr>
<td>2</td>
<td>cd</td>
<td>cd</td>
</tr>
<tr>
<td>3</td>
<td>cef</td>
<td>cef</td>
</tr>
<tr>
<td>4</td>
<td>cef</td>
<td>cef</td>
</tr>
<tr>
<td>5</td>
<td>bcd</td>
<td>bcd</td>
</tr>
<tr>
<td>6</td>
<td>bcd</td>
<td>bcd</td>
</tr>
<tr>
<td>7</td>
<td>bcdg</td>
<td>bcd</td>
</tr>
<tr>
<td>8</td>
<td>bde</td>
<td>bde</td>
</tr>
<tr>
<td>9</td>
<td>bd</td>
<td>bd</td>
</tr>
<tr>
<td>10</td>
<td>bh</td>
<td>b</td>
</tr>
<tr>
<td>11</td>
<td>bi</td>
<td>b</td>
</tr>
<tr>
<td>12</td>
<td>b</td>
<td>b</td>
</tr>
</tbody>
</table>

**Header Table**

<table>
<thead>
<tr>
<th>Item</th>
<th>Frequency</th>
<th>Head</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>8</td>
<td>☑</td>
</tr>
<tr>
<td>c</td>
<td>7</td>
<td>☑</td>
</tr>
<tr>
<td>d</td>
<td>6</td>
<td>☑</td>
</tr>
<tr>
<td>e</td>
<td>3</td>
<td>☑</td>
</tr>
<tr>
<td>f</td>
<td>2</td>
<td>☑</td>
</tr>
</tbody>
</table>
### 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
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

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

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

<table>
<thead>
<tr>
<th>Conditional Pattern</th>
<th>Item</th>
<th>Cond. Patterns</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>∅</td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>b:3, ∅</td>
<td></td>
</tr>
<tr>
<td>d</td>
<td>bc:3, b:2, c:1</td>
<td></td>
</tr>
<tr>
<td>e</td>
<td>c:2, bd:1</td>
<td></td>
</tr>
<tr>
<td>f</td>
<td>ce:2</td>
<td></td>
</tr>
</tbody>
</table>

Example: e-conditional FP-Tree

<table>
<thead>
<tr>
<th>Item</th>
<th>Frequency</th>
<th>∅</th>
<th>e</th>
</tr>
</thead>
<tbody>
<tr>
<td>c</td>
<td>2</td>
<td></td>
<td>e</td>
</tr>
<tr>
<td>b</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>d</td>
<td>1</td>
<td>c:2</td>
<td></td>
</tr>
</tbody>
</table>
## Major Steps to Mine FP-Tree: Conditional FP-Tree

- Build conditional FP-Trees for each item

<table>
<thead>
<tr>
<th>Item</th>
<th>Cond. Patterns</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>∅</td>
</tr>
<tr>
<td>c</td>
<td>b:3, ∅</td>
</tr>
<tr>
<td>d</td>
<td>bc:3, b:2, c:1</td>
</tr>
<tr>
<td>e</td>
<td>c:2, bd:1</td>
</tr>
<tr>
<td>f</td>
<td>ce:2</td>
</tr>
</tbody>
</table>

\[
\emptyset | b = \emptyset \\
\emptyset | c \\
\emptyset | d \\
\emptyset | e \\
\emptyset | f \\
\emptyset | b:3 \\
\emptyset | b:5 \\
\emptyset | c:1 \\
\emptyset | c:2 \\
\emptyset | c:2 \\
\emptyset | c:3 \\
\emptyset | e:2 \\
\]
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**

<table>
<thead>
<tr>
<th>$\emptyset$</th>
<th>$f$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>c:2</td>
<td>$\sim$</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>e:2</td>
<td></td>
</tr>
</tbody>
</table>

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

\[
\begin{array}{c|c}
\emptyset & d \\
\hline
\emptyset & \emptyset \\
\hline
b:5 & c:1 \\
\hline
\emptyset & db = \emptyset \\
\hline
\emptyset & dc \\
\hline
b & \emptyset \\
\hline
c & b:3, \emptyset \\
\hline
\end{array}
\]

4. Unsupervised Methods

4.3 Frequent Pattern Mining
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 an order of magnitude 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

4. Unsupervised Methods
4.3 Frequent Pattern Mining
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 \(Y \supset X\): \(\text{supp}(Y) < \text{supp}(X)\).

\(\Rightarrow\) 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 \(Y \supset X\): \(\text{supp}(Y) < \text{minSup}\).

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

\(\Rightarrow\) More compact representation