# Data Analysis and Knowledge Discovery

## Unsupervised Learning 2

Jari Björne

University of Turku
Department of Computing

jari.bjorne@utu.fi

# Outline

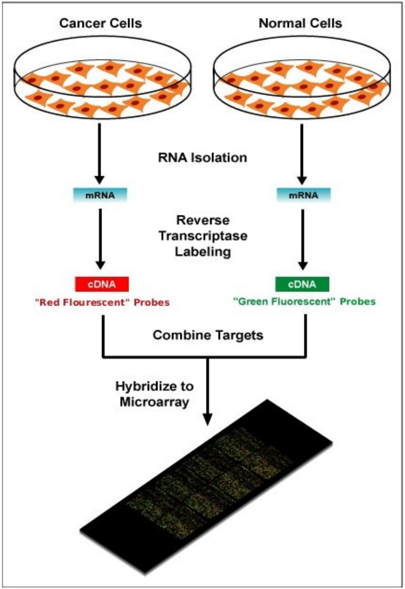
Section 1

# Clustering Application: DNA Microarrays

# DNA Microarrays

- A DNA microarray is a grid of tiny DNA spots (probes) "printed" on a solid (e.g. glass) surface, the chip
- Microarrays are used to measure the expression levels of large numbers of genes (e.g. from a tissue sample)
- The DNA samples are labeled with a fluorescent label $\rightarrow$ The more labeled DNA bound to a probe spot, the brighter the light
- Labeling different samples with different colors allows comparative expression analyses on a single chip. Usually two samples are compared, the test sample (with e.g. a drug) and a normal baseline sample.

# A Typical Microarray Experiment

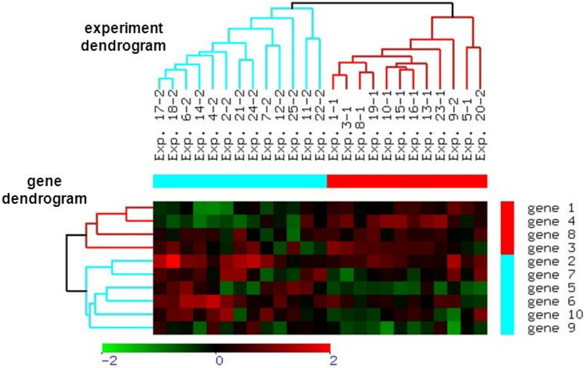



Source: Wikimedia Commons

# DNA Microarrays and Clustering

- When multiple conditions (e.g. drugs) are tested using a set of microarrays, the result is an $M \times N$ matrix
  - Each column is one experiment (one microarray).
  - Each row is the probe spots (genes) of a single microarray.
- Both the experiments and genes can be clustered
  - Experiments: with similarity between all genes in different experiments
  - Genes: with similarity of expression level across all experiments
- Dendrograms can be used to visualize both clusterings

# Microarray data analysis




NIH Center for Information Technology

Jeff Howbert | Introduction to Machine Learning | Winter 2014 | ‹#›

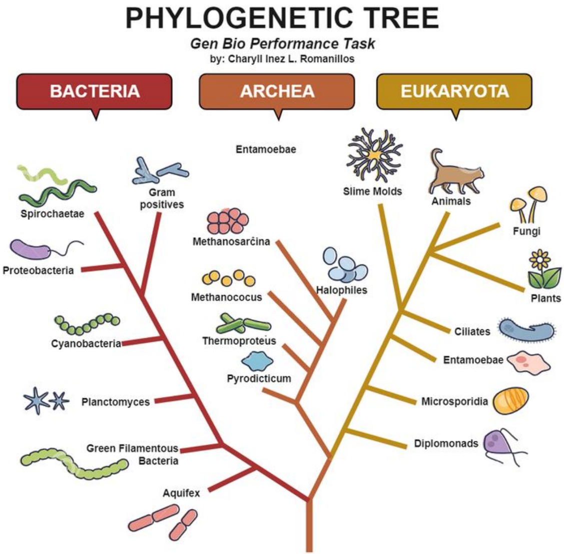
Section 2

# Clustering Application: Phylogenetics

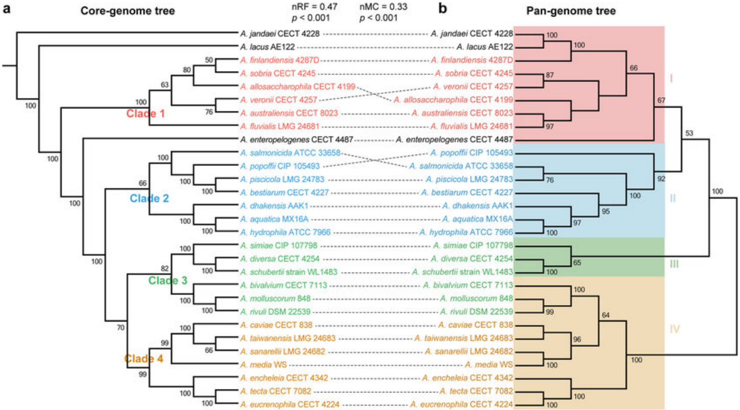
# Phylogenetics

- Phylogenetics is the study of evolutionary history and relationships.
- A phylogenetic tree is a dendrogram showing the speciation from the last common ancestor.
- Traditionally taxonomy was largely based on the morphology of organisms, but genetics is more important today.

# A Phylogenetic Tree



- Phylogenetic trees can be constructed using bioinformatics datasets, such as the sequenced genomes of the species.
- The closer two species are at the genomic level, the closer they are in the evolutionary phylogenetic tree.
- Genomic datasets are complex and there are different ways of constructing and interpreting them $\rightarrow$ constructed trees may differ



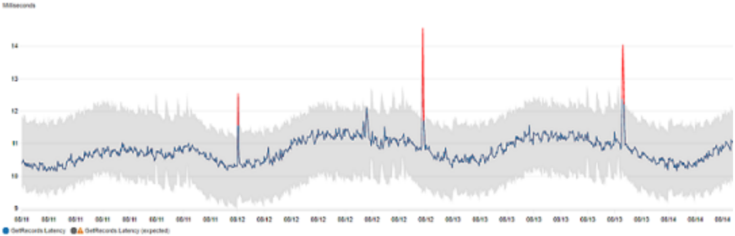

Source: Zhong, Chaofang & Han, Maozhen & Yang, Pengshuo & Chen, Chaoyun & Yu, Hui & Wang, Lusheng & Ning, Kang. (2019). Comprehensive Analysis Reveals the Evolution and Pathogenicity of Aeromonas , Viewed from Both Single Isolated Species and Microbial Communities. mSystems. 4. 10.1128/mSystems.00252-19.

Section 3

# Anomaly Detection

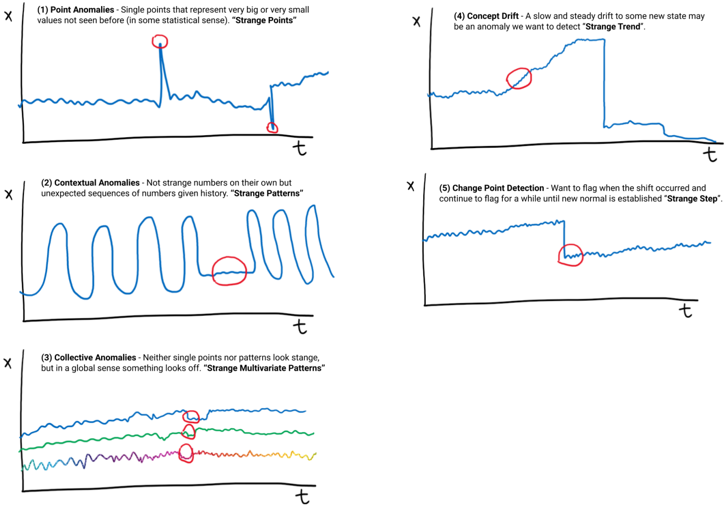
# Anomaly Detection

- Anomaly detection means detection of rare outliers which differ significantly from the rest of the data
- Anomalies can indicate medical problems, a mechanical failure or a cybersecurity intrusion.
- Anomaly detection is based on detecting values outside the regular distribution of the data.




Source: Amazon CloudWatch User Guide

# Types of Time Series Anomalies




Source: Andrew Maguire,
`https://andrewm4894.com/2020/10/19/different-types-of-time-series-anomalies/`

# Anomaly Detection

- **Supervised anomaly detection** requires labeled data for normal and abnormal cases. Very unbalanced data makes it a difficult approach.
- **Semi-supervised anomaly detection** is usually based on a model of normal behaviour which analyses the probability of a data point being within the normal range.
- **Unsupervised anomaly detection** methods are the most widely used approaches.

# Anomaly Detection Methods

- Statistical
- Bayesian Networks
- Hidden Markov Models
- Clustering
- Deviations from association rules and frequent itemsets
- Classifiers (supervised)
- ...

# Deviation analysis and subgroup discovery

- Deviation analysis is a form of subgroup discovery.
- In contrast to association analysis there is usually some target property given.
- The goal is to find subgroups of the populations that are statistically most interesting, that is,
  - they are as large as possible and
  - deviate from the whole population with respect to the property of interest as much as possible.

Example. 10% of customers in the database have bought product $A$, but 30% of customers in the subgroup *female & married* have bought product $A$.

# Deviation analysis and subgroup discovery

The ingredients of deviation analysis are

- a target measure and a verification test serving as a filter for irrelevant or patterns,
- a quality measure to rank subgroups (often the same as the target measure) and
- a search method that enumerates candidates subgroups systematically.

# Deviation analysis and subgroup discovery

Example.

- Assume we are interested in identifying subgroups of churners in our customer database with $N$ customers.
- Assume that a proportion of $p_0$ customers in the whole database are churners.
- Consider a subgroup (for instance *male & under 30*).
- Assume that in this subgroup the proportion of churners $p$.
- If $p$ highly deviates from $p_0$ this seems to be an indicator for a subgroup with different (churn) behaviour.
- But how much must $p$ deviate from $p_0$ in order to consider the effect as significant?

# Deviation analysis and subgroup discovery

- Could be measured with the z-score

$$z = \frac{np_0 - np}{\sqrt{n\,p_0(1-p_0)}} = \frac{\sqrt{n}(p-p_0)}{\sqrt{p_0(1-p_0)}},$$

  the difference between the expected number of churners ($np_0$) in the subgroup and the true number of churners ($np$) in the subgroup, divided by the variance $\sqrt{n\,p_0(1-p_0)}$ (of the underlying binomial distribution, assuming that the customers in the subgroup are picked randomly with replacement).
- Overall depending on the case there are many possible statistical tests, such as Chi2-test, Kolmogorov-Smirnov, weighted relative accuracy, etc...
- Another measure is the weighted relative accuracy

$$WRAcc = (p - p_0) \cdot \frac{n}{N}.$$

Section 4

# Rule Mining

# Association rule mining

- Association rule induction: Originally designed for **market basket analysis**.
- Aims at finding patterns in the shopping behaviour of customers of supermarkets, mail-order companies, on-line shops etc.
- More specifically:
  **Find sets of products that are frequently bought together**.
- Example of an association rule:
  *If a customer buys bread and wine,*
  *then she/he will probably also buy cheese.*

# Association rule mining

- Possible applications of found association rules:
  - Improve arrangement of products in shelves, on a catalog's pages.
  - Support of cross-selling (suggestion of other products), product bundling.
  - Finding business rules and detection of data quality problems.
  - ...

# Association rule mining

- transaction database:
  1. {wine, bread, butter, cheese, jam }
  2. {steak, beer, mustard, sausage }
  3. {diapers, baby food, beer, mustard }
  4. {bread, cheese, wine, olives, dried ham }
  5. {sausage, mustard, corn, coals }
- each product is an *item*
- *item set*: subset of the set of all items
- item sets ordered by frequency
  1. {mustard}:3 {wine}:2, {bread}:2, {cheese}:2, {beer}:2, {sausage}:2 {mustard}:2, {butter}:1, {jam}:1...
  2. {wine, bread}:2, {wine, cheese}:2, {bread, cheese}:2, {mustard, sausage}:2, {wine, butter}:1,...
  3. {wine, bread, cheese}:2, {wine, bread, butter}:1...
  4. ...
- Association rules: {wine, bread}$\rightarrow$ {cheese}, {sausage} $\rightarrow$ {mustard}

# Association rules

A$\rightarrow$ B (If A then B) does not imply B $\rightarrow$ A

- transaction database:
  1. {pea soup, mustard, onions}
  2. {pea soup, mustard, bread, butter}
  3. {sausage, mustard, beer, coals}
  4. {sausage, mustard, ketchup, french fries}
  5. {pea soup, mustard, milk, joghurt}
  6. {sausage, mustard, potatoes, eggs, cream}
- {sausage} $\rightarrow$ {mustard} rule correct in all cases
- {mustard} $\rightarrow$ {sausage} not!

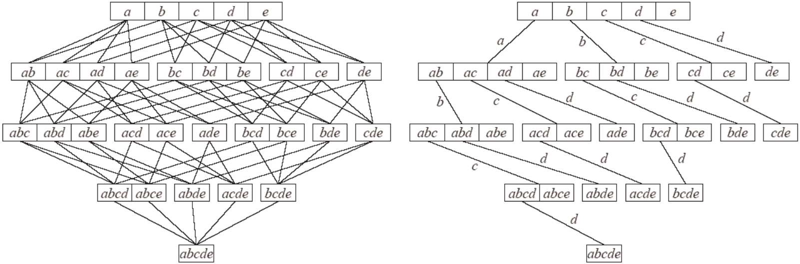
# Association rules

- Assessing the quality of association rules:
  - **Support of an item set**:
    Proportion of transactions (shopping baskets/carts) that contain the item set.
  - **Support of an association rule** $X \to Y$:
    Either: Support of $X \cup Y$ :
    (more common: rule is correct)
    Or: Support of $X$
    (more plausible: rule is applicable)
  - **Confidence of an association rule** $X \to Y$:
    The percentage of all transactions satisfying $X$ that also satisfy $Y$.
    Support of $X \cup Y$ divided by support of $X$ (estimate of $P(Y \mid X)$).

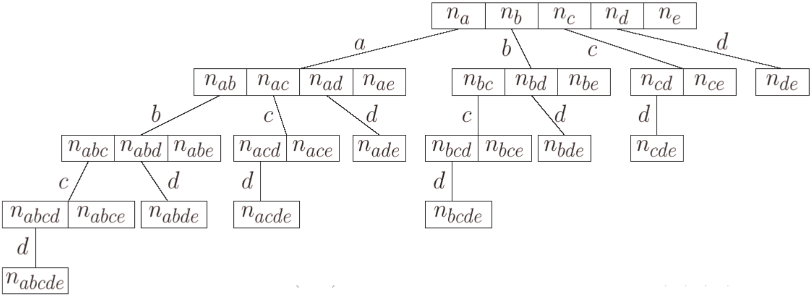
- Two step implementation of the search for association rules:
  - Find the **frequent item sets** (also called *large item sets*), i.e., the item sets that have at least a user-defined **minimum support**.
  - Form rules using the frequent item sets found and select those that have at least a user-defined **minimum confidence**.

# Finding frequent item sets

Subset lattice and a prefix tree for five items:




- It is not possible to determine the support of all possible item sets, because their number grows exponentially with the number of items.
- Efficient methods to search the subset lattice are needed.




A (full) item set tree for the five items $a, b, c, d$, and $e$.

- Based on a global order of the items.
- The item sets counted in a node consist of
  - all items labeling the edges to the node (common prefix) and
  - one item following the last edge label.

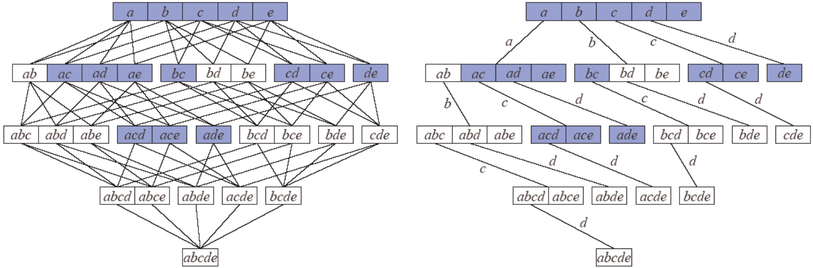
# Item set tree pruning

In applications item set trees tend to get very large, so pruning is needed.

- **Structural Pruning:**
  - Make sure that there is only one counter for each possible item set.
  - Explains the unbalanced structure of the full item set tree.
- **Size Based Pruning:**
  - Prune the tree if a certain depth (a certain size of the item sets) is reached.
  - Idea: Rules with too many items are difficult to interpret.
- **Support Based Pruning:**
  - **No superset of an infrequent item set can be frequent.**
  - No counters for item sets having an infrequent subset are needed.

# Searching the subset lattice

**Boundary** between frequent (blue) and infrequent (white) item sets:




- **Apriori**: Breadth-first search (item sets of same size).
- **Eclat**: Depth-first search (item sets with same prefix).

# Apriori: Breadth first search

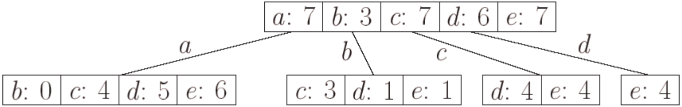
1: $\{a, d, e\}$
2: $\{b, c, d\}$
3: $\{a, c, e\}$
4: $\{a, c, d, e\}$
5: $\{a, e\}$
6: $\{a, c, d\}$
7: $\{b, c\}$
8: $\{a, c, d, e\}$
9: $\{c, b, e\}$
10: $\{a, d, e\}$

| | | | | |
|---|---|---|---|---|
| $a$: 7 | $b$: 3 | $c$: 7 | $d$: 6 | $e$: 7 |

- Example transaction database with 5 items and 10 transactions.
- Minimum support: 30%, i.e., at least 3 transactions must contain the item set.
- All one item sets are frequent $\rightarrow$ full second level is needed.

# Apriori: Breadth first search

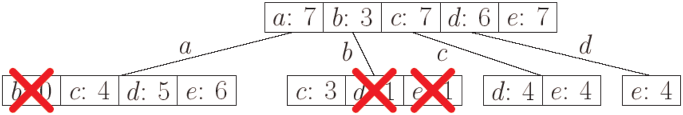
1: $\{a, d, e\}$
2: $\{b, c, d\}$
3: $\{a, c, e\}$
4: $\{a, c, d, e\}$
5: $\{a, e\}$
6: $\{a, c, d\}$
7: $\{b, c\}$
8: $\{a, c, d, e\}$
9: $\{c, b, e\}$
10: $\{a, d, e\}$




- Determining the support of item sets: For each item set traverse the database and count the transactions that contain it (highly inefficient).
- Better: Traverse the tree for each transaction and find the item sets it contains (efficient: can be implemented as a simple double recursive procedure).

# Apriori: Breadth first search

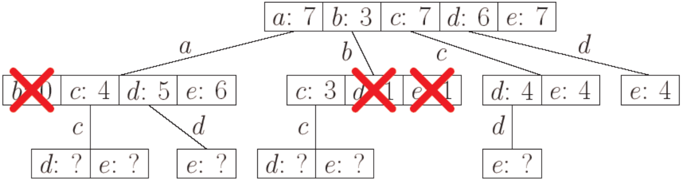
1: $\{a, d, e\}$
2: $\{b, c, d\}$
3: $\{a, c, e\}$
4: $\{a, c, d, e\}$
5: $\{a, e\}$
6: $\{a, c, d\}$
7: $\{b, c\}$
8: $\{a, c, d, e\}$
9: $\{c, b, e\}$
10: $\{a, d, e\}$




- Minimum support: 30%, i.e., at least 3 transactions must contain the item set.
- Infrequent item sets: $\{a, b\}$, $\{b, d\}$, $\{b, e\}$.
- The subtrees starting at these item sets can be pruned.

# Apriori: Breadth first search

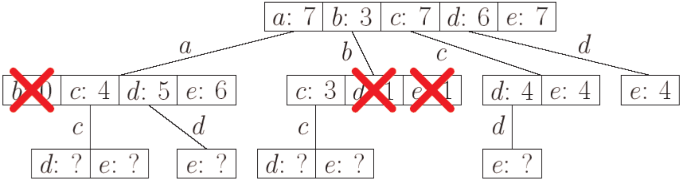
1: $\{a, d, e\}$
2: $\{b, c, d\}$
3: $\{a, c, e\}$
4: $\{a, c, d, e\}$
5: $\{a, e\}$
6: $\{a, c, d\}$
7: $\{b, c\}$
8: $\{a, c, d, e\}$
9: $\{c, b, e\}$
10: $\{a, d, e\}$




- Generate candidate item sets with 3 items (parents must be frequent).

# Apriori: Breadth first search

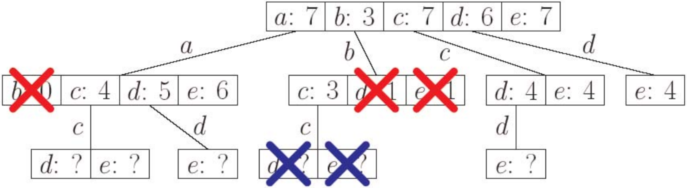
1: $\{a, d, e\}$
2: $\{b, c, d\}$
3: $\{a, c, e\}$
4: $\{a, c, d, e\}$
5: $\{a, e\}$
6: $\{a, c, d\}$
7: $\{b, c\}$
8: $\{a, c, d, e\}$
9: $\{c, b, e\}$
10: $\{a, d, e\}$




- Before counting, check whether the candidates contain an infrequent item set.
  - An item set with $k$ items has $k$ subsets of size $k-1$.
  - The parent is only one of these subsets.

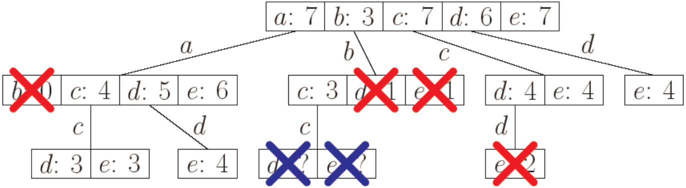
# Apriori: Breadth first search

1: $\{a, d, e\}$
2: $\{b, c, d\}$
3: $\{a, c, e\}$
4: $\{a, c, d, e\}$
5: $\{a, e\}$
6: $\{a, c, d\}$
7: $\{b, c\}$
8: $\{a, c, d, e\}$
9: $\{c, b, e\}$
10: $\{a, d, e\}$



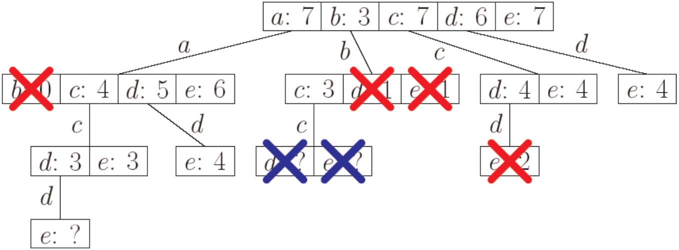

- The item sets $\{b, c, d\}$ and $\{b, c, e\}$ can be pruned, because
  - $\{b, c, d\}$ contains the infrequent item set $\{b, d\}$ and
  - $\{b, c, e\}$ contains the infrequent item set $\{b, e\}$.
- Only the remaining four item sets of size 3 are evaluated.

1: $\{a, d, e\}$
2: $\{b, c, d\}$
3: $\{a, c, e\}$
4: $\{a, c, d, e\}$
5: $\{a, e\}$
6: $\{a, c, d\}$
7: $\{b, c\}$
8: $\{a, c, d, e\}$
9: $\{c, b, e\}$
10: $\{a, d, e\}$



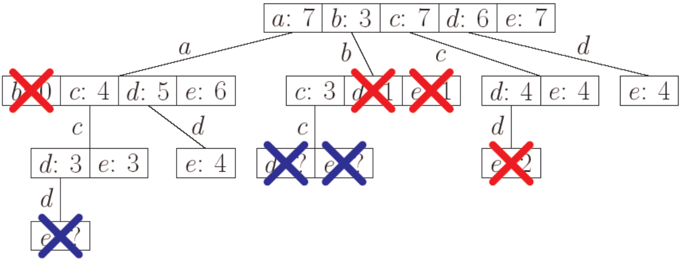

- Minimum support: 30%, i.e., at least 3 transactions must contain the item set.
- Infrequent item set: $\{c, d, e\}$.

1: $\{a, d, e\}$
2: $\{b, c, d\}$
3: $\{a, c, e\}$
4: $\{a, c, d, e\}$
5: $\{a, e\}$
6: $\{a, c, d\}$
7: $\{b, c\}$
8: $\{a, c, d, e\}$
9: $\{c, b, e\}$
10: $\{a, d, e\}$




- Generate candidate item sets with 4 items (parents must be frequent).
- Before counting, check whether the candidates contain an infrequent item set.

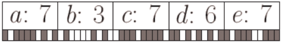
1: $\{a, d, e\}$
2: $\{b, c, d\}$
3: $\{a, c, e\}$
4: $\{a, c, d, e\}$
5: $\{a, e\}$
6: $\{a, c, d\}$
7: $\{b, c\}$
8: $\{a, c, d, e\}$
9: $\{c, b, e\}$
10: $\{a, d, e\}$




- The item set $\{a, c, d, e\}$ can be pruned, because it contains the infrequent item set $\{c, d, e\}$.
- Consequence: No candidate item sets with four items.
- Fourth access to the transaction database is not necessary.

# Eclat: Depth first search

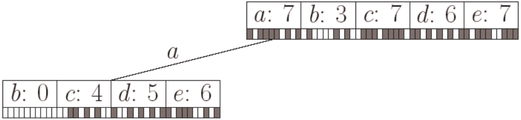
1: $\{a, d, e\}$
2: $\{b, c, d\}$
3: $\{a, c, e\}$
4: $\{a, c, d, e\}$
5: $\{a, e\}$
6: $\{a, c, d\}$
7: $\{b, c\}$
8: $\{a, c, d, e\}$
9: $\{c, b, e\}$
10: $\{a, d, e\}$




- Form a transaction list for each item. Here: bit vector representation.
  - grey: item is contained in transaction
  - white: item is not contained in transaction
- Transaction database is needed only once (for the single item transaction lists).

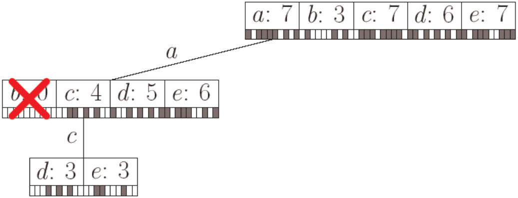
# Eclat: Depth first search

1: $\{a, d, e\}$
2: $\{b, c, d\}$
3: $\{a, c, e\}$
4: $\{a, c, d, e\}$
5: $\{a, e\}$
6: $\{a, c, d\}$
7: $\{b, c\}$
8: $\{a, c, d, e\}$
9: $\{c, b, e\}$
10: $\{a, d, e\}$




- Intersect the transaction list for item *a* with the transaction lists of all other items.
- Count the number of set bits (containing transactions).
- The item set $\{a, b\}$ is infrequent and can be pruned.

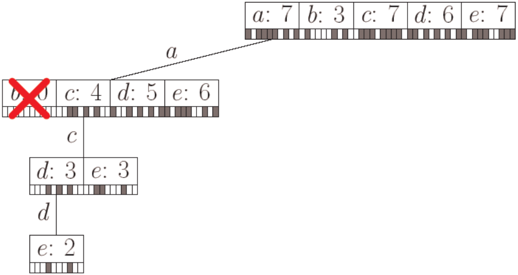
1: $\{a, d, e\}$
2: $\{b, c, d\}$
3: $\{a, c, e\}$
4: $\{a, c, d, e\}$
5: $\{a, e\}$
6: $\{a, c, d\}$
7: $\{b, c\}$
8: $\{a, c, d, e\}$
9: $\{c, b, e\}$
10: $\{a, d, e\}$




- Intersect the transaction list for $\{a, c\}$ with the transaction lists of $\{a, x\}$, $x \in \{d, e\}$.
- Result: Transaction lists for the item sets $\{a, c, d\}$ and $\{a, c, e\}$.

# Eclat: Depth first search

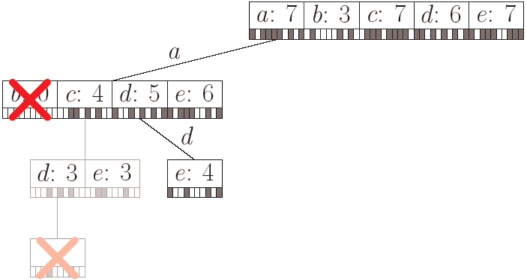
1: $\{a, d, e\}$
2: $\{b, c, d\}$
3: $\{a, c, e\}$
4: $\{a, c, d, e\}$
5: $\{a, e\}$
6: $\{a, c, d\}$
7: $\{b, c\}$
8: $\{a, c, d, e\}$
9: $\{c, b, e\}$
10: $\{a, d, e\}$




- Intersect the transaction list for $\{a, c, d\}$ and $\{a, c, e\}$.
- Result: Transaction list for the item set $\{a, c, d, e\}$.
- With Apriori this item set could be pruned before counting, because it was known that $\{c, d, e\}$ is infrequent.

# Eclat: Depth first search

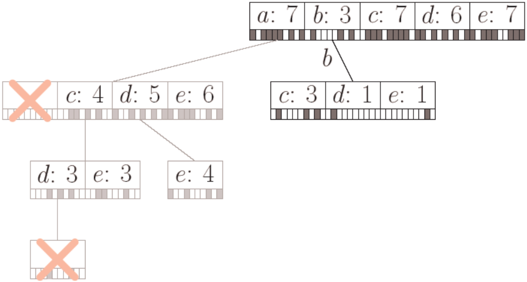
1: $\{a, d, e\}$
2: $\{b, c, d\}$
3: $\{a, c, e\}$
4: $\{a, c, d, e\}$
5: $\{a, e\}$
6: $\{a, c, d\}$
7: $\{b, c\}$
8: $\{a, c, d, e\}$
9: $\{c, b, e\}$
10: $\{a, d, e\}$




- Backtrack to the second level of the search tree and intersect the transaction list for $\{a, d\}$ and $\{a, e\}$.
- Result: Transaction list for $\{a, d, e\}$.

# Eclat: Depth first search

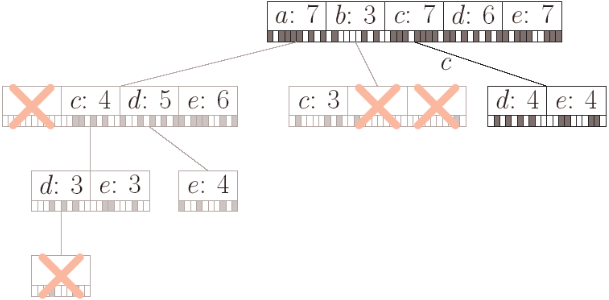
1: $\{a, d, e\}$
2: $\{b, c, d\}$
3: $\{a, c, e\}$
4: $\{a, c, d, e\}$
5: $\{a, e\}$
6: $\{a, c, d\}$
7: $\{b, c\}$
8: $\{a, c, d, e\}$
9: $\{c, b, e\}$
10: $\{a, d, e\}$




- Backtrack to the first level of the search tree and intersect the transaction list for $b$ with the transaction lists for $c$, $d$, and $e$.
- Result: Transaction lists for the item sets $\{b, c\}$, $\{b, d\}$, and $\{b, e\}$.
- Only one item set with sufficient support $\rightarrow$ prune all subtrees.

# Eclat: Depth first search

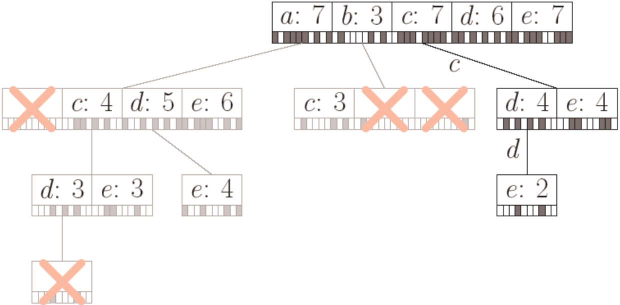
1: $\{a, d, e\}$
2: $\{b, c, d\}$
3: $\{a, c, e\}$
4: $\{a, c, d, e\}$
5: $\{a, e\}$
6: $\{a, c, d\}$
7: $\{b, c\}$
8: $\{a, c, d, e\}$
9: $\{c, b, e\}$
10: $\{a, d, e\}$




- Backtrack to the first level of the search tree and intersect the transaction list for $c$ with the transaction lists for $d$ and $e$.
- Result: Transaction lists for the item sets $\{c, d\}$ and $\{c, e\}$.

# Eclat: Depth first search

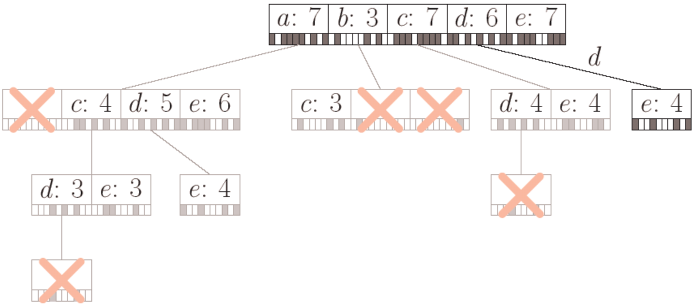
1: $\{a, d, e\}$
2: $\{b, c, d\}$
3: $\{a, c, e\}$
4: $\{a, c, d, e\}$
5: $\{a, e\}$
6: $\{a, c, d\}$
7: $\{b, c\}$
8: $\{a, c, d, e\}$
9: $\{c, b, e\}$
10: $\{a, d, e\}$




- Intersect the transaction list for $\{c, d\}$ and $\{c, e\}$.
- Result: Transaction list for $\{c, d, e\}$.
- Infrequent item set: $\{c, d, e\}$.

# Eclat: Depth first search

1: $\{a, d, e\}$
2: $\{b, c, d\}$
3: $\{a, c, e\}$
4: $\{a, c, d, e\}$
5: $\{a, e\}$
6: $\{a, c, d\}$
7: $\{b, c\}$
8: $\{a, c, d, e\}$
9: $\{c, b, e\}$
10: $\{a, d, e\}$




- Backtrack to the first level of the search tree and intersect the transaction list for $d$ with the transaction list for $e$.
- Result: Transaction list for the item set $\{d, e\}$.
- With this step the search is finished.

# Frequent item sets

| 1 item | 2 items | | 3 items |
|---|---|---|---|
| $\{a\}^+$: 70% | $\{a,c\}^+$: 40% | $\{c,e\}^+$: 40% | $\{a,c,d\}^{+*}$: 30% |
| $\{b\}$: 30% | $\{a,d\}^+$: 50% | $\{d,e\}$: 40% | $\{a,c,e\}^{+*}$: 30% |
| $\{c\}^+$: 70% | $\{a,e\}^+$: 60% | | $\{a,d,e\}^{+*}$: 40% |
| $\{d\}^+$: 60% | $\{b,c\}^{+*}$: 30% | | |
| $\{e\}^+$: 70% | $\{c,d\}^+$: 40% | | |

**Types of frequent item sets**

- **Free Item Set**: Any frequent item set (support is higher than the minimal support).
- **Closed Item Set** (marked with $^+$): A frequent item set is called *closed* if no superset has the same support.
- **Maximal Item Set** (marked with $^*$): A frequent item set is called *maximal* if no superset is frequent.

# Generating association rules

For each frequent item set $S$:

- Consider all pairs of subsets $X, Y \subseteq S$ with $X \cup Y = S$ and $X \cap Y = \emptyset$. Common restriction: $|Y| = 1$, i.e. only one item in consequent (then-part).
- Form the association rule $X \rightarrow Y$ and compute its confidence.

$$\mathrm{conf}(X \rightarrow Y) = \frac{\mathrm{supp}(X \cup Y)}{\mathrm{supp}(X)} = \frac{\mathrm{supp}(S)}{\mathrm{supp}(X)}$$

- Report rules with a confidence higher than the minimum confidence.

**Example:** $S = \{a, c, e\}$, $X = \{c, e\}$, $Y = \{a\}$.

$$\mathrm{conf}(c, e \rightarrow a) = \frac{\mathrm{supp}(\{a, c, e\})}{\mathrm{supp}(\{c, e\})} = \frac{30\%}{40\%} = 75\%$$

**Minimum confidence: 80%**

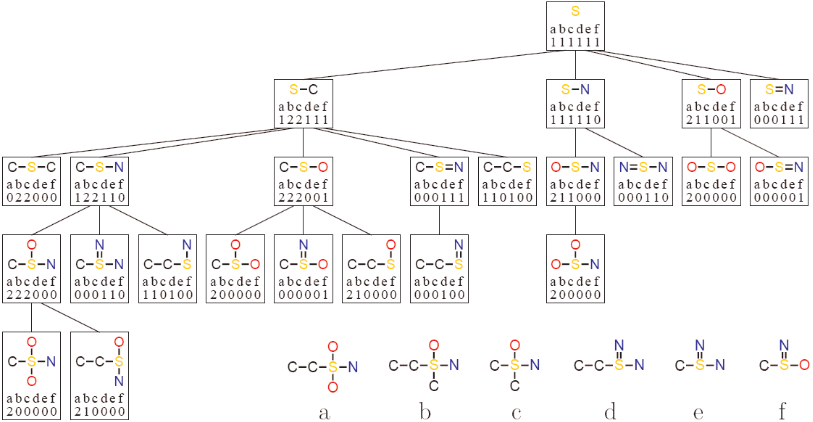
| association rule | support of all items | support of antecedent | confidence |
|---|---|---|---|
| $b \rightarrow c$: | 30% | 30% | 100% |
| $d \rightarrow a$: | 50% | 60% | 83.3% |
| $e \rightarrow a$: | 60% | 70% | 85.7% |
| $a \rightarrow e$: | 60% | 70% | 85.7% |
| $d, e \rightarrow a$: | 40% | 40% | 100% |
| $a, d \rightarrow e$: | 40% | 50% | 80% |

| 1 item | 2 items | | 3 items |
|---|---|---|---|
| $\{a\}^{+}$: 70% | $\{a, c\}^{+}$: 40% | $\{c, e\}^{+}$: 40% | $\{a, c, d\}^{+*}$: 30% |
| $\{b\}$: 30% | $\{a, d\}^{+}$: 50% | $\{d, e\}$: 40% | $\{a, c, e\}^{+*}$: 30% |
| $\{c\}^{+}$: 70% | $\{a, e\}^{+}$: 60% | | $\{a, d, e\}^{+*}$: 40% |
| $\{d\}^{+}$: 60% | $\{b, c\}^{+*}$: 30% | | |
| $\{e\}^{+}$: 70% | $\{c, d\}^{+}$: 40% | | |

# Summary association rules

- **Association Rule Induction is a Two Step Process**
  - Find the frequent item sets (minimum support).
  - Form the relevant association rules (minimum confidence).
- **Finding the Frequent Item Sets**
  - Top-down search in the subset lattice / item set tree.
  - Apriori: Breadth first search;  Eclat: Depth first search.
  - Other algorithms: FP-growth, H-Mine, LCM, Mafia, Relim etc.
  - Search Tree Pruning: *No superset of an infrequent item set can be frequent.*
- **Generating the Association Rules**
  - Form all possible association rules from the frequent item sets.
  - Filter "interesting" association rules.

# Finding frequent molecule substructures



# Applications

- Finding business rules and detection of data quality problems.
  - Association rules with confidence close to 100% could be business rules.
  - Exceptions might be caused by data quality problems.
- Construction of partial classifiers.
  - Search for association rules with a given conclusion part.
  - If . . ., then the customer probably buys the product.

# Resources for association rule analysis

- Association rule analysis not part of standard Python machine learning packages, but some implementations of Apriori exist
- apryori: `https://pypi.org/project/apyori/`, see also `https://stackabuse.com/association-rule-mining-via-apriori-algorithm-in-pytho`
- Another implementation: `https://www.kaggle.com/datatheque/association-rules-mining-market-basket-analysis`
- see also data used in examples
- have not verified the quality of these implementations