

Introduction to Data Mining

- What is data mining?
- Data mining process



Image sources:

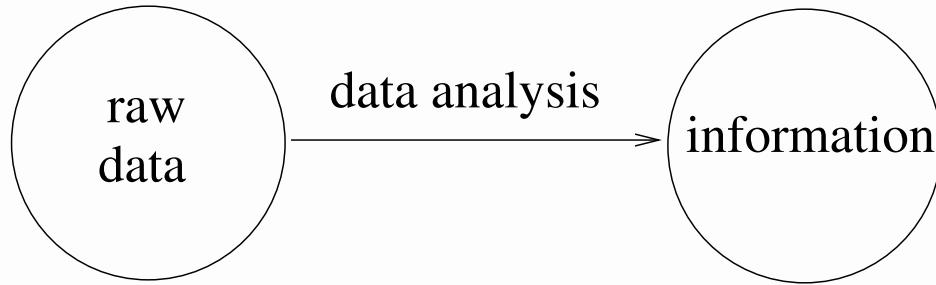
<https://www.pikist.com/>

<https://howtofindgoldnuggets.com/metal-detect-fringe-spots-find-gold-nuggets/>

What is Data mining (DM)?

- no definite and clear answer
- computationally nontrivial data analysis for finding new useful **information** from large collections of **data**
 - interesting patterns like relationships and groupings
- Challenge: data volumes are all the time increasing!
 - ⇒ more efficient algorithms needed
 - ⇒ number of patterns and spurious discoveries increases ⇒ How to find interesting and reliable patterns?

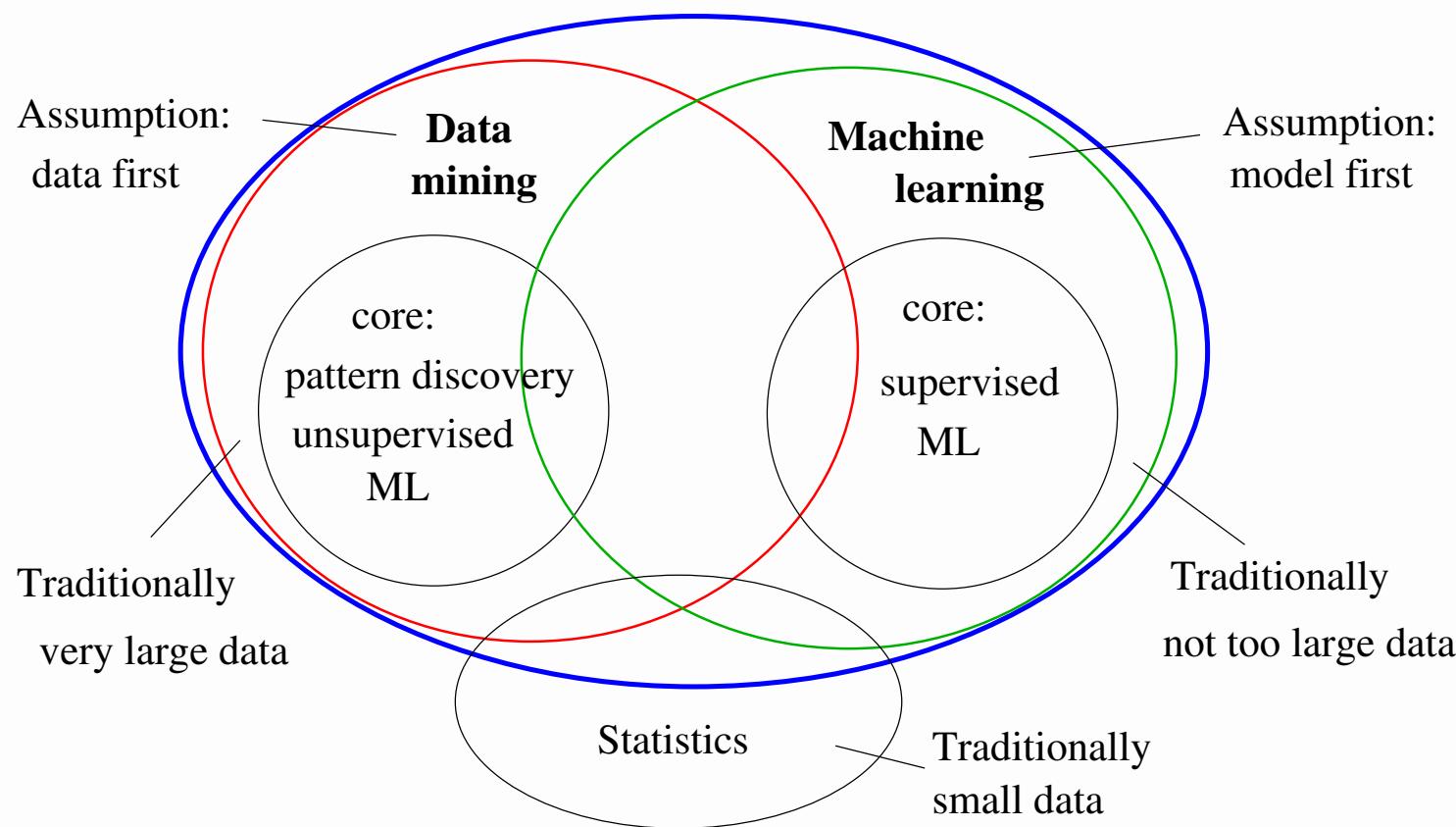
Data vs. Information?



- raw data = unprocessed, uninterpreted facts (e.g, measurements)
- information = knowledge that has meaning, “interpreted data”
- relative terms: the resulting information from one process may be source data for another process

Relationship to closest neighbouring fields

DM ~ knowledge discovery (from databases) (KDD)
Machine learning strongly overlapping/synonymous!



Model vs. pattern?

Model

- global (fits entire data)
- e.g., course success (passing the course) can be predicted from exercise points, time spent on course and participation in exercise groups

Pattern

- local model (describes some part of data)
- e.g., if students obtain high points in assignment 2 they tend to obtain high points also in the exam task 3

DM process

1. Defining problem

understanding data and background



2. Preprocessing

cleaning, feature extraction, selection



3. Data mining (modelling/pattern discovery)



4. Evaluating reliability of results



5. Presenting and interpreting results

1. Defining the problem

- Understanding data: what variables measure/describe?
- What are data types? How much there is data?
- What kinds of patterns would be interesting or useful to find?
- What is already known?
- It is worth studying some background theory!
- Difficulty: How people from different fields find the same language?

Example: defining problem

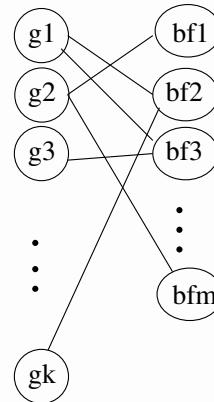
Medical scientist: How TNF- α stimulation affects gene regulation in prostate cancer cells and which biological functions are involved?

Computer scientist: First, explain the data

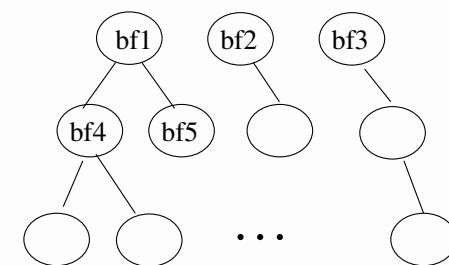
Data matrix: expression of genes g₁,...,g_k and class

id	expression values of						class
	g1	g2	g3	...	gk		
s1	7.1	2.3	4.6			3.1	cancer
s2	6.5	8.0	4.9			5.4	healthy
s3							cancer
				⋮			healthy
							healthy
							cancer

Biological functions of genes



Ontology of biological functions



So, I should find g_i 's that differ significantly in two groups and corresponding bfs ?

2. Preprocessing

- Combining data from different sources (may require transformations)
- Preliminary analysis: means, standard deviations and distributions of variables, correlations, ... (e.g., with statistical tools)
- Data cleaning: handling missing values, detecting and correcting errors
- Feature selection and extraction
- Possibly dimension reduction (combines feature extraction and selection)

3. Data mining

- Typical building blocks dependency analysis, classification, clustering, outlier detection
- Always good to begin from dependency analysis! → choosing features and modelling methods
- Usually descriptive modelling helps in building a predictive model
 - e.g., gene–habit–disease data
 - Descriptive: Find 100 most significant association rules related to variable Diabetes
 - Predictive: Learn (from selected data) the best model that predicts diabetes

4. Evaluating reliability of results

- Are discovered patterns or models sensible?
 - it is possible there are no models or patterns in the data – but **the methods tend to return something even from random data!**
- validating predictive models easy (test set, cross validation)
- evaluating reliability of descriptive models more difficult
- Goal: Some guarantees that the **discovered pattern is not due to chance**
- tools: statistical significance testing, use of validation data

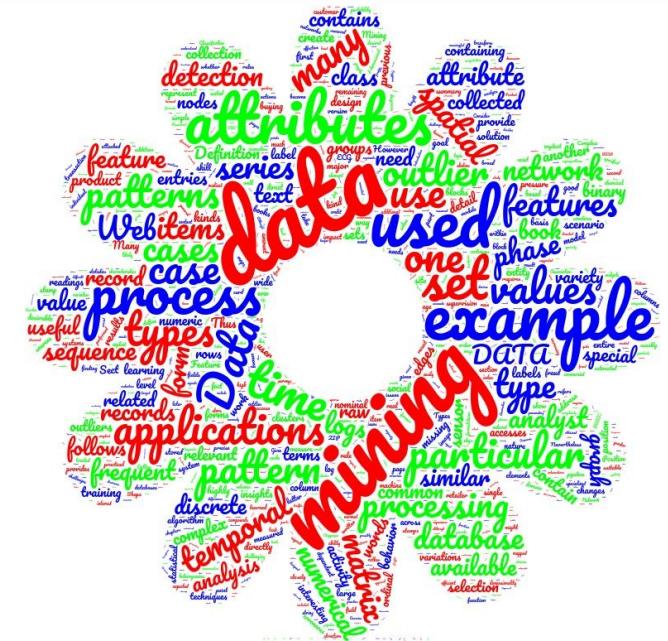
5. Presenting and interpreting results

- present results illustratively so that essential things are emphasized
- domain experts have an important role!
- Did you find something new? Could you formulate a hypothesis based on results? What should be studied further?
- leads often to a new DM round; try new variables and possibly other methods
- finish the iterative DM process when you are satisfied or nothing new seems to be discovered

III Data types

Many ways to characterize data types

- structured or unstructured
- dependency-oriented or nondependency-oriented
- numerical, categorical or mixed
- static \leftrightarrow temporal; spatial; spatio-temporal



Structured vs. Unstructured

- **Structured**
 - has a predefined structure (e.g., rows and features)
 - e.g., multidimensional, graph-formed, time series
- **Unstructured**
 - no pre-defined format, just a string
 - e.g., text, audio, video, signal data
- **Semistructured**
 - contains internal tags that identify separate data elements
 - e.g., XML documents, emails

Dependency-orientation

- Nondependency-oriented: no specified dependencies between objects or attributes
- Dependency-oriented: data objects or values related temporally, spatially or through network links
 - 1. **explicit dependencies**
 - relationships in graph or network data
 - 2. **implicit dependencies**
 - known to typically occur
 - e.g., consecutive temperature readings likely similar

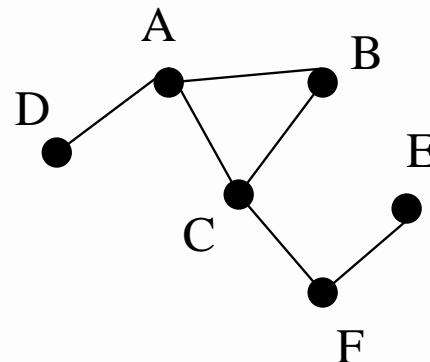
Difference: dependencies in data type vs. patterns in data instances

DATA TYPE



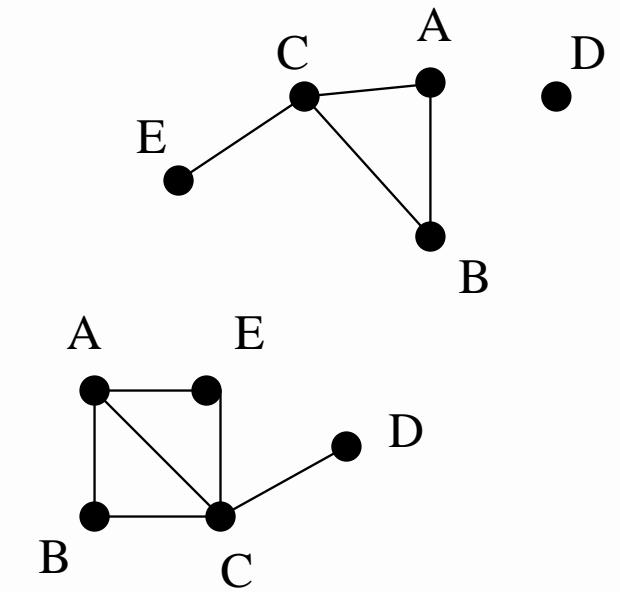
DATA INSTANCES

graph



Dependencies in data structure:

edges present relationships



Discovered dependency: clique of A, B and C occurs frequently

Implicit dependencies harder to separate from patterns!

Basic data type: Multidimensional data

- a set of records, whose fields are features
- notate $\mathcal{D} = \{\overline{X}_1, \dots, \overline{X}_n\}$, where $\overline{X}_i = (x_i^1, \dots, x_i^d)$
 - n rows (records, data points, instances, objects) and d features (fields, attributes, dimensions)
- suitable for a relational database, e.g., cow data:

name	race	weight	parity	milk/d	activity
Rose	Holstein	640	2	35	4800
Daisy	Ayrshire	675	3	37	5100
Strawberry	Finnecattle	615	4	28	7200
Molly	Ayrshire	650	1	32	6300

Numerical, categorical or mixed?

Depending on the type of variables, data may be called numerical (quantitative), categorical or mixed (both).

Variables can be classified by measurement scales:

1 Categorical

1.1 Nominal: values are only labels, no order

- e.g., gender (binary), colour, home city, occupation
- mode (most common value) is defined

1.2 Ordinal: values have an order

- e.g., satisfaction with services: very unsatisfied, unsatisfied, neutral, satisfied, very satisfied
- mode and median (the middle value) defined

Measurement scales (cont'd)

2 Numerical

2.1 Interval scale: difference between values is defined, but **not ratio**

- no true zero point
- temperature 20°C is not twice as warm as 10°C !
- mean and standard deviation defined

2.2 Ratio scale: also **ratio** is defined

- absolute zero = absence of the measured property
- temperature in Kelvins, length, weight, duration
- mean, standard deviation, geometric mean $((\prod x_i)^{1/n})$, coefficient of variation (σ/μ) defined

Circular variables

Idea: Values are ordered categories, where the last category precedes the first

1. Interval circular

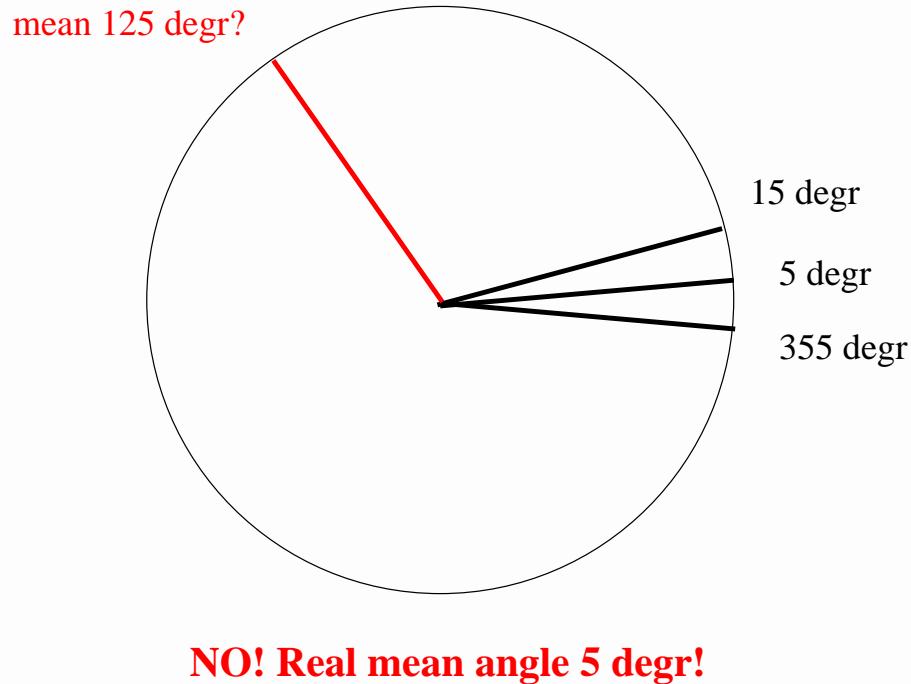
- e.g., compass direction (angles), time of day, day of year
- zero on the measurement scale not meaningful!

2. Ordinal circular

- e.g., days of the week (Mon, Tue,...), compass aspect (N, NE, E,...)

Be careful! E.g, cannot calculate arithmetic mean or normal correlation.

Example: What is the mean angle??



- present angles α_i by $(\cos(\alpha_i), \sin(\alpha_i))$
- $S = \sum_i \sin(\alpha_i)$, $C = \sum_i \cos(\alpha_i)$
- $\theta = \arctan\left(\frac{S}{C}\right)$, if $S \geq 0$,
 $C > 0$
- $\theta = \arctan\left(\frac{S}{C}\right) + \pi$, if $C < 0$
- $\theta = \arctan\left(\frac{S}{C}\right) + 2\pi$, if $S < 0$,
 $C \leq 0$
- $\theta = \pi/2$, if $S > 0$, $C = 0$
- undefined, if $S = 0$, $C = 0$

Present other circular variables first as angles (e.g., $\alpha = \frac{h*2\pi}{24}$)

Warning: Number codes ≠ numerical variables

Categorical values have often arbitrary numerical codes that can't be interpreted as numbers!

Gender: 1 = Female, 2 = Male

Cow's race: 0 = Holstein, 1 = Ayrshire, 2 = Finncattle

- cannot measure distance or ratio or calculate mean or Pearson correlation
- you can get numerical presentation by creating dummy (binary indicator) variables for each value
 - e.g., $I_{Holstein}=1$, if race=Holstein, and 0 otherwise

Warning (cont'd)

The same holds for ordinal variables:

Opinion: 1 = fully disagree, 2 = disagree, 3 = neutral, 4 = agree, 5 = fully agree

- if fully ordinal and distances between categories equal, variable may be treated as numerical (but not always optimal)
- more typical when many categories (≥ 7)
- Be careful!

Opinion: 0 = Don't know, 1 = fully disagree, 2 = disagree, 3 = neutral, 4 = agree, 5 = fully agree

Other data types

- time series
- discrete sequences
- spatial data
- network and graph data
- text

Time series

- continuous measurements over time
- e.g., from environmental sensors, health monitoring devices, ECG
- at time stamps t_1, \dots, t_n measurements (Y_1, \dots, Y_n)
- may also be multivariate time series $(\bar{Y}_1, \dots, \bar{Y}_n)$, where $\bar{Y}_i = (y_i^1, \dots, y_i^d)$
- e.g., heart rate, oxygen saturation, diastolic and systolic blood pressure at every minute
- often temporal correlations (like dependencies between consecutive values or periodic patterns)

Discrete sequences

- like time series, but sequences of categorical variables
- special case: strings (no time stamps, but positions)
- e.g., event logs, strings of nucleotides (DNA, genes)

Event ID	Class	Type	Severity	Date/Time	Description
958	Audit	Log	minor	Fri Apr 23 15:03:30 2010	root : Open Session : object = /session/type : value = www : success
957	Fault	Fault	critical	Fri Apr 23 13:02:41 2010	Fault detected at time = Fri Apr 23 13:02:41 2010. The suspect component: /SYS/BL3/NET1 has fault io pciex/fabric fatal with probability=50. Refer to http://www.sun.com/msg/SPX86-8001-95 for details.
956	Fault	Fault	critical	Fri Apr 23 13:02:41 2010	Fault detected at time = Fri Apr 23 13:02:41 2010. The suspect component: /SYS/BL3/NETD has fault io pciex/fabric fatal with probability=50. Refer to http://www.sun.com/msg/SPX86-8001-95 for details.
955	IPMI	Log	critical	Fri Apr 23 13:02:38 2010	ID = 1d1 : 04/23/2010 : 13:02:38 : Critical Interrupt : BIOS : PCI SERR: IOH 3 ESI
954	IPMI	Log	critical	Fri Apr 23 13:02:38 2010	ID = 1d0 : 04/23/2010 : 13:02:38 : Critical Interrupt : BIOS : PCI SERR: IOH 2 ESI
953	IPMI	Log	critical	Fri Apr 23 13:02:38 2010	ID = 1cf : 04/23/2010 : 13:02:38 : Critical Interrupt : BIOS : PCI SERR: IOH 1 ESI

Figure from <https://docs.oracle.com/cd/E19140-01/html/821-0796/gjfwa.html>

Difficulty: how to combine temporal data when the measuring frequency varies?

Example from a cow-house:

- body temperature and rumen acidity are measured every minute
- activity device records average activity every 15 min
- milk production (amount, protein and fat contents etc.) is measured daily
- feeding automaton event log contains time stamp, automaton id, cow id, feed type, amount and duration for every visit
- drinking automaton event log contains time stamp, cow id, amount of water and duration

Spatial and spatiotemporal data

- spatial: measurements of non-spatial attributes in spatial locations (typically 2D)
 - e.g. sea surface temperature
- spatiotemporal data
 - e.g., temperature over time or ship trajectories

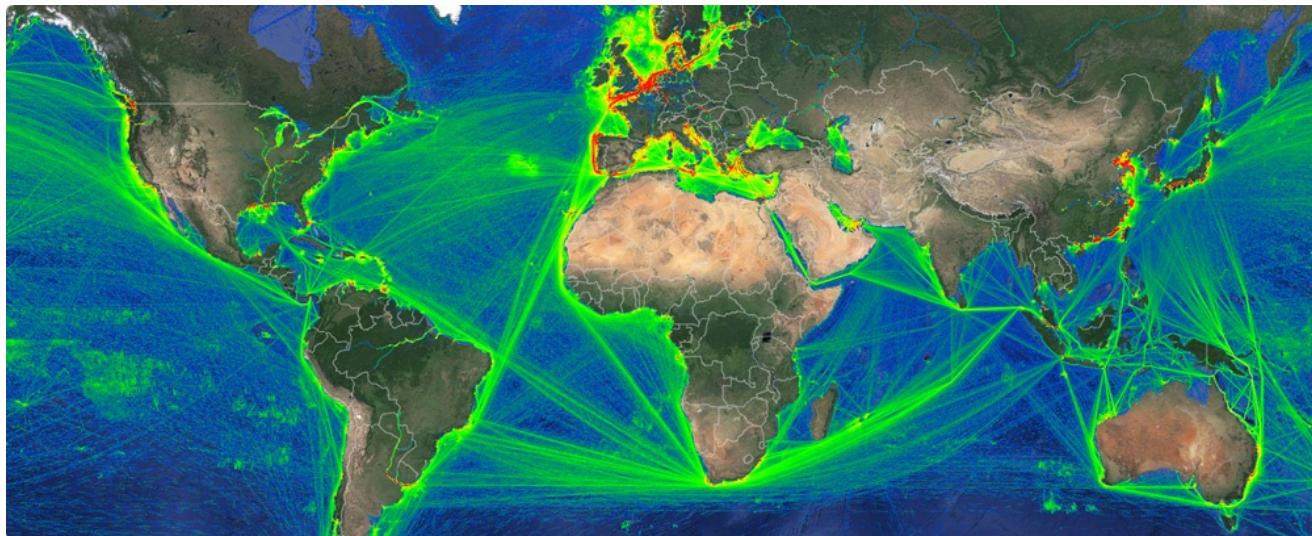


Figure from <http://www.elane.com/EN/Detail106.html>

Spatiotemporal data: contextual and behavioural attributes

Contextual attributes define the context

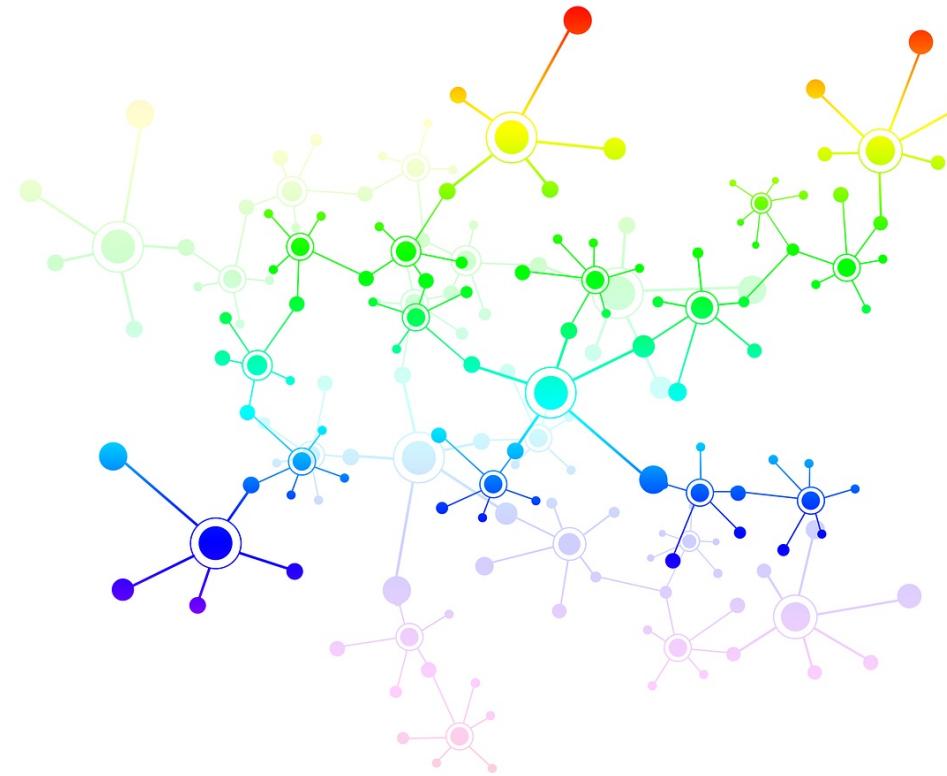
Behavioural attributes are measured in this context

Two main types of spatiotemporal data:

1. Both spatial and temporal attributes define the context where some behavioural attribute (like temperature) is measured
2. Temporal attribute is contextual and spatial attributes are behavioural (e.g., trajectory analysis)

Network and graph data

- nodes correspond objects and edges relationships
+ attributes may be associated with nodes or edges
- directed (web structure) or undirected (social network)



Example: wikipedia hyperlink structure

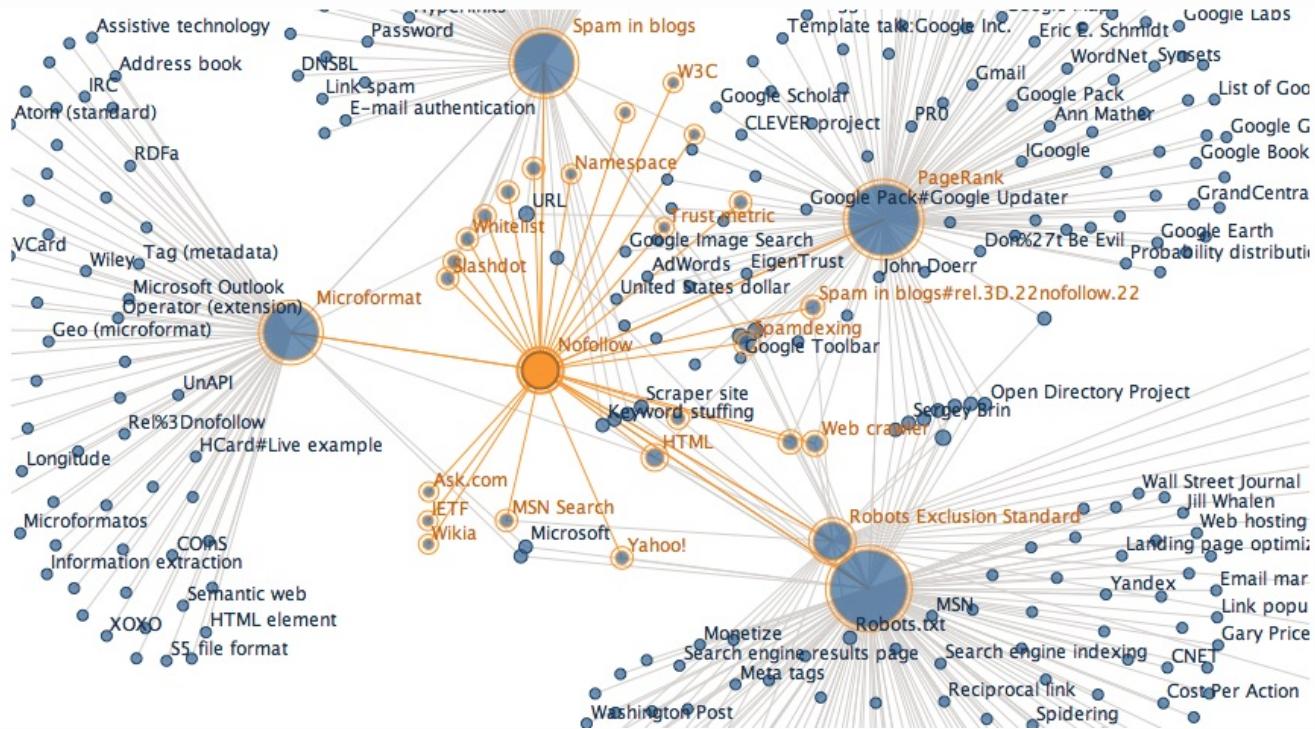
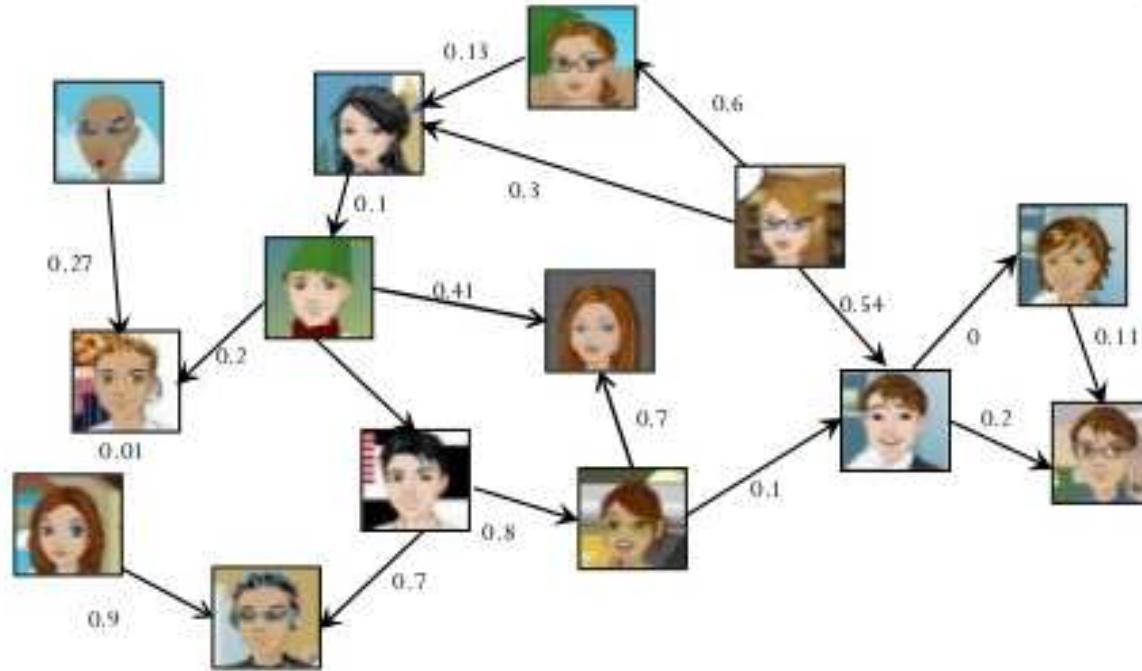


Figure from

<https://wiki.digitalmethods.net/Dmi/>

Wikipedia Analysis

Example: social network structure



- **Nodes:** Individuals in the network
- **Edges:** Links/relationships between individuals
- **Edge weight on (i,j) :** Influence weight $w_{i,j}$

Source: Lu and Lakshmanan ICDM 2012

<https://www.slideshare.net/WeiLu12/>

profit-maximization-over-social-networks

Text data

- raw text is a string, i.e., dependency-oriented
- often represented as a **bag-of-words** or **document-term matrix** (nondependency-oriented)
- which can be presented in vector space (as multidimensional data)
 - how often terms occur in document? \Rightarrow numerical features for term frequencies
 - \Rightarrow often transformed to tf-idf values (contains weighting + log scaling)

More on the text mining lecture!

Example: tf-idf presentation of sentences

d0: Simple example with cats and mouse

d1: Another simple example with dogs and cats

d2: Another simple example with mouse and cheese

	and	another	cats	cheese	dogs	example	mouse	simple	with
0	1	0	1	0	0	1	1	1	1
1	1	1	1	0	1	1	0	1	1
2	1	1	0	1	0	1	1	1	1

	and	another	cats	cheese	dogs	example	mouse	simple	with
0	0.0	0.000000	0.067578	0.000000	0.000000	0.0	0.067578	0.0	0.0
1	0.0	0.057924	0.057924	0.000000	0.156945	0.0	0.000000	0.0	0.0
2	0.0	0.057924	0.000000	0.156945	0.000000	0.0	0.057924	0.0	0.0

Example from <https://medium.com/@MSalnikov/text-clustering-with-k-means-and-tf-idf-f099bcf95183>

Data preprocessing: main tasks

1. Data cleaning: handling errors and missing values
2. Feature extraction: creating new features by combining and transforming existing ones
 - a **crucial step!** ⇒ what patterns you can find
 - application specific ⇒ understanding the domain
3. Data reduction
 - sampling
 - feature selection
 - dimension reduction by transformations

1. Data cleaning

Goal: detect & eliminate errors, missing values, duplicates, noise, sometimes outliers

- but outliers may also reveal some interesting event!

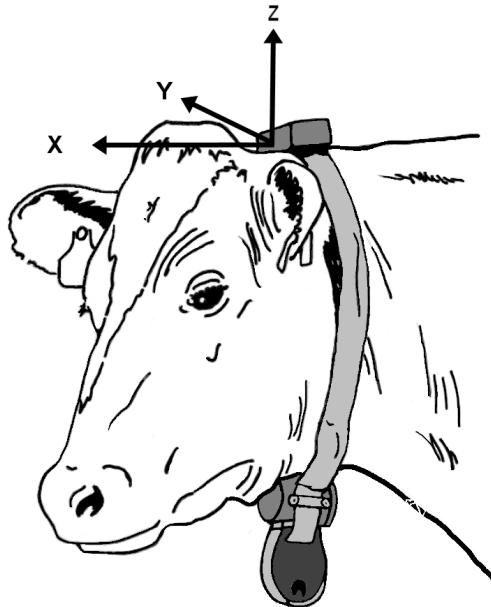
Sources:

- automatic measuring devices may stop reading or transmit duplicates (e.g., HW failures or battery exhaustion)
- users may not want to specify (correct) information for privacy reasons
- manually entered data contains very often errors!
- automatically produced text (from scanned documents or speech) prone to errors

Real world example

Task: predict cows' activities (walking, standing, lying, ...)

Data: sequences of accelerometer measurements for time intervals when an animal performs an activity (class).

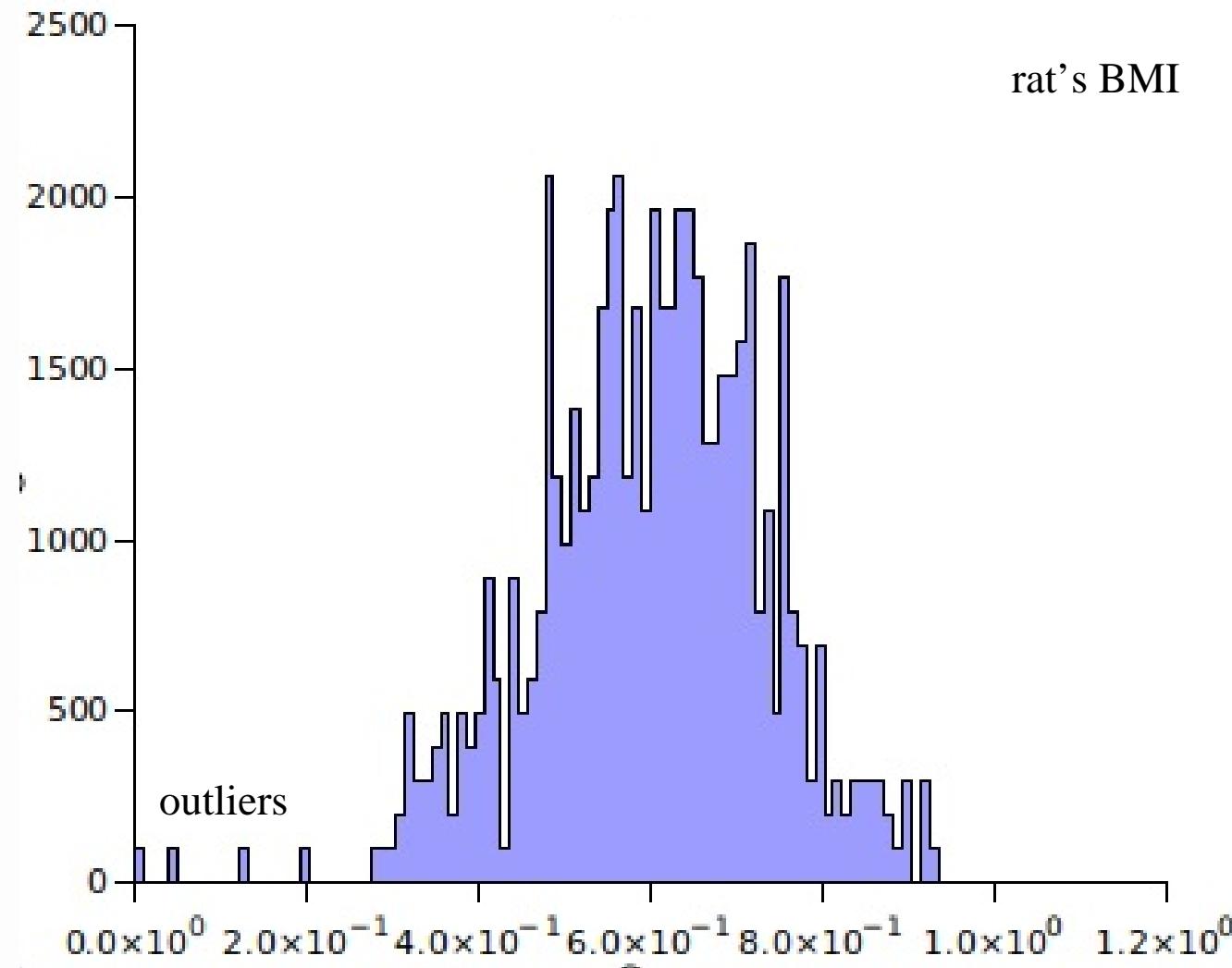


- faulty devices
- lack of calibration
- transmission breaks
- noise
- individual fluctuations
- errors in human annotation

Errors and inconsistencies: strategies

- check inconsistencies between different data sources
 - e.g., name spelling
- use domain knowledge
 - known ranges of values (age cannot be 800 yrs)
 - known relationships (if country='USA', city≠'Sanghai')
- check outliers and extreme values (error candidates)
 - not errors, if they have a reasonable explanation
- data smoothing reduces noise and random fluctuations
 - e.g., scaling, discretization, dimension reduction
- use robust methods in the modelling phase

Example: outliers may reveal errors



Missing values: strategies

If possible, **replace with correct values**. Otherwise,

- if a feature has many missing values, **prune the feature**
- if a record has many missing value, **prune the record**
- **impute missing values**
 - mean or median of the feature (among all or similar records/nearest neighbours)
 - predict the missing value using other features (e.g., random forests imputation)
 - **Warning!** Imputation may have a strong effect the results!
- use a **modelling technique that allows missing values** (just replace with special values like “NA”)

2. Feature extraction methods

- scaling and normalization: numerical → numerical
- discretization: numerical → categorical
- binarization: categorical → binary (0/1)
- creating similarity graphs: any type → graph
- transformations for dimension reduction: create new less redundant features and keep the best ones
 - both feature extraction + data reduction

Scaling and normalization

Problem: Features with large magnitudes often dominate
⇒ transform to the same scale or standardize distributions

- **min-max scaling:**

$$y = \frac{x - \min(x)}{\max(x) - \min(x)} \quad (\text{new range } [0, 1])$$

- **mean normalization:**

$$y = \frac{x - \text{mean}(x)}{\max(x) - \min(x)} \quad (\text{new range } [-1, 1], \text{mean}(y) = 0)$$

- **Beware!** outliers can affect a lot!

Standardization or z -score normalization

$$z = \frac{x - \text{mean}(x)}{\text{stdev}(x)}$$

$$\text{mean}(z) = 0$$

$$\text{stdev}(z) = 1$$

If the distribution is normal:

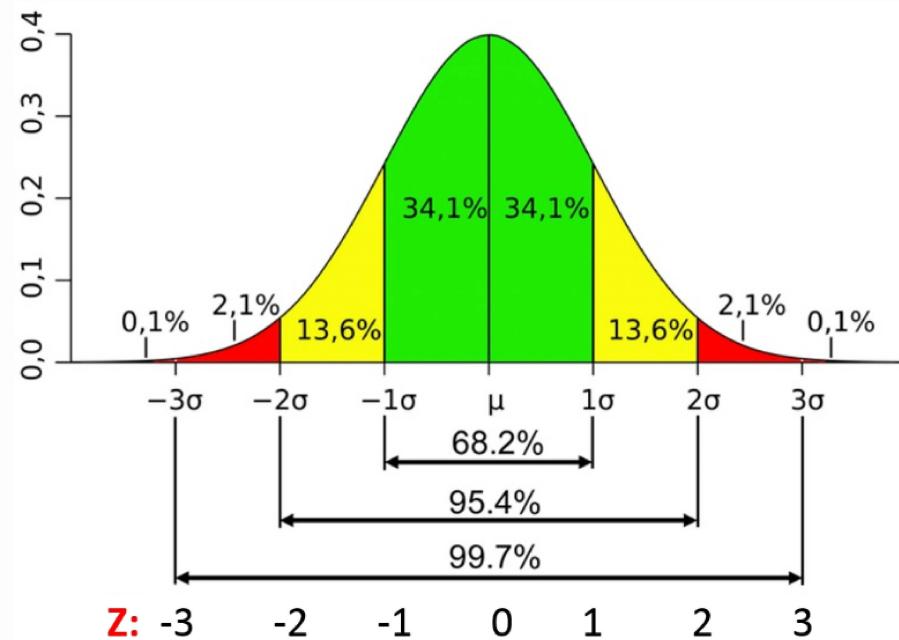


image source:

<https://sphweb.bumc.bu.edu/otlt/MPH-Modules/PH717-QuantCore/PH717-Module6-RandomError/PH717-Module6-RandomError5.html>

Discretization: numerical → categorical

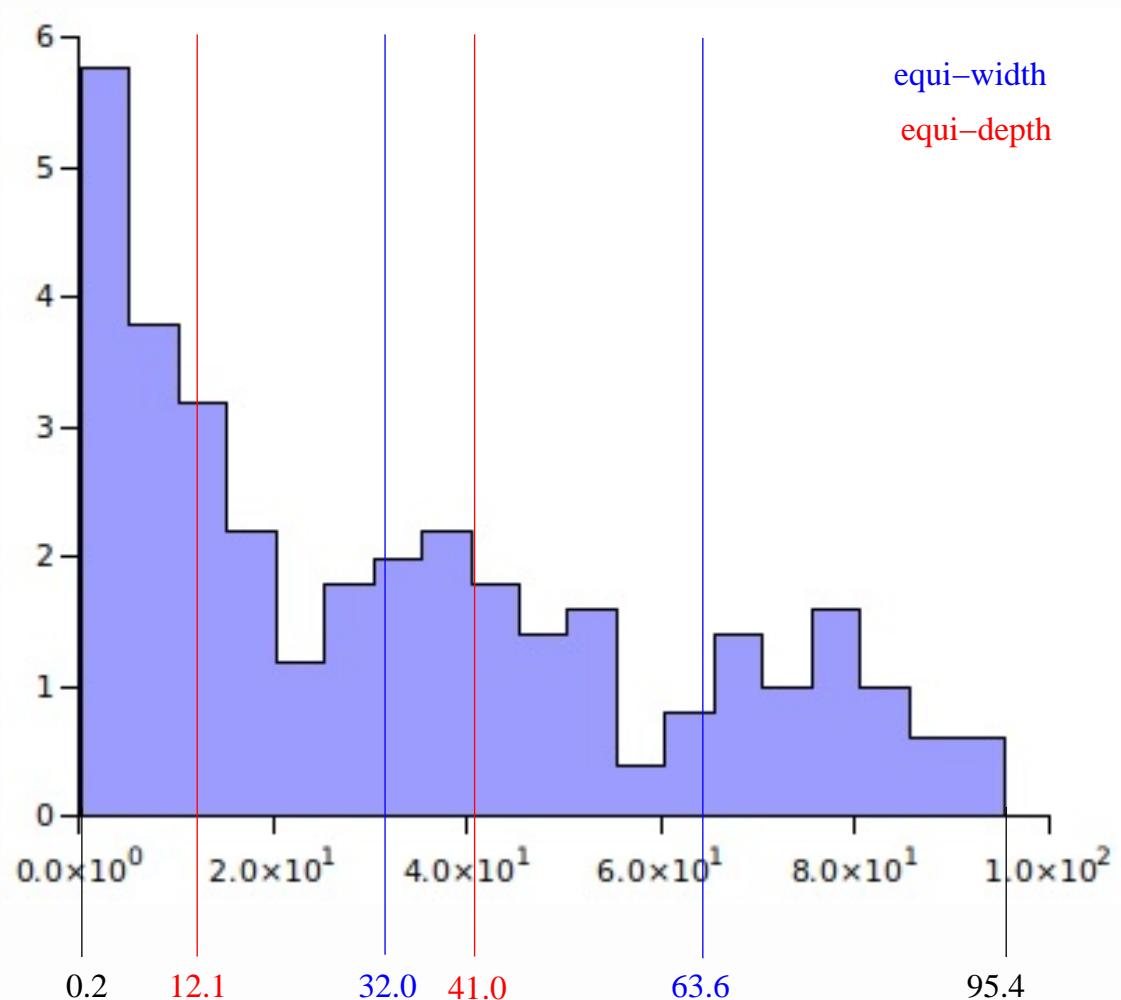
- divide the numerical range into intervals (bins) + give labels to bins
- temperature could be discretized as $T < 0^\circ\text{C}$ cold, $0 \leq T < 15^\circ\text{C}$ cool, $T \geq 15^\circ\text{C}$ warm
- **binarization**: a special case when the new variable is binary (true/false or 1/0)
- e.g., $\text{frost}=1$, if $T < 0$ and $\text{frost}=0$ otherwise
- Note! Also categorical variables can be binarized
 - $\text{eye-colour}=\{\text{blue, brown, green, grey}\} \Rightarrow \text{blue-eyed}=1$, if $\text{eye-colour}=\text{blue}$, and 0 otherwise

Some discretization methods

- Equi-width discretization
 - equally wide bins
 - good if uniform distribution
- Equi-depth (equal frequency)
 - each bin has an equal number of records
- Many supervised methods if class labels available
- Visual/manual: often best results, but can be worksome

Example: internet users/100 people in countries

Equi-width or equi-depth wouldn't present natural groups



Discretization: benefits and limitations

- + good way to handle mixed data
 - + removes noise and individual variation
- ⇒ it is often worth of analyzing a discretized version of purely numerical data
- + less noise, clearer patterns
 - + more efficient algorithms
 - + discrete patterns may help to choose the right modelling method also for numerical data
- loses some information
 - optimal discretization difficult! (optimal discretization of one variable may depend on other variables)

Useful type transport: any type → similarity graph

- idea: present **pairwise similarities** among closest neighbours by a neighbourhood/similarity graph
- suitable for **any data type** if the distance/similarity function can be defined
- for any application based on the notion of similarity/distances
 - e.g., clustering, recommendations based on similarity
- enables use of numerous network algorithms
- Beware: can be time consuming for large data! (brute force $O(n^2)$, n =number of objects)

Constructing nearest neighbour graph (idea)

Given objects O_1, \dots, O_n , a distance measure d and a user-defined parameter ϵ or K .

1. create a node for each O_i
2. create an edge between a pair near/similar objects:
 - i) if $d(O_i, O_j) \leq \epsilon \Rightarrow$ undirected edge $O_i - O_j$ **or**
 - ii) if O_j is among K nearest neighbours of $O_i \Rightarrow$ directed edge $O_i \rightarrow O_j$ (direction can be ignored)
3. give weights to edges reflecting similarity, e.g.,

$$w_{ij} = e^{-d(O_i, O_j)^2/t^2} \quad (\text{heat kernel, } t \text{ user-defined})$$

3. Data reduction: approaches

1. sampling (select a subset of records)
2. feature selection (select a subset of features)
 - application specific!
 - **filtering** methods: prune features before modelling
 - **wrapper** methods: use modelling (e.g., clustering) to evaluate goodness of feature sets
 - **hybrid** methods: candidates by filtering + evaluation by modelling
3. dimension reduction
 - by axis rotation (PCA, SVD)
 - with type transformation

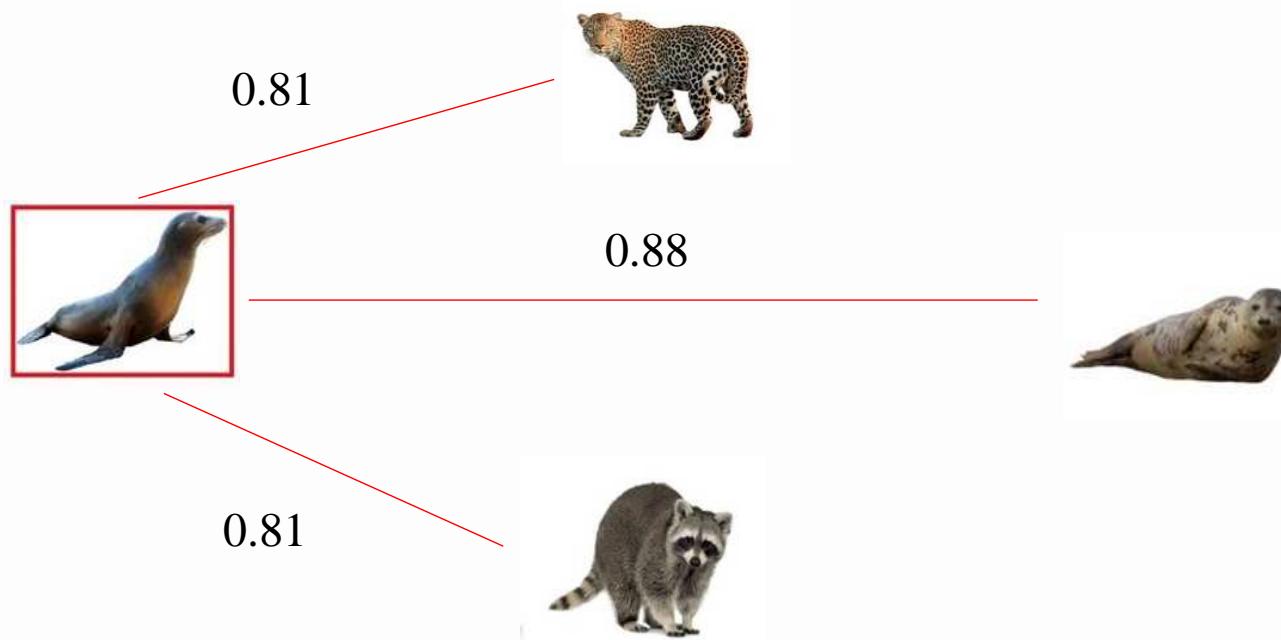
Main messages

- careful with data types
- careful with preprocessing (data often dirty!)
- feature extraction has a strong effect

Reading for lecture 1:

Book Ch 1 and Ch 2 except 2.4.3–2.4.4

Lecture 3: Similarity and distance measures



Book chapter 3

image source Lin 2018. <https://www.linkedin.com/pulse/cosine-similarity-classification-michael-lin>

Contents

- Concepts of distance, similarity, metric
- Measures for numerical data (L_p -norms, similarity measures, accounting for distribution)
- Measures for categorical and mixed data
- Measures for sets, strings, text

What is distance?

Let S be a space of data objects. A distance function has the type

$$d : S \times S \rightarrow \mathbb{R}^+ \cup \{0\}$$

Intuitively: Let $x, y, z \in S$ be objects.

- if $d(x, y)$ small, x and y are close or similar
- If $d(x, y) < d(x, z)$, x is closer/more similar to y than z

Similarity vs. distance

Similarity function $s : \mathcal{S} \times \mathcal{S} \rightarrow \mathbb{R}$

- $s(\mathbf{x}, \mathbf{y})$ large when \mathbf{x} and \mathbf{y} similar (and $d(\mathbf{x}, \mathbf{y})$ small)
- often $s : \mathcal{S} \times \mathcal{S} \rightarrow [0, 1]$
- \Rightarrow possible to induce distance $d_s = 1 - s$
- if $d : \mathcal{S} \times \mathcal{S} \rightarrow [0, 1]$, possible to induce similarity
 $s_d = 1 - d$
- if not, then e.g.,
 $s_d = 1 - \frac{d}{D}$ (D =maximal possible distance) or
 $s_d = \frac{1}{1+d}$

Metric: distance d that satisfies 4 properties

1. $d(x, y) \geq 0$ (non-negativity or separation)
2. $d(x, y) = 0$ if and only if $x = y$ (coincidence axiom)
3. $d(x, y) = d(y, x)$ (symmetry)
4. $d(x, z) \leq d(x, y) + d(y, z)$ (**triangle inequality**)

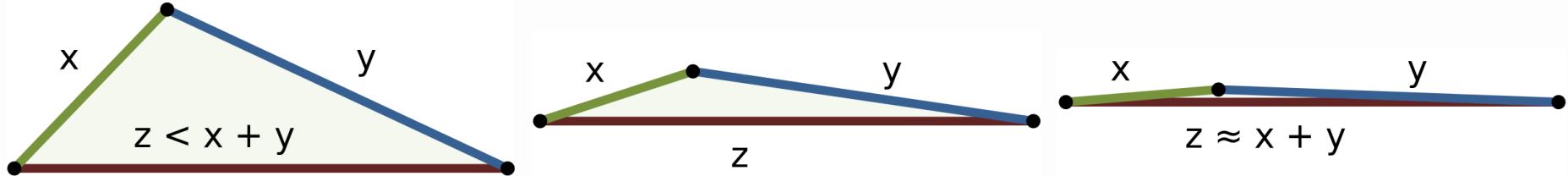


Image source https://en.wikipedia.org/wiki/Triangle_inequality

Metric space

Metric space (S, d) = data space equipped with a metric

- e.g., 3-D Euclidean space or any normed vector space
- no need to be a vector space! (e.g., space of strings + suitable metric)

Why they are so nice?

- many tasks can be performed more efficiently!
- especially similarity search (find nearest neighbours, closest cluster centers, similar documents,...)

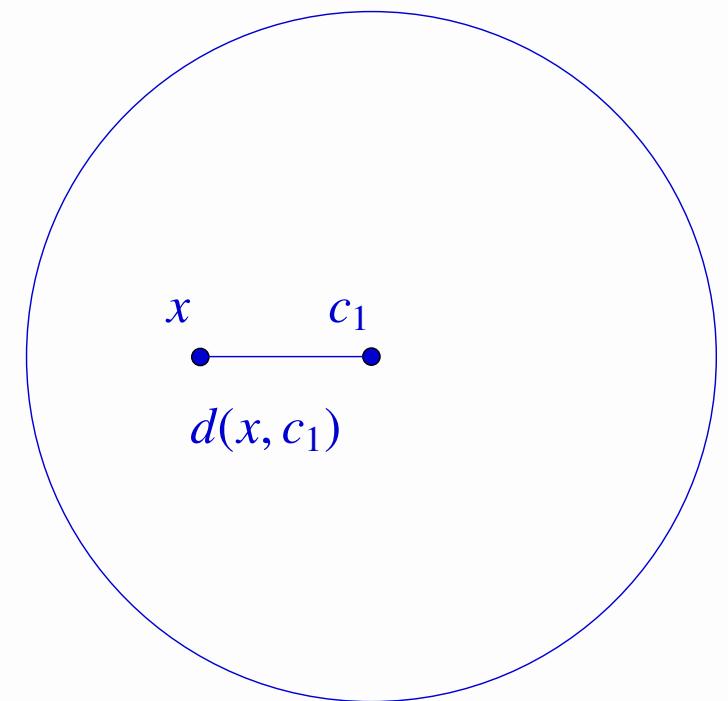
Example how Δ inequality can speed up things

Problem: Given cluster centroids $\mathbf{c}_1, \dots, \mathbf{c}_K$, find the closest \mathbf{c}_i for all data points \mathbf{x} . (d is a metric)

1. Naive solution: calculate all $d(\mathbf{x}, \mathbf{c}_i)$. (nK calculations)
2. Pruning trick: Given $d(\mathbf{c}_i, \mathbf{c}_j)$ for all i, j and $d(\mathbf{x}, \mathbf{c}_1)$ to the currently closest \mathbf{c}_1 .

Test: If $d(\mathbf{c}_1, \mathbf{c}_2) > 2d(\mathbf{x}, \mathbf{c}_1)$, then \mathbf{c}_2 cannot be closer to \mathbf{x} !

If \mathbf{c}_2 was closest to \mathbf{c}_1 , then \mathbf{c}_1 is closest to \mathbf{x} .



c_2 cannot be inside the circle
since $d(c_1, c_2) > 2d(x, c_1)$

Example how Δ inequality can speed up things

More pruning by utilizing upper and lower bounds of distances!

Further reading:

- Elkan: Using the triangle inequality to accelerate k-means. ICML 2003.
- Hamerly: Making k-means even faster. SDM 2010.

Do you know distance or similarity measures for these data types?

- numerical
- categorical
- mixed
- sets
- binary
- strings
- text
- graphs

Multidimensional numerical: L_p -norm

Objects are $\mathbf{x} = (x_1, \dots, x_k)$ and $\mathbf{y} = (y_1, \dots, y_k)$, $x_i, y_i \in \mathbb{R}$

Most common measure L_p -norm or **Minkowski distance**:

$$L_p(\mathbf{x}, \mathbf{y}) = \left(\sum_{i=1}^k |x_i - y_i|^p \right)^{\frac{1}{p}}$$

- different variants by setting p
- e.g., **Euclidean distance** $L_2(\mathbf{x}, \mathbf{y}) = \left(\sum_i |x_i - y_i|^2 \right)^{1/2}$
- **metric, if $p \geq 1$**

Manhattan (“city block”) distance L_1

$$L_1(\mathbf{x}, \mathbf{y}) = \sum_i |x_i - y_i|$$

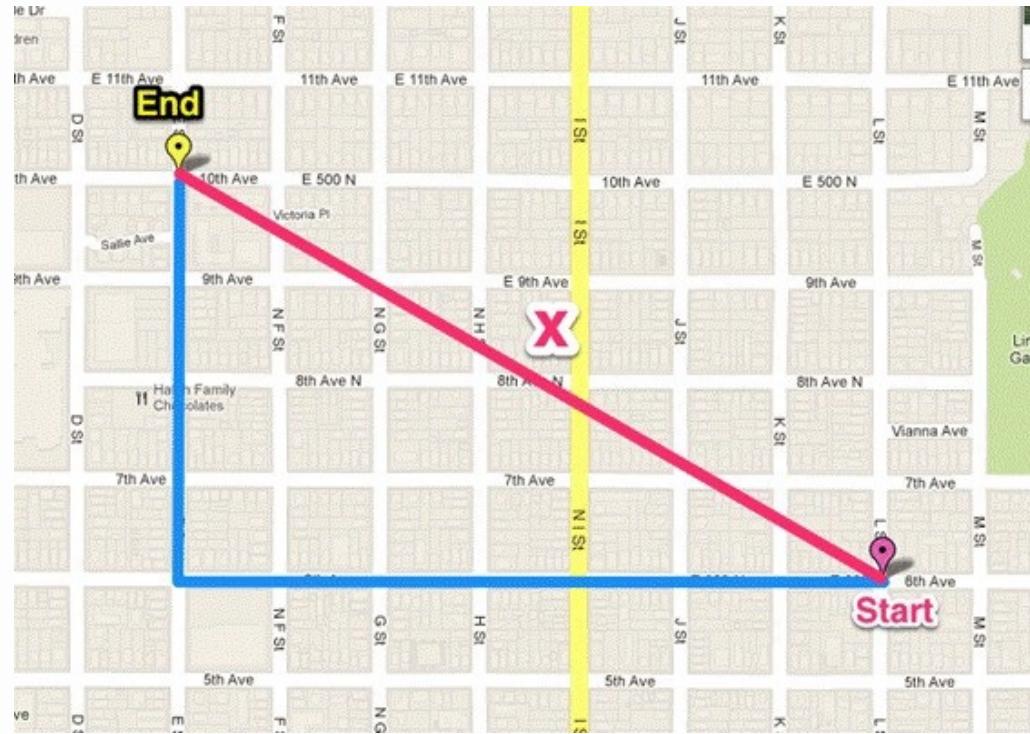
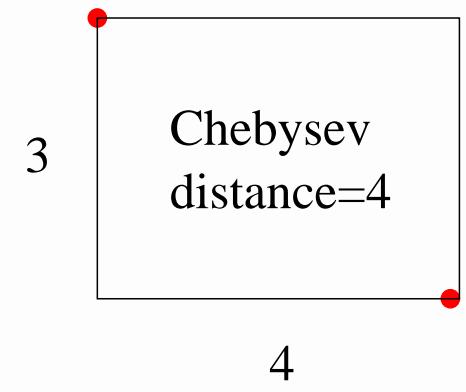
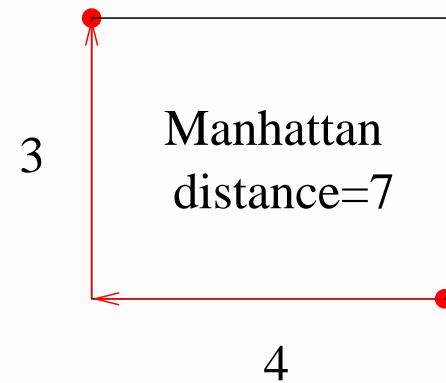
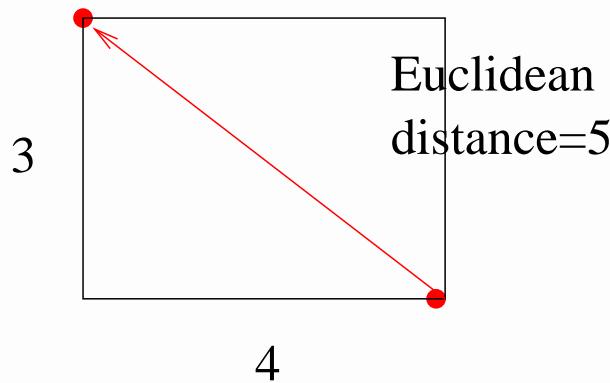


image source <https://medium.com/@paubric/the-square-circle-exploiting-distance-cef434f7f550>

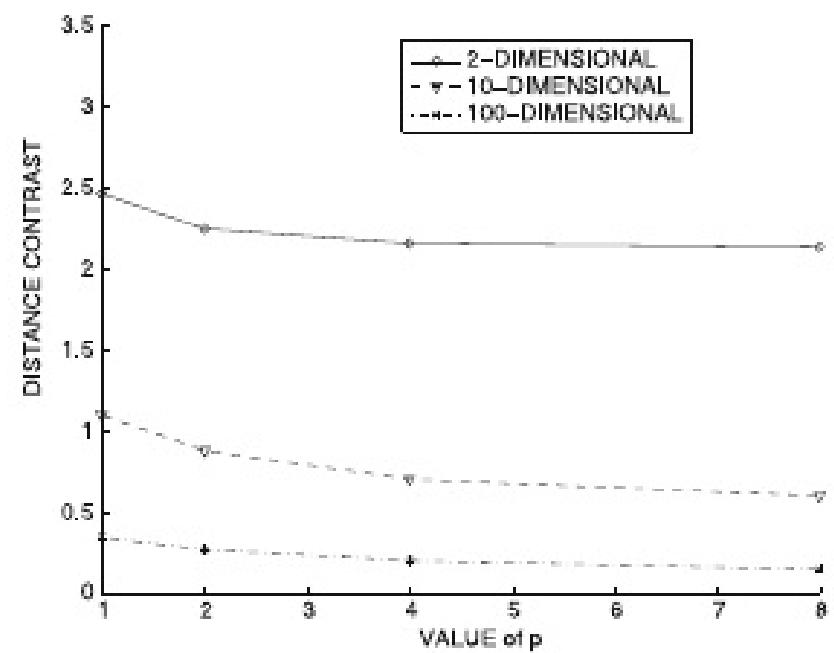
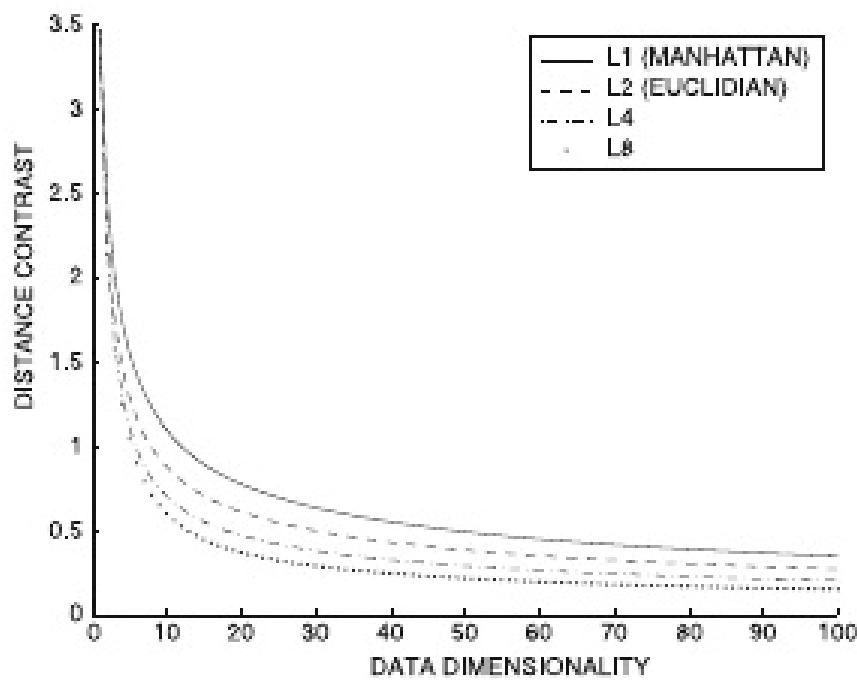
L_p -norms

- $p = 1$: Manhattan distance $L_1(\mathbf{x}, \mathbf{y}) = \sum_i |x_i - y_i|$
- $p = 2$: Euclidean distance $L_2(\mathbf{x}, \mathbf{y}) = \left(\sum_i |x_i - y_i|^2 \right)^{1/2}$
- $p \rightarrow \infty$: Chebyshev distance $L_\infty(\mathbf{x}, \mathbf{y}) = \max_i |x_i - y_i|$



L_p -norms do not work well in high dimensions

Curse of dimensionality: Contrasts $\frac{D_{\max} - D_{\min}}{D_{\text{avg}}}$ between largest and smallest distances disappear. Behaviour in random data:



L_p -norms do not work well in high dimensions

- irrelevant features tend to dominate L_2, \dots, L_∞
- Consider $L_\infty(\mathbf{x}, \mathbf{y})$, when \mathbf{x} and \mathbf{y} have similar value in 999 dimensions but dissimilar in 1 irrelevant attribute!

⇒

- generalized Minkowski distance give weights a_i reflecting importance: $L_p(\mathbf{x}, \mathbf{y}) = (\sum_i a_i |x_i - y_i|^p)^{\frac{1}{p}}$
- fractional L_p quasinorms set $p \in]0, 1[$ (**not metrics**)
- match-based similarity with proximity thresholding

Match-based similarity with proximity thresholding

Observations:

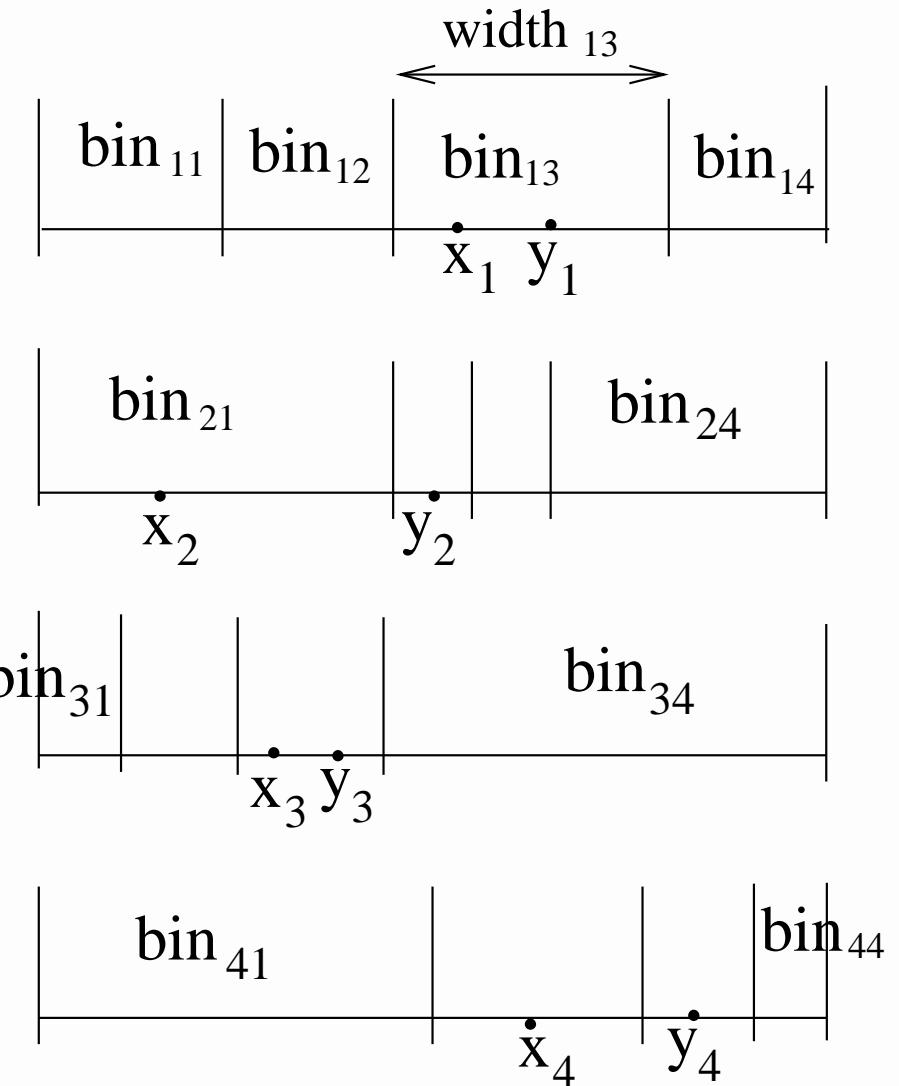
1. Features may be only **locally relevant** (e.g., blood glucose for diabetic patients but not for epileptic).
2. In large dimensions, two objects are unlikely to have similar values, unless the feature is relevant.

⇒ **emphasize dimensions where objects are close/similar!**

(Euclidean and pals do the opposite)

Match-based similarity with proximity thresholding

- discretize all dimensions to m equi-depth bins, bin_{ij} (i =dimension, j =bin number)
- **x and y are in proximity on dimension i , if**
 $x_i, y_i \in bin_{ij}$ for some j
- **proximity set $S(\mathbf{x}, \mathbf{y}, m) =$**
list of dimensions, where
 x_i and y_i in the same bin
e.g., here $S(\mathbf{x}, \mathbf{y}, 4) = \{1, 3\}$



Match-based similarity with proximity thresholding

Similarity measure

$$PSelect(\mathbf{x}, \mathbf{y}, m) = \left[\sum_{i \in S(\mathbf{x}, \mathbf{y}, m); x_i \in bin_{i,j}} \left(1 - \frac{|x_i - y_i|}{width_{i,j}} \right)^p \right]^{1/p}$$

- ignores dimensions where \mathbf{x} and \mathbf{y} not in proximity
- value when i) $\mathbf{x} = \mathbf{y}$? ii) $S(\mathbf{x}, \mathbf{y}, m) = \emptyset$?
- how to choose parameters? ($m \propto k$ + e.g., $p = 1$ or $p = 2$)

Aggarwal & Yu (2000): The IGrid Index: Reversing the Dimensionality Curse For Similarity Indexing in High Dimensional Space.

Cosine similarity and distance

Cosine similarity:

$$\cos(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\| \|\mathbf{y}\|}$$

- suitable for numerical (continuous or integers) and binary data
- in $[-1, 1]$, most similar if $\cos(\mathbf{x}, \mathbf{y}) = 1$
- popular for text documents (their numerical presentation)

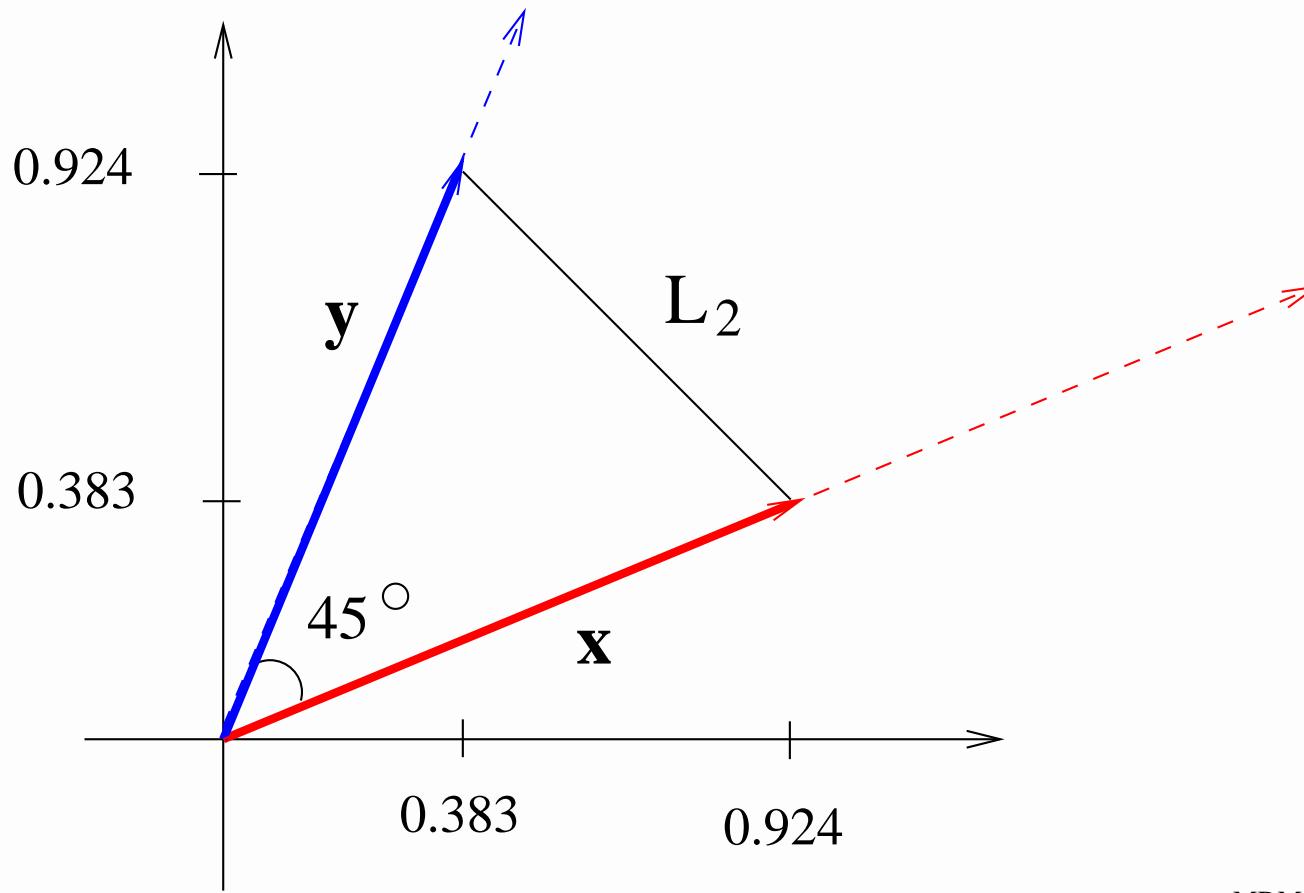
Cosine distance: $1 - \cos(\mathbf{x}, \mathbf{y})$

- $[0, 1]$ if all vector elements non-negative ($x_i \geq 0$)

Cosine similarity and distance

Relationship to Euclidean distance L_2 :

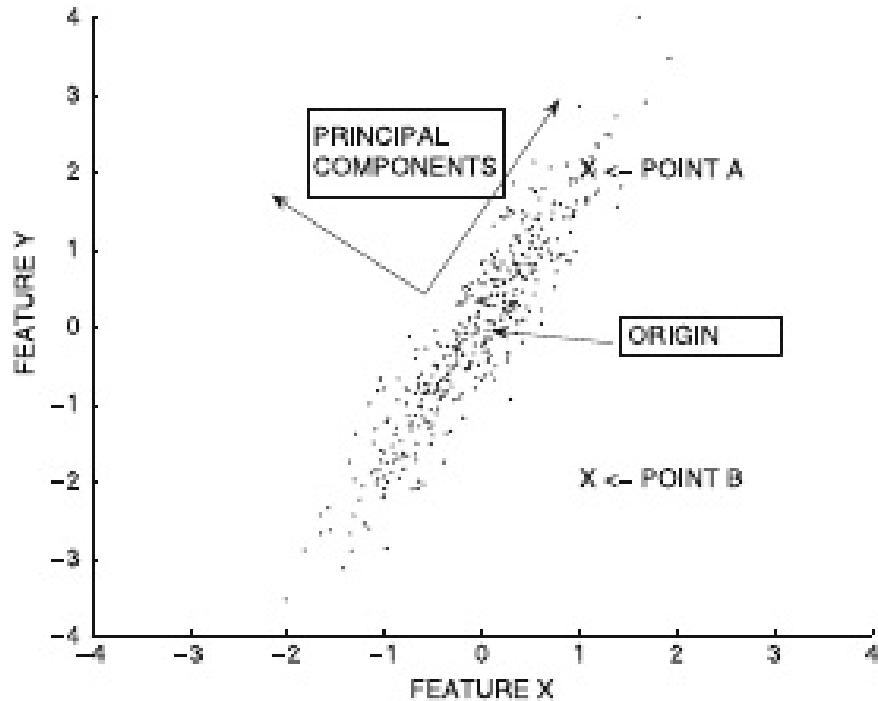
if vectors are normalized (length 1), $L_2^2(\mathbf{x}, \mathbf{y}) = 2(1 - \cos(\mathbf{x}, \mathbf{y}))$



Should the distance reflect data distribution?

Should A and B be equally distant from the origin?

high variance direction
⇒ more likely to be
distant ⇒
could consider A closer
than B ⇒
Mahalanobis distance



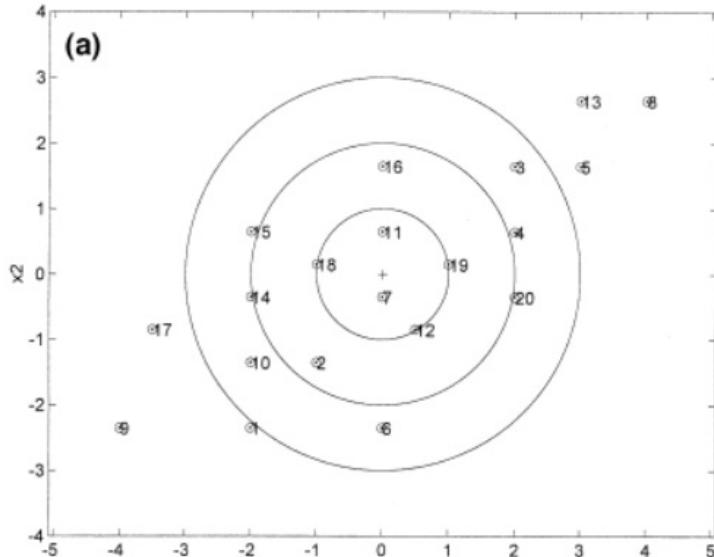
$$Maha(\mathbf{x}, \mathbf{y}) =$$

$$\sqrt{(\mathbf{x} - \mathbf{y})\Sigma^{-1}(\mathbf{x} - \mathbf{y})^T}$$

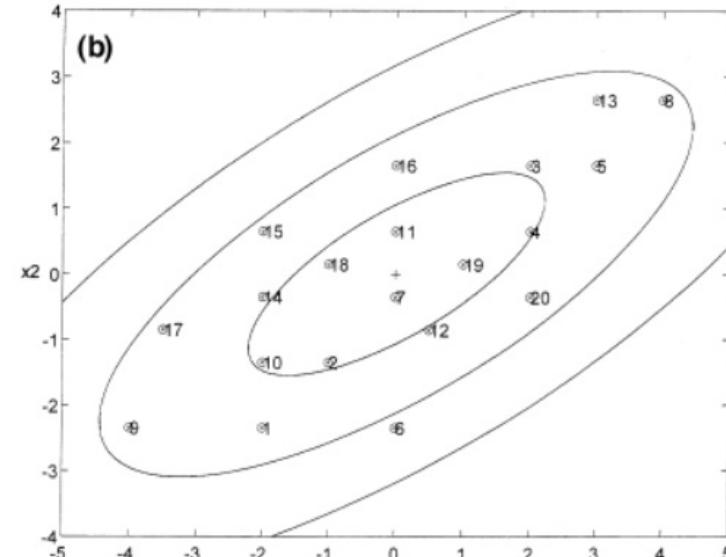
(Σ = covariance matrix)

Mahalanobis distance

$$Maha(\mathbf{x}, \mathbf{y}) = \sqrt{(\mathbf{x} - \mathbf{y})\Sigma^{-1}(\mathbf{x} - \mathbf{y})^T}$$



Points on each circle have the same Euclidean distance to the origin.



Points on each ellipse have the same Mahalanobis distance to the origin.

image source <https://queirozf.com/entries/similarity-measures-and-distances-basic-reference-for-data-science-practitioners>

Should the distance reflect data distribution?

Which pair of points are closest to one another?

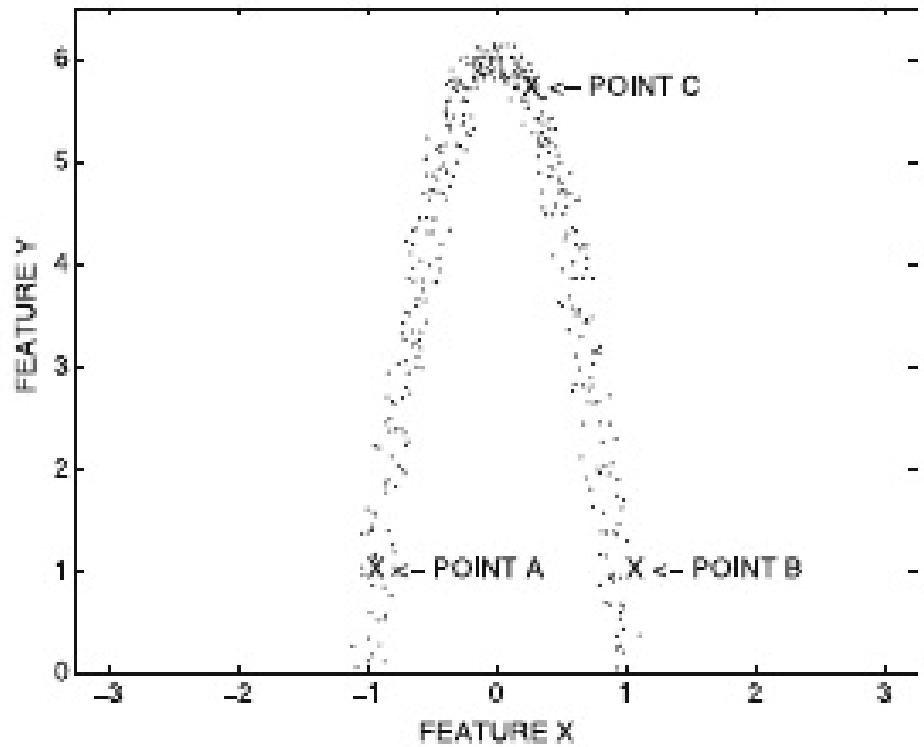


image source Aggarwal 3.2.1.7

Analogy: what is your walking distance to the other shore?



Idea: Measure distances along shortest paths in a nearest neighbour graph

ISOMAP method:

1. Create a nearest neighbour graph $G = (V, E)$ where each $v \in V$ is connected to K nearest neighbours and edge weights represent distances.
2. For any points $v_1, v_2 \in V$

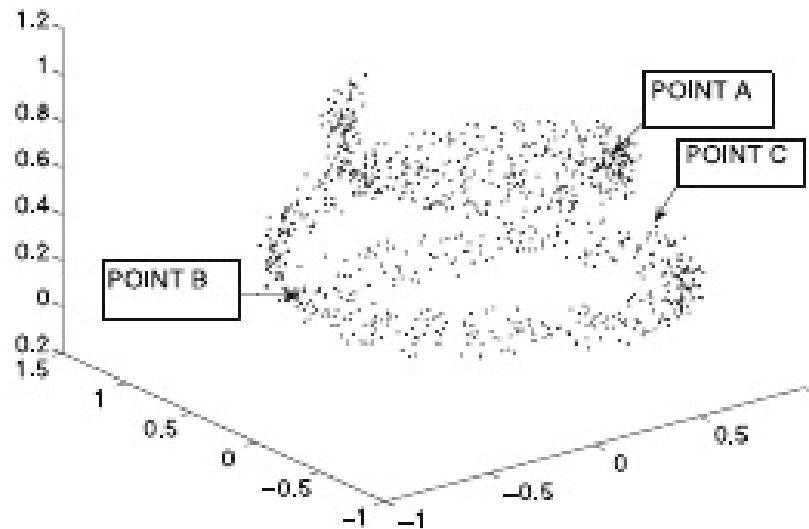
$$Dist(v_1, v_2) = |\text{shortest-path}(v_1, v_2)|$$

3. Optional step: embed the data into multidimensional space with multidimensional scaling \rightarrow lower dimensional representation

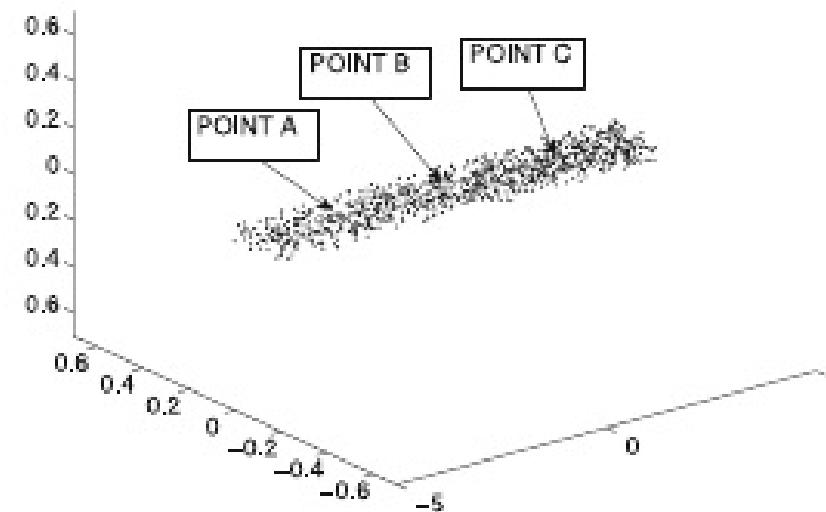
Use either $Dist(v_1, v_2)$ or L_p distances in the new space

ISOMAP

The data shape becomes straightened out:



(a) A and C seem close
(original data)

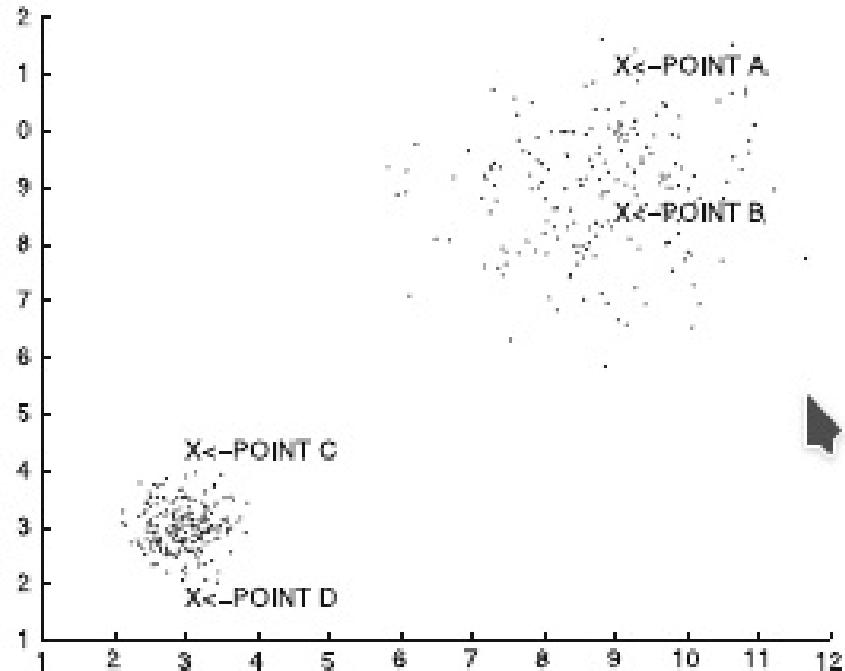


(b) A and C are actually far away
(ISOMAP embedding)

Figure 3.5: Impact of *ISOMAP* embedding on distances

Should the distance reflect data distribution?

Should $d(A, B) < d(C, D)$ or vice versa?



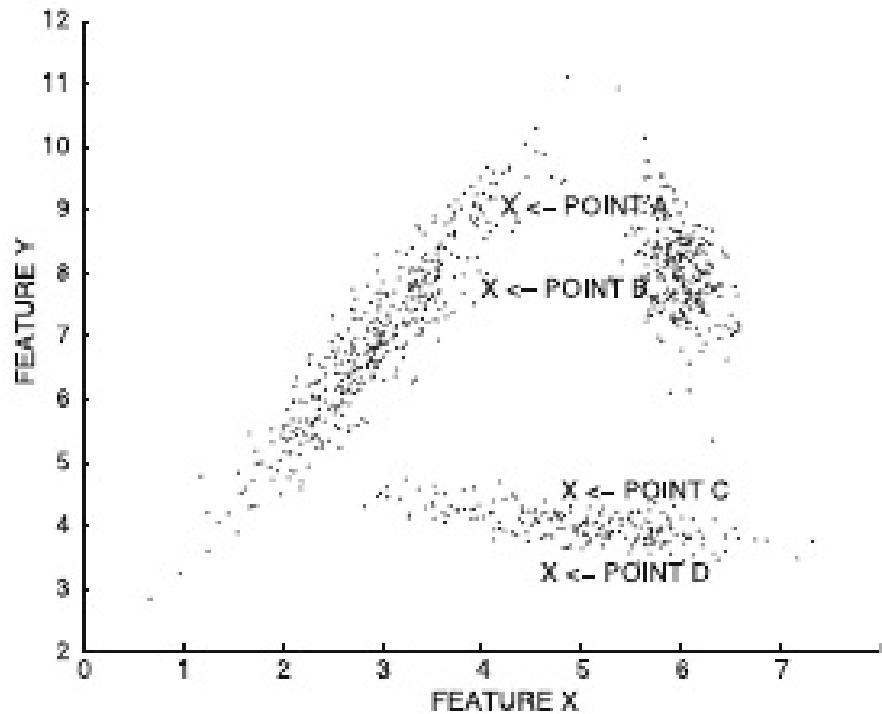
(a) local density variation

shared nearest-neighbour
similarity = number of shared
neighbours
⇒ similarity graph

Read Aggarwal 3.2.1.8

Should the distance reflect data distribution?

Should $d(A, B) < d(C, D)$ or vice versa?



(b) local orientation variation

- partition data and use local statistics to adjust distances (local Mahalanobis)
- but partitioning already requires distance measures!

Read Aggarwal 3.2.1.8

Categorical data: similarity

Generic function:

$$sim(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^k w_i s(x_i, y_i)$$

- typically weight $w_i = \frac{1}{k}$ (k =number of features)
- many choices for s , e.g., in **overlap similarity** s is

$$s(x_i, y_i) = \begin{cases} 1 & \text{if } x_i = y_i \\ 0 & \text{otherwise} \end{cases} \Rightarrow$$

overlap similarity = fraction of dimensions where \mathbf{x} and \mathbf{y} have an equal value

Categorical data: similarity

Or take into account frequency of value:

$$p_i(x_i) = \frac{fr(A_i = x_i)}{n} = \text{fraction of records having } A_i = x_i$$

Goodall measure (its one variant):

$$s(x_i, y_i) = \begin{cases} 1 - p_i^2(x_i) & \text{if } x_i = y_i \\ 0 & \text{otherwise} \end{cases}$$

Further reading Boriah et al. (2008): Similarity measures for categorical data: A comparative evaluation.

Task

Create a similarity graph using overlap similarity. Include only edges where similarity is $\geq 2/3$. Which foxes are most similar to Bella? What if you use the Goodall measure instead?

name	sex	colour	character
Bella	F	red	tame
Molly	F	red	shy
Teddy	M	red	tame
Ruby	F	red	brave
Coco	F	silver	cool
Max	M	silver	brave

$$\text{overlap} = \frac{\#\text{(overlapping feature values)}}{\#\text{features}}$$

$$\text{Goodall} = \frac{\sum_{A_i \text{ shared}} (1 - p_i^2(\text{shared value}))}{\#\text{features}}$$

Task

$$\text{overlap} = \frac{\#\text{(overlapping feature values)}}{3}$$

pair	common	overlap	Goodall
Bella–Molly	F, red	2/3	
Bella–Teddy	red, tame	2/3	
Bella–Ruby	F, red	2/3	
Molly–Ruby	F, red	2/3	

Task

$$\text{Goodall} = \frac{\sum_{A_i \text{ shared}} (1 - p_i^2(\text{shared value}))}{\# \text{features}}$$

$p_1(F)=2/3, p_1(M)=1/3, p_2(\text{red})=2/3, p_2(\text{silver})=1/3,$
 $p_3(\text{tame})=p_3(\text{brave})=1/3, p_3(\text{shy})=p_3(\text{cool})=1/6$

$$1-p_1^2(F)+1-p_2^2(\text{red})=10/9$$

$$1-p_2^2(\text{red})+1-p_3^2(\text{tame})=13/9$$

pair	common	overlap	Goodall
Bella–Molly	F, red	2/3	10/27
Bella–Teddy	red, tame	2/3	13/27
Bella–Ruby	F, red	2/3	10/27
Molly–Ruby	F, red	2/3	10/27

Similarity in mixed data (without transformations)

Give weights to numerical and categorical components:

$$sim(\mathbf{x}, \mathbf{y}) = \lambda \cdot NumSim + (1 - \lambda) \cdot CatSim$$

- How to choose λ ? ($\lambda \in [0, 1]$)
- e.g., fraction of numerical features in data
- $NumSim$ and $CatSim$ often in different scales ⇒
 - calculate standard deviations (σ_N and σ_C) of pairwise similarities with $NumSim$ and $CatSim$

$$sim(\mathbf{x}, \mathbf{y}) = \lambda \cdot NumSim/\sigma_N + (1 - \lambda) \cdot CatSim/\sigma_C$$

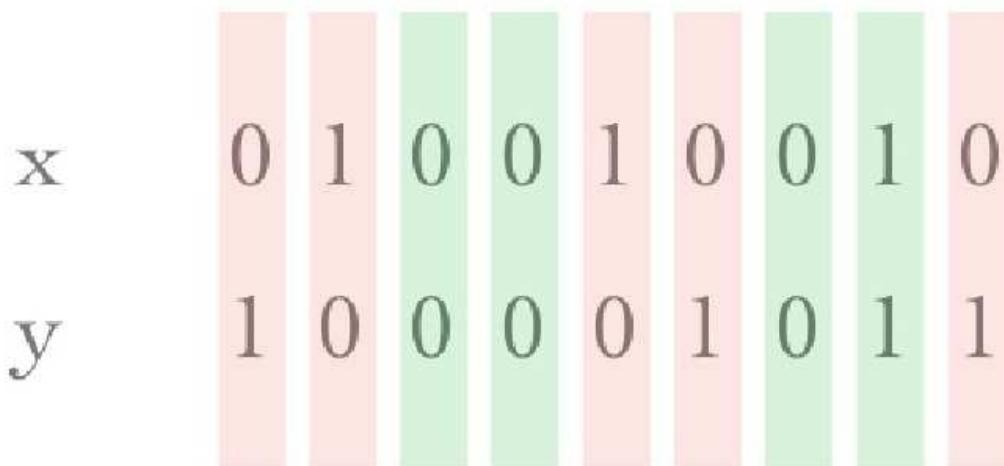
Binary data: distance and similarity

Data points \mathbf{x} and \mathbf{y} are bit strings (length k)

Hamming distance = L_1 norm for binary data

$$L_1(\mathbf{x}, \mathbf{y}) = \sum_i |x_i - y_i|$$

=number of positions where bits differ



Hamming distance 5

image source CS-E4600
fall 2019 slides

Set data can be presented as binary

e.g., basket1: {white bread, cheese} \Rightarrow 001100000000...

	low fat milk	apple juice	white bread	edam cheese	oranges
basket1	0	0	1	1	0
basket2	1	1	0	0	0
basket3	0	1	0	1	0
basket4	1	0	1	0	1
basket5		:			

- transactions (like market baskets)
- occurrence of words in documents
- over-expressed or under-expressed genes in samples

Set data often **very sparse** (= most values are 0s) \Rightarrow number of common elements more important

Hamming distance for transaction data?

995 common bits

x: 111111... y: 111111...

11000001111
111111100000



1. Two sets with 1000 items and 995 common
2. Two sets with 5 items, but none common

Both have Hamming distance 10

x: 1111100000
y: 0000011111

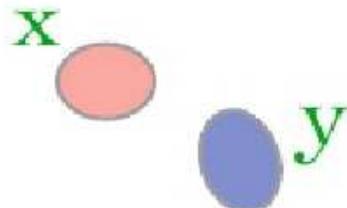


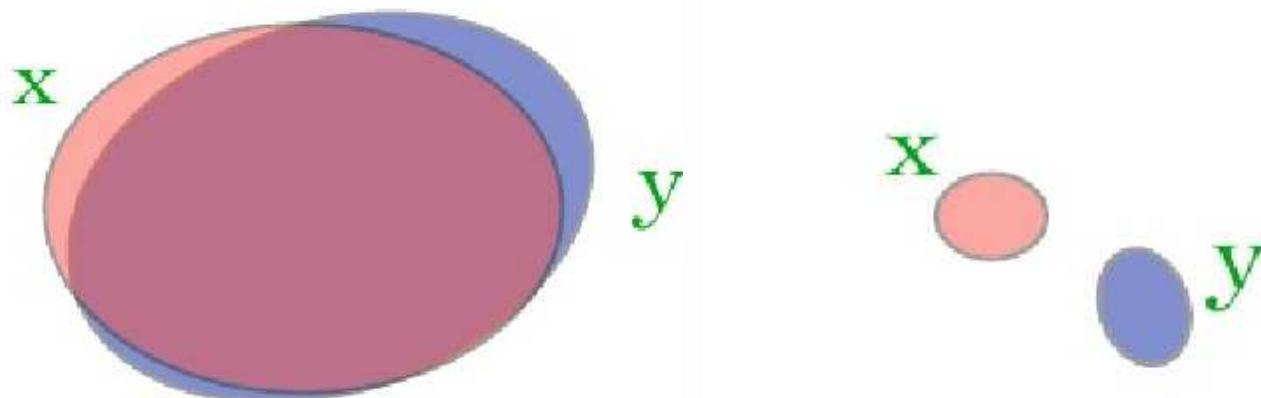
image source CS-E4600 fall 2019 slides/Aris Gionis

Jaccard coefficient for set similarity

Given sets x and y

$$J(x, y) = \frac{|x \cap y|}{|x \cup y|}$$

- treats 0s and 1s differently
- Previous example, case 1: $J = \frac{995}{1005} \approx 0.99$, case: 2
 $J = 0$

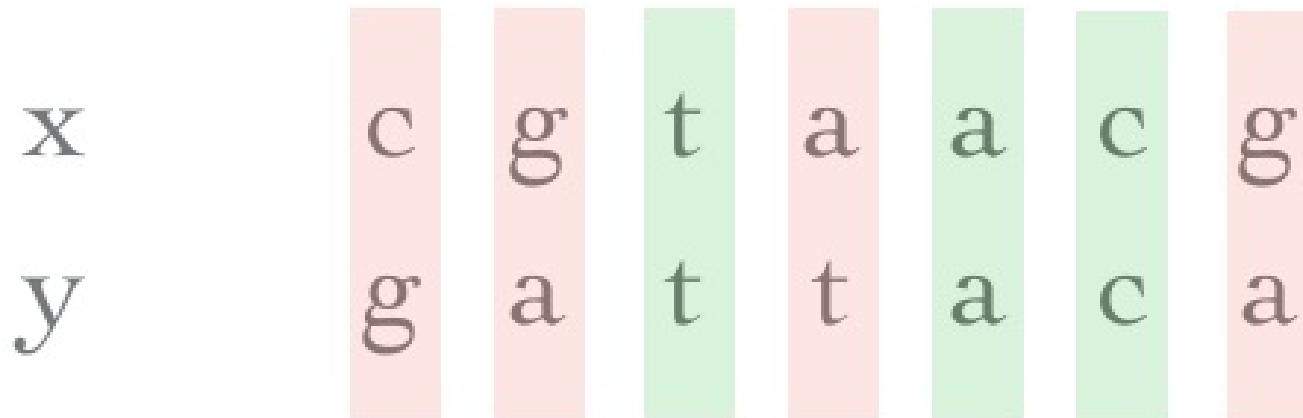


String data: distance

Given strings x and y of the same length.

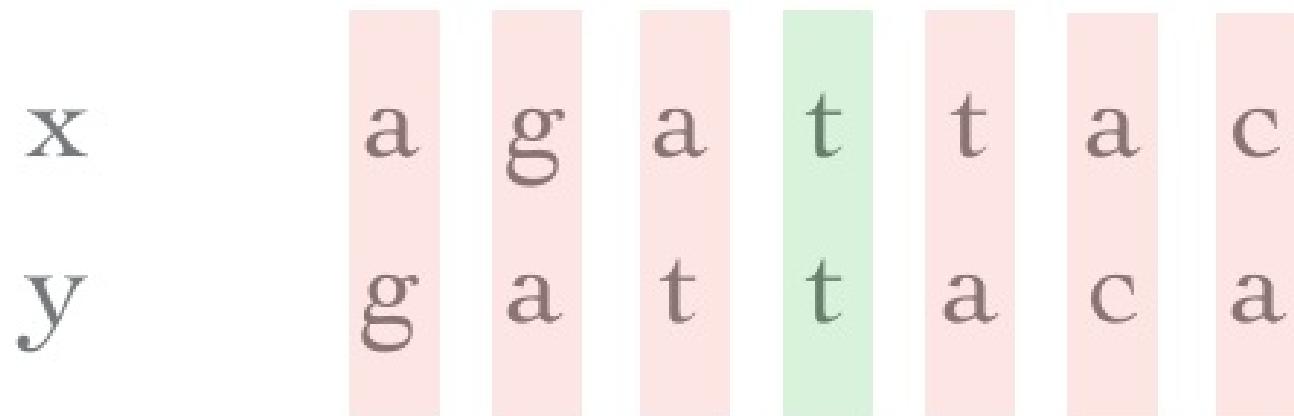
Modification of the Hamming distance

- add 1 for all positions that are different



Is Hamming distance good for strings?

- Strings must have equal length
- Punishes a lot for small typos:



string Hamming distance = 6

String edit distance

Given two strings x and y , try to change one to another!

- only single-character edits are allowed
 - insert character
 - delete character
 - substitute character
- edit distance=minimum cost of such operations
- **Levenshtein distance**=minimum number of such operations (unit costs)
- edit operations can have different costs w_{ins} , w_{del} , w_{sub}
- **metric**, if positive costs and each operation has an inverse operation with the same cost

String edit distance examples

x	a	g	a	t	t	a	c
							remove a
y	g	a	t	t	a	c	add a

Levensteihnn(kitten, sitting)=3:

1. kitten → sitten (substitute "s" for "k")
2. sitten → sittin (substitute "i" for "e")
3. sittin → sitting (insert "g" at the end)

Text data: similarity between documents

Let's present text documents as **document-term matrices**.

- \mathbf{x} and \mathbf{y} are m -dimensional vectors (m = lexicon size)
- x_i = frequency of term i in the document \mathbf{x}
 - alternatively tf-idf value (tf-idf presentation) or binary value (Boolean model)
- then take **cosine similarity**:

$$\cos(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\| \|\mathbf{y}\|}$$

- in the Boolean model, Jaccard coefficient also possible

Task: Simplify the equation of cosine similarity when data is binary

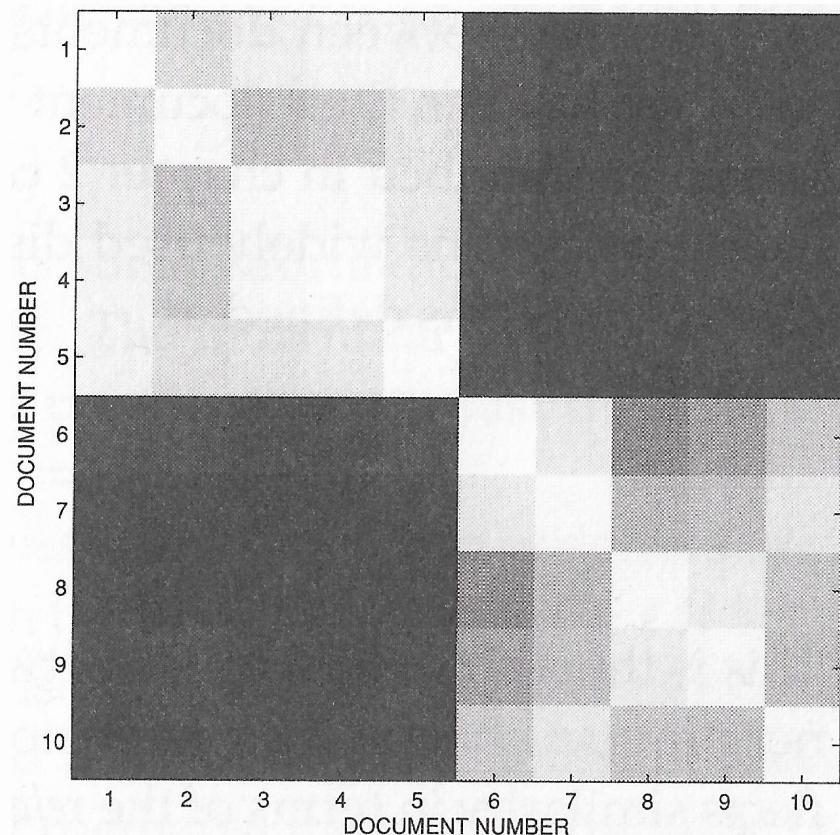
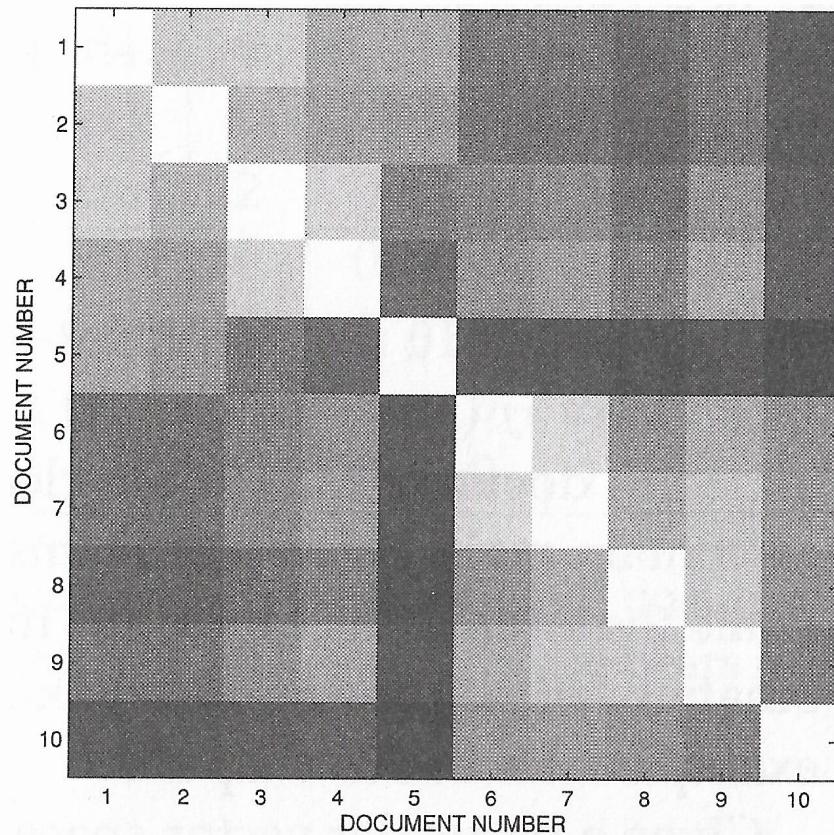
Text data: Example (Hand et al. 2001)

	t1	t2	t3	t4	t5	t6
d1	24	21	9	0	0	3
d2	32	10	5	0	3	0
d3	12	16	5	0	0	0
d4	6	7	2	0	0	0
d5	43	31	20	0	3	0
d6	2	0	0	18	7	16
d7	0	0	1	32	12	0
d8	3	0	0	22	4	2
d9	1	0	0	34	27	25
d10	6	0	0	17	4	23

source: Hand, Mannila, Smyth: Principles of data mining, 2001

Text data: Example (Hand et al. 2001)

Left: Euclidean distance (bright=small distance), right: cosine similarity (bright=large similarity)



Other data types

See the text book!

- time series: Ch 3.4
- graphs: Ch 3.5 and later in the course

Warning: There are many variants of the same measures and the names are not fixed! Give always the equation of the measure you use (+ a literature reference)!

Summary

- Choose distance and similarity measures carefully!
- Curse of dimensionality → for multidimensional data consider L_p with small p , cosine or match-based similarity
- If the distribution is very heterogenous, it is beneficial to adjust to local variations in distances (but costs!)
- Metric distances can speed similarity search, but non-metrics may perform better in high dimensions

Metrics: Examples

1. $d(x, y) \geq 0$ (non-negativity or separation)
2. $d(x, y) = 0$ if and only if $x = y$ (coincidence axiom)
3. $d(x, y) = d(y, x)$ (symmetry)
4. $d(x, z) \leq d(x, y) + d(y, z)$ (triangle inequality)

E.g., if $x, y \in \mathbb{R}$

1. $|x - y|$ is metric (check all properties)
2. $|x^2 - y^2|$ not metric (coincidence deoesn't hold)
3. $(x - y)^2$ not metric (triangle inequality doesn't hold)

Metrics: proving

1. To show that d is metric, show that all 4 properties hold for arbitrary $\mathbf{x}, \mathbf{y}, \mathbf{z}$.
2. To show that d is not a metric, one counter-example (with any $\mathbf{x}, \mathbf{y}, \mathbf{z}$), not satisfying any one of the 4 properties suffices.

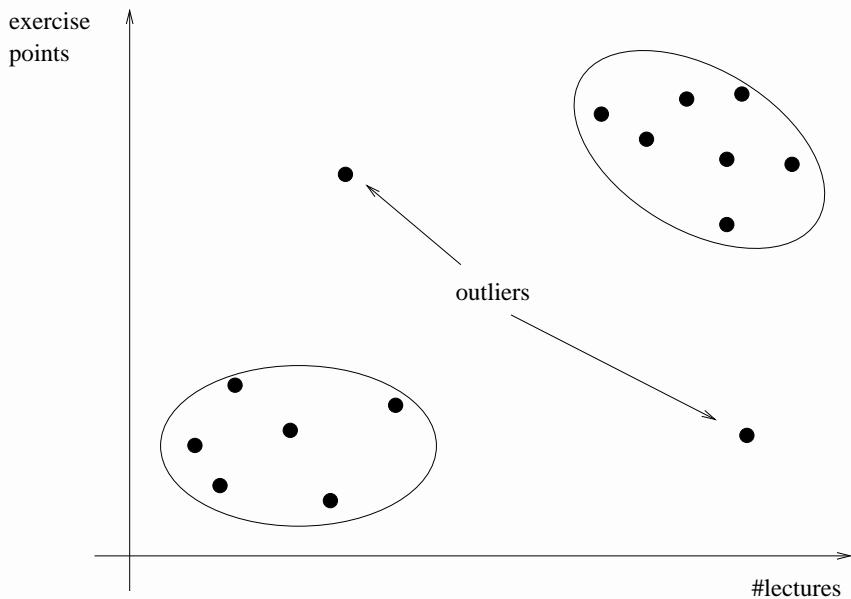
Why fractional L_p are not metrics? (now $p \in]0, 1[$)

Counter-example, when $p = 0.5$: let $\mathbf{x} = (4, 0)$, $\mathbf{y} = (0, 3)$, $\mathbf{z} = (0, 0)$.

Then $d(\mathbf{x}, \mathbf{y}) = (\sqrt{4} + \sqrt{3})^2 = 4 + 2\sqrt{4}\sqrt{3} + 3 > 4 + 3 = (\sqrt{4})^2 + (\sqrt{3})^2 = d(\mathbf{x}, \mathbf{z}) + d(\mathbf{z}, \mathbf{y})$. (Triangle inequality doesn't hold.)

Clustering

Intuitively: Partition data \mathcal{D} into K clusters C_1, \dots, C_K such that points in each cluster are similar to one another but dissimilar to points in other clusters.



- **hard clustering:** each point belongs to one cluster
- **soft clustering:** a point can belong to multiple clusters with different probabilities or weights

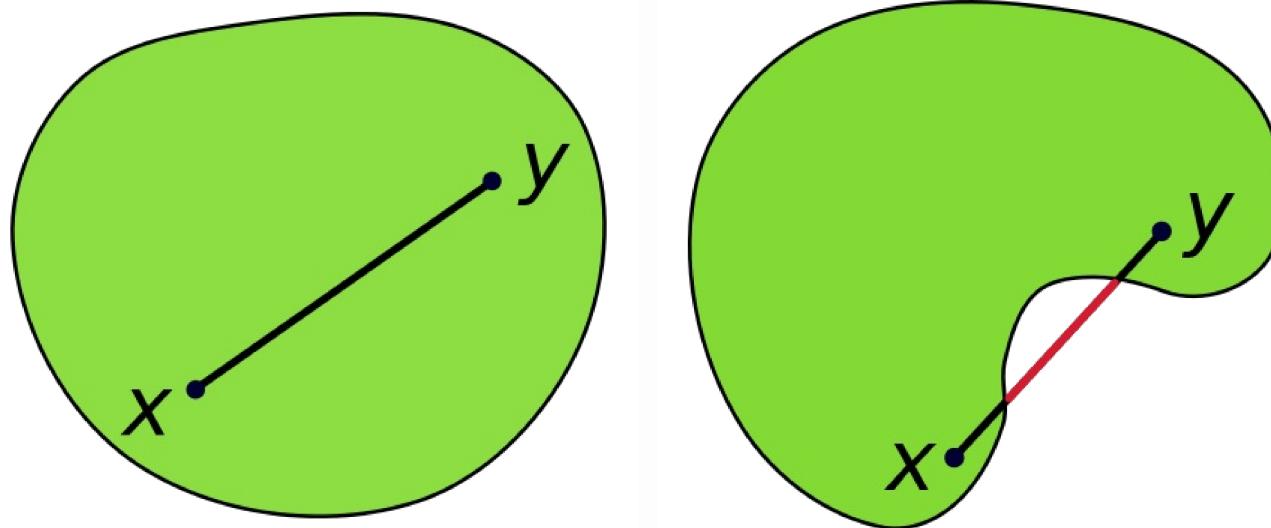
What is the objective of clustering?

What kind of clusters should be found?

- shape: is the shape of clusters fixed (e.g., hyperspherical) or arbitrary?
- size: balanced clusters or clusters of different sizes?
- density: equal or variable?
- overlapping or well-separated clusters?
- outliers?

⇒ different methods, objective functions and distance measures

Shapes: convex or non-convex?

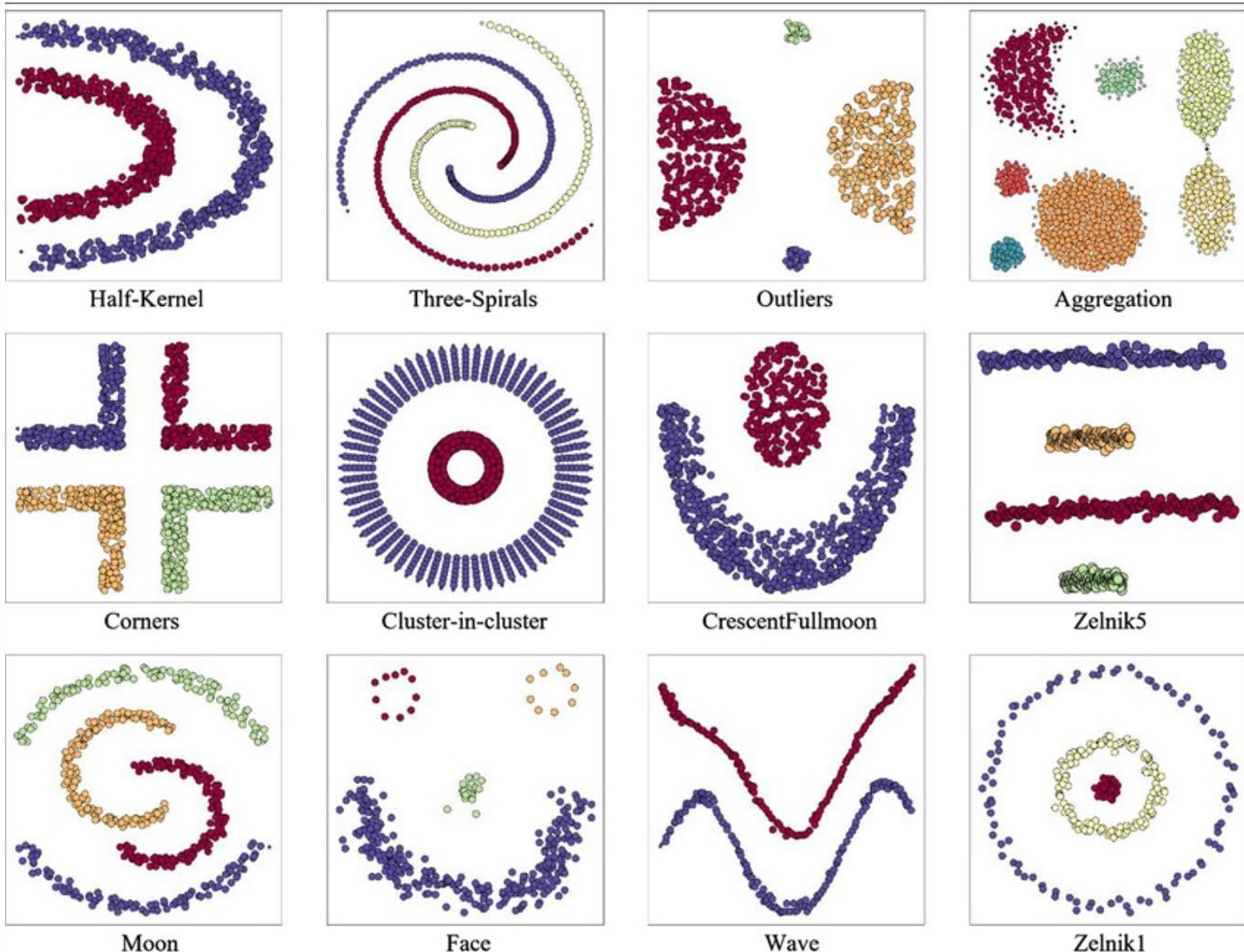


E.g., hyperspheres, hyperrectangles, and Voronoi cells are convex.

Image source: wikipedia,

https://en.wikipedia.org/wiki/Convex_set

Examples of tricky cluster structures (Senol 2023)



What is needed?

- distance measure d (or similarity measure)
- distance measure D for inter-cluster distances
 - sometimes needed
- vector space representation of \mathcal{D} ?
 - sometimes needed
 - sometimes a similarity or distance graph suffices
- objective (score) function to evaluate clustering
 - algorithm tries to optimize this
 - not always explicit
- number of clusters K (often needed)

Examples of objective functions

Usually combine two objectives: **minimize within-cluster-variation wc** and **maximize between-cluster variation bc**

Let $\mathbf{C} = \{C_1, \dots, C_K\}$ clusters, $\mathbf{c}_1, \dots, \mathbf{c}_K$ their centroids and d distance function. Examples of wc :

$$wc(\mathbf{C}) = \sum_{i=1}^K wc(C_i) = \sum_{i=1}^K \sum_{\mathbf{x} \in C_i} d^2(\mathbf{x}, \mathbf{c}_i) \rightarrow \text{hyperspherical clusters}$$

$$wc(C_p) = \max_i \overbrace{\min_{\mathbf{y} \in C_p} \{d(\mathbf{x}_i, \mathbf{y}) \mid \mathbf{x}_i \in C_p, \mathbf{x}_i \neq \mathbf{y}\}}^{\mathbf{x}_i \text{ 's distance to its nearest neighbour in } C_p} \rightarrow \text{elongated clusters}$$

Examples of objective functions

Let $\mathbf{C} = \{C_1, \dots, C_K\}$ clusters, $\mathbf{c}_1, \dots, \mathbf{c}_K$ their centroids and d distance function. Example of bc :

$$bc(\mathbf{C}) = \sum_{1 \leq i < j \leq K} d^2(\mathbf{c}_i, \mathbf{c}_j)$$

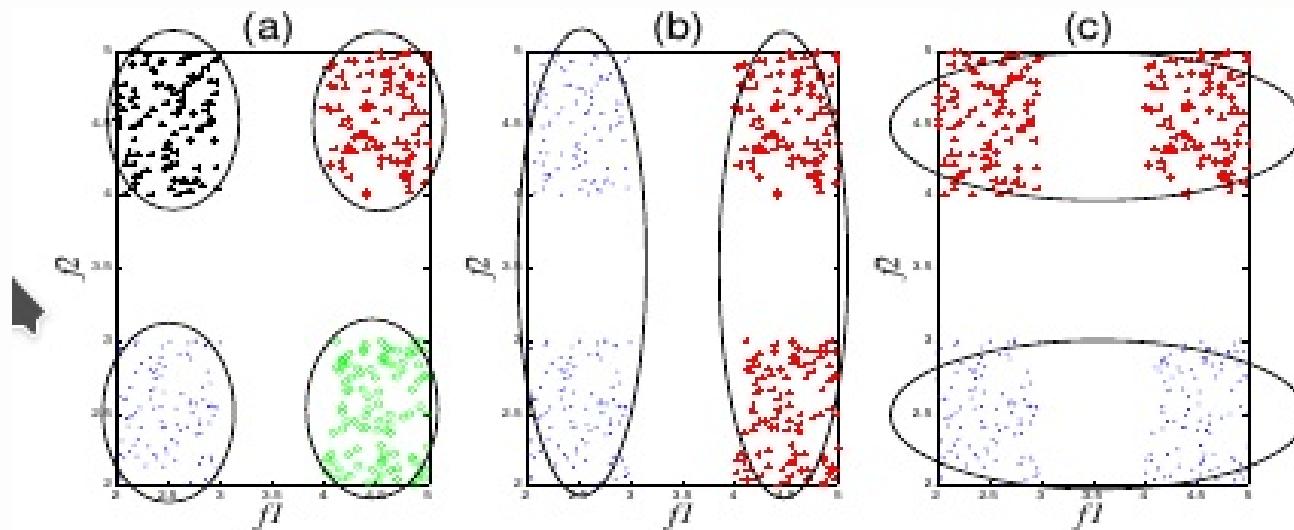
An example of an overall measure is **K -means criterion**:

$$SSE(\mathbf{C}) = \sum_{i=1}^K \sum_{\mathbf{x} \in C_i} L_2^2(\mathbf{x}, \mathbf{c}_i)$$

(minimizing SSE minimizes within-cluster variance and maximizes between-cluster variance)

Clusters depend on the features!

Example: Features f_1 and f_2 distinguish 4 clusters, while f_1 alone or f_2 alone distinguish 2 clusters:



⇒ Is there **clustering tendency** when the data is presented with the given features?

Source: Aleyani et al. (2018): Feature Selection for Clustering: A Review

Preprocessing has a crucial role in clustering!

- feature extraction
- feature selection and dimension reduction
- to scale or not to scale?
 - if features have very different scales, some scaling usually needed
 - **but sometimes normalization distorts separation**

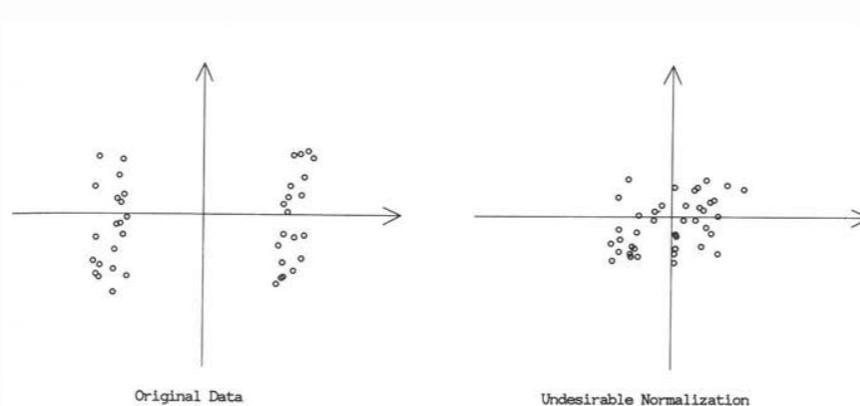


image source: Jain and Dubes: Algorithms for clustering data. 1988

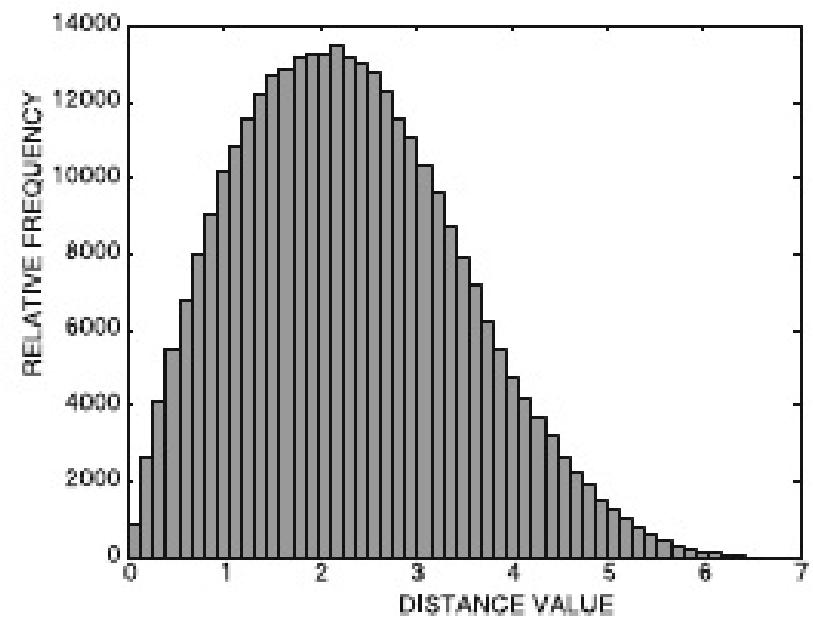
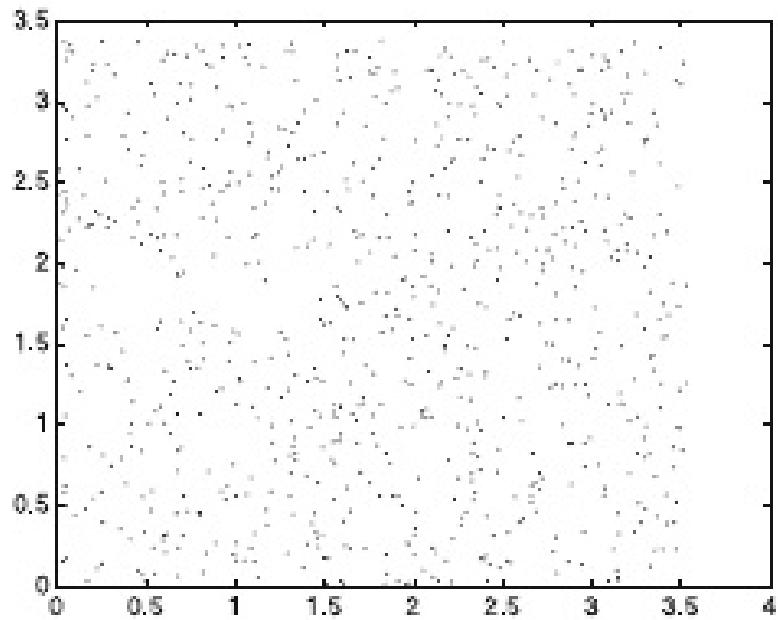
How to study clustering tendency and choose features?

Approaches:

1. Visual inspection of pairwise distance distributions
 - only hints
2. Filtering methods, e.g.,
 - Entropy-based measures
 - Hopkins statistic
3. Wrapper models + cluster validation indices
 - e.g., average silhouette, Calinski-Harabasz, Davies-Bouldin, and external indices → next lecture

1. Visual inspection: Distance distributions

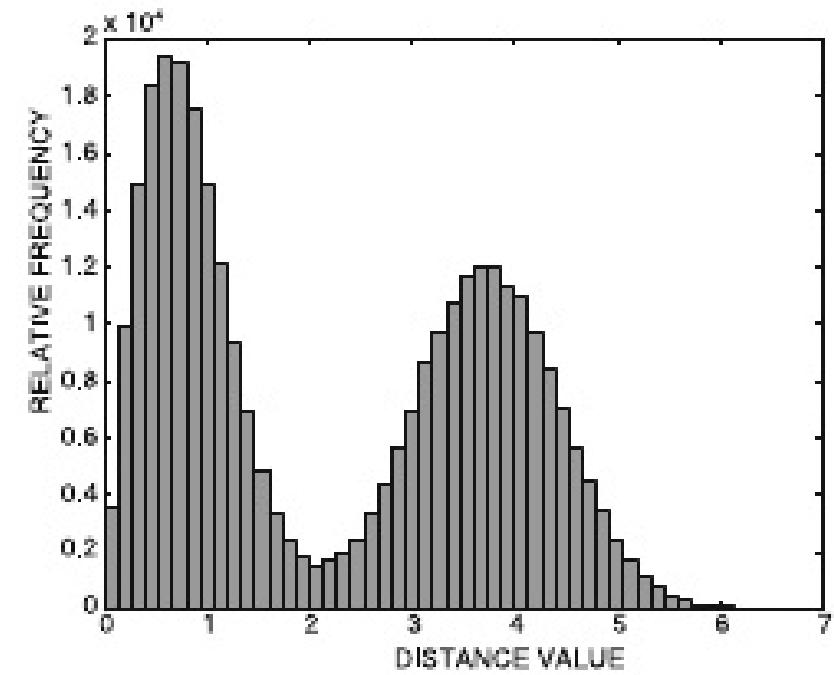
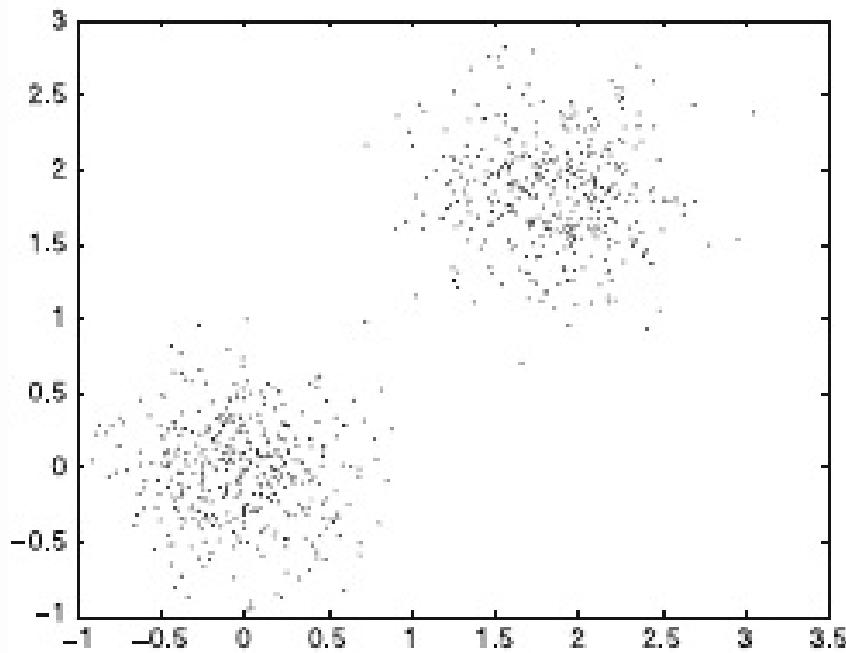
Plot a histogram of pairwise distances in data. What the distribution looks if there are no clusters?



Source: Aggarwal Ch 6

Distance distributions (cont'd)

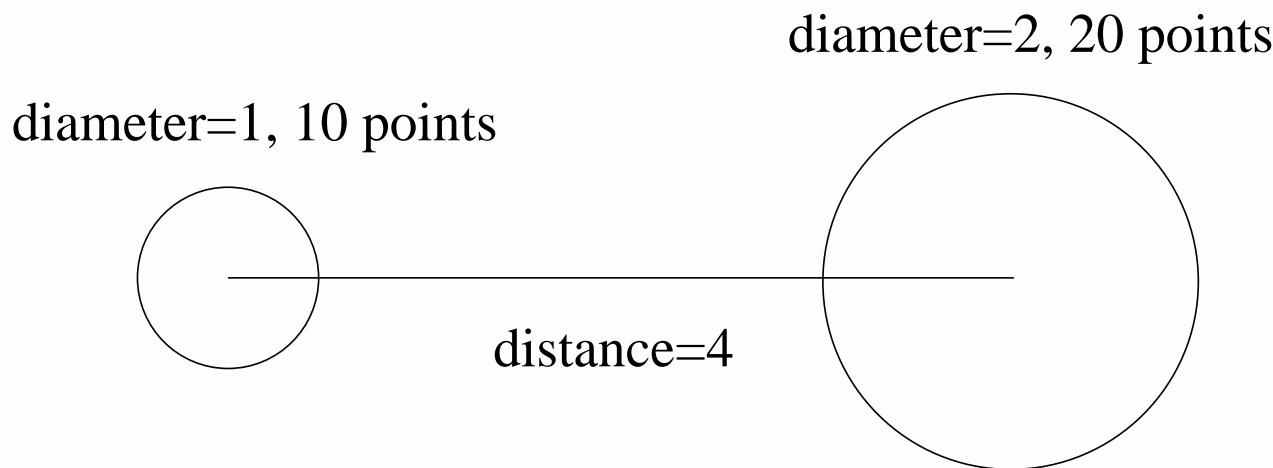
Distribution has more peaks if there are clear clusters!
Why?



Distance distributions: Task

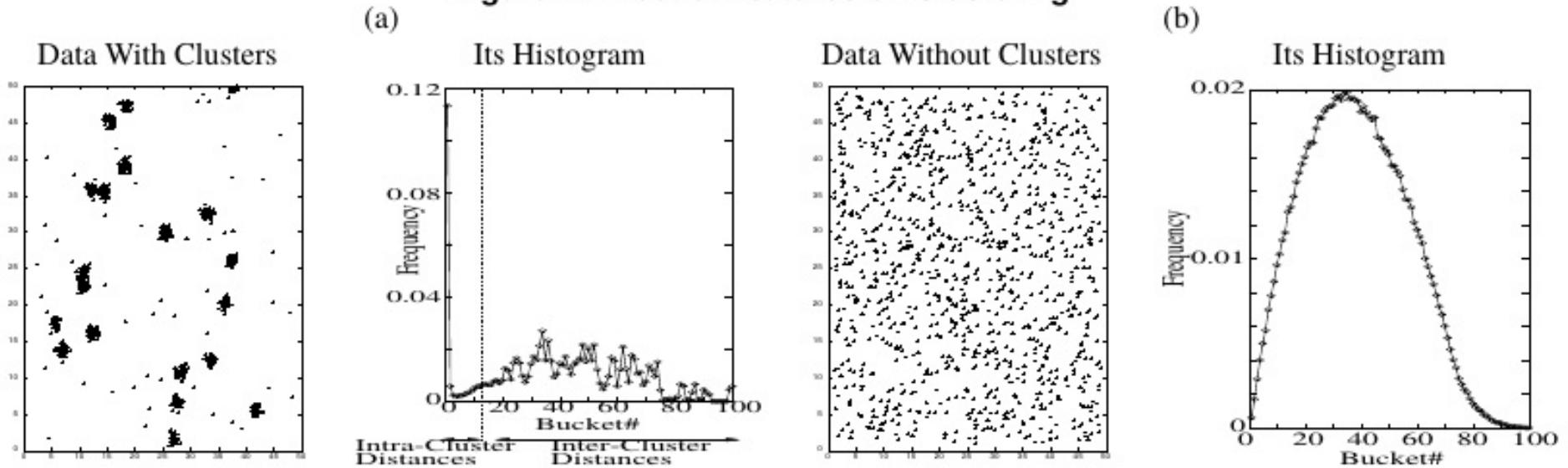
Assume that points are distributed evenly inside each cluster (nothing outside).

What are the ranges of intra-cluster and inter-cluster distances? What does the pairwise distance distribution look like?



Distance distributions: Another example

Figure 1. Effect of Features on Clustering



Source: Dash et al.: Feature selection for clustering – a filter solution.
ICDM, 2002.

2.1 Entropy-based measures

Idea: In random data (uniform distribution), the entropy is high, and in clustered data low.

Approach 1:

- Discretize data into m multidimensional grid regions
 p_i =fraction of data points in region i
- evaluate probability-based entropy

$$E = - \sum_{i=1}^m [p_i \log(p_i) + (1 - p_i) \log(1 - p_i)]$$

Note: Independent, binary region variables (region is occupied with probability p_i or empty with $1 - p_i$).

Entropy-based measures (cont'd)

Problems:

- p_i can be hard to estimate accurately in large dimensionality
- how to choose m ?
- m should be approximately the same for different feature subsets
 - e.g., for each of k dimensions, $\lceil m^{1/k} \rceil$ bins

Entropy-based measures (cont'd)

Approach 2:

- calculate pairwise distances between points
- discretize distances onto m bins
- p_i =fraction of distances in the i th bin
- calculate E

$$E = - \sum_{i=1}^m [p_i \log(p_i) + (1 - p_i) \log(1 - p_i)]$$

Entropy-based measures: choosing features

Often iterative, greedy search, either:

1. Forward selection: at each round add the best feature
 - largest decrease in entropy; **or**
2. Backward selection: at each round drop the worst feature
 - largest increase in entropy

More on entropy-based methods: Aggarwal 6.2.1.3 and
Dash et al.: Feature Selection for Clustering – A Filter
Solution. ICDM, 2002.

2.2 Hopkins statistic

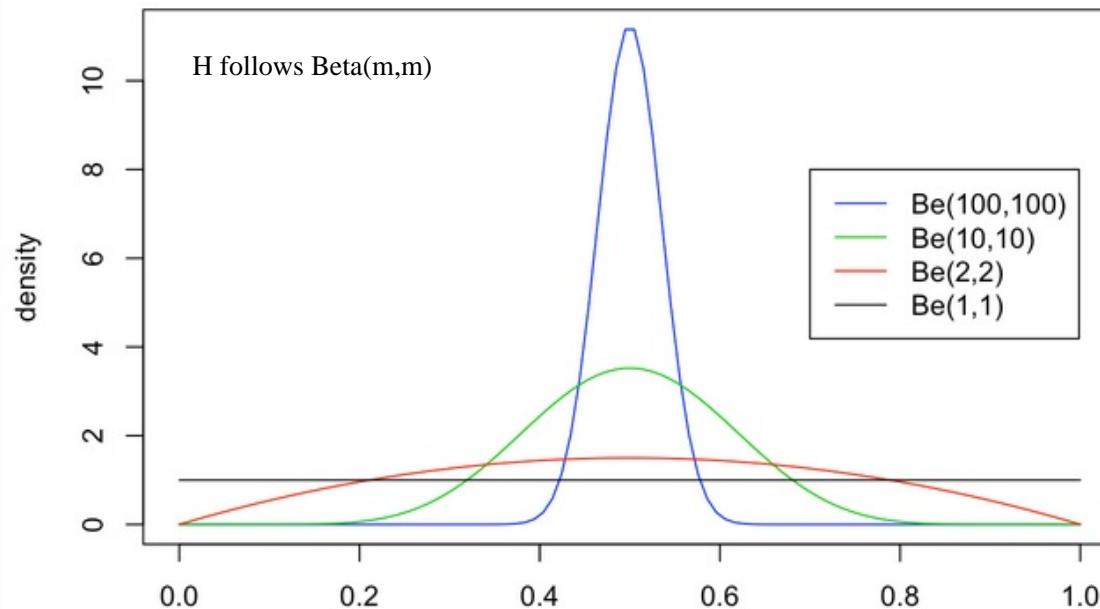
Idea: Compare nearest neighbour distances from the original data and random data points.

- Take a sample R of size r from original data \mathcal{D}
- Generate random data (from uniform distribution) and take a sample S of size r from it
- Calculate for all $x \in R$ distances to their nearest neighbours (in \mathcal{D}). Let these be $\alpha_1, \dots, \alpha_r$
- Calculate for all $x \in S$ distances to their nearest neighbours (in \mathcal{D}). Let these be β_1, \dots, β_r

Hopkins statistic H

$$H = \frac{\sum_{i=1}^r \beta_i}{\sum_{i=1}^r (\alpha_i + \beta_i)}$$

- if \mathcal{D} has uniform distribution, $H \approx 0.5$
- if there are clusters, H approaches 1



source: betadistr.eps <https://stephens999.github.io/fiveMinuteStats/beta.html>

m = sample size (our r)
MDM course Aalto 2023 – p.20/25

Hopkins statistic: Problems

1. distance distribution often very different in the center of data than on edges
⇒ choose sample points inside a hypersphere centered at the mean of data and containing 50% of data points
2. results vary with different executions
⇒ repeat multiple times and calculate average

3. Wrapper models and validation indices

Idea: Iteratively cluster data with different feature sets and use validity indexes to find good features.

First approach:

- Cluster data and calculate some internal cluster validity index
 - can't try all feature subsets → use e.g., greedy heuristic
 - results depend on the validity criterion (and clustering method)

Wrapper models and validation indices (cont'd)

Second approach:

- Create artificial class labels and identify discriminative features in a supervised manner
 - cluster data and use cluster identifies as class labels
 - evaluate each feature separately utilizing class labels (goodness measures for classification)
 - circular definition: features are good if the clustering is good, but good clustering requires good features

Summary

- Try to understand your clustering objective
- How to evaluate clustering tendency (given features)?

Further reading:

- Gan, Ma, Wu: Data clustering – theory, algorithms, and applications. SIAM 2007.
- Jain and Dubes: Algorithms for clustering data. Prentice-Hall 1988. (math properties, clustering tendency)

References

- Senol (2023): MCMSTClustering: defining non-spherical clusters by using minimum spanning tree over KD-tree-based micro-clusters. Neural Computing and Applications 35(1):1-21.

Lecture programme

Lecture 3:

- Dimension reduction with PCA and SVD
- Clustering I (clustering tendency)

Book: Sec. 2.4.3, 6.1–6.2

- L4: Clustering II (K -representatives, hierarchical)
- L5: Clustering III (spectral, validation)

Why all eigen and singular stuff?

Goal: nice low-dimensional representation for data, when

- original data high-dimensional
- only pairwise distances/similarities known

Recall: Given $n \times n$ matrix \mathbf{A} , λ is eigenvalue and $\mathbf{v} \neq \mathbf{0}$ corresponding eigenvector, if $\mathbf{Av} = \lambda\mathbf{v}$

Good news: Everything will be real-valued!

- $k \times k$ covariance matrix and $n \times n$ Laplacian matrices positive semidefinite $\stackrel{a}{\Rightarrow}$ real λ s and \mathbf{v} s
- $n \times k$ data matrix real \Rightarrow singular vectors real (singular values always real)

$$^a \mathbf{z}^T \mathbf{A} \mathbf{z} \geq 0 \text{ for all real } \mathbf{z} \neq \mathbf{0}$$

Dimension reduction: motivation

1. **Curse of dimensionality:** hard to distinguish close and far neighbours in high dimensional data! → how to find clusters??
2. **Redundancy in data:** If features strongly correlated, the same information can be presented with a smaller number of features
 - **intrinsic dimensionality** r may be $r \ll d$
⇒ Idea: reduce dimensionality by removing this redundancy!

Principal component analysis (PCA) assumptions

1. High **variance** reflects important structures of data.
2. Data can be presented well as a **linear** combination of suitable **orthogonal** basis vectors (PCs).

Idea: Given $n \times d$ data and suitable $d \times r$ ($r < d$) matrix \mathbf{P}_r , new data will be $n \times r$ matrix \mathbf{DP}_r

Remember: These assumptions may not always reflect reality!

PCA intuition

Rotate axes to match highest variance directions + choose the best new axes

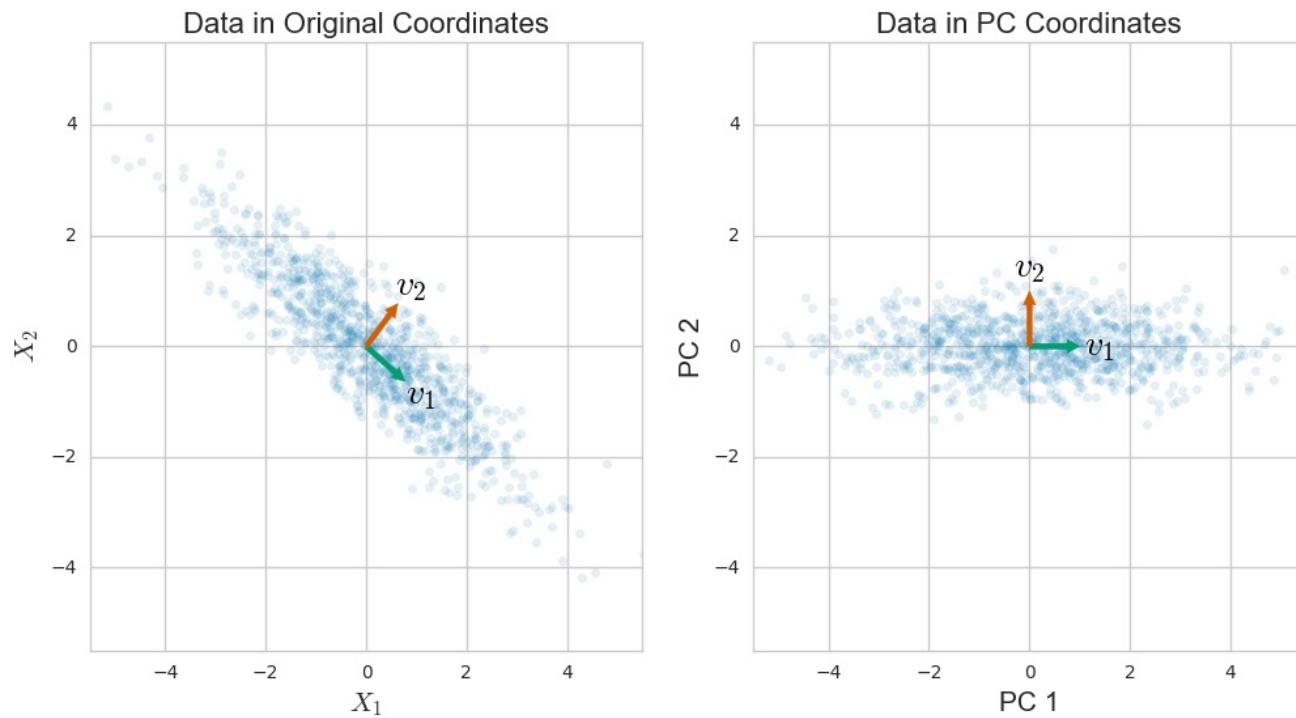


Image source: <https://intoli.com/blog/pca-and-svd>

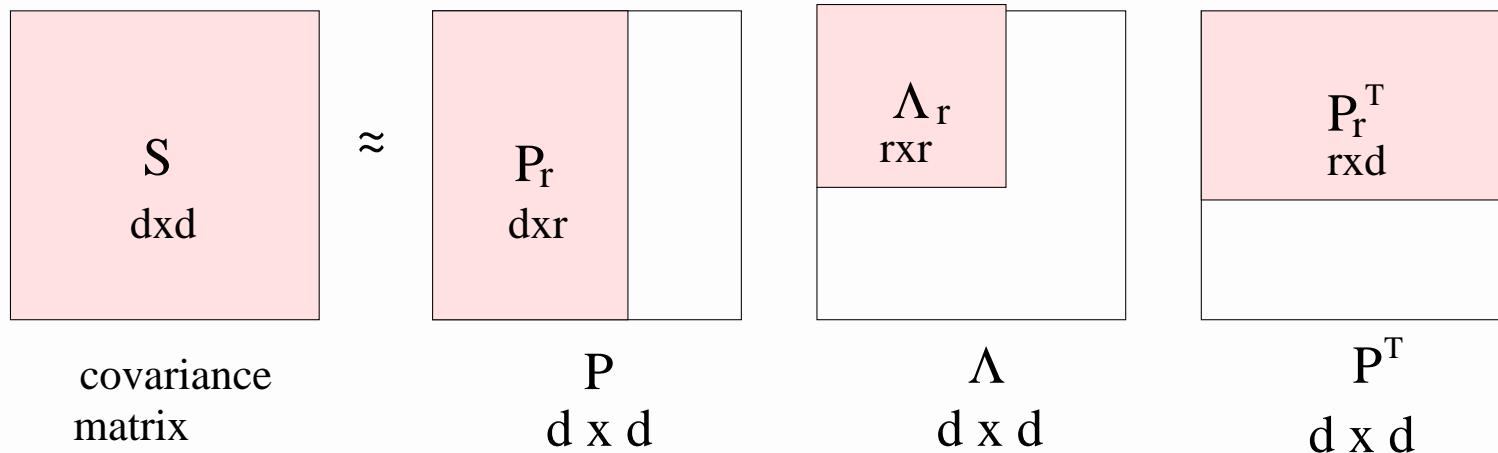
PCA: Use eigen-decomposition

- Assume \mathbf{D} mean-centered. Covariance matrix
 $\mathbf{C} = \frac{1}{n-1}\mathbf{D}^T\mathbf{D}$ ^a
- since \mathbf{C} positive semidefinite, it can be diagonalized:
 $\mathbf{C} = \mathbf{P}\boldsymbol{\Lambda}\mathbf{P}^T$
 - \mathbf{P} 's columns = \mathbf{C} 's orthonormal eigenvectors
 - $\boldsymbol{\Lambda}$ diagonal, Λ_{ii} = eigenvalues
- transformed data $\mathbf{D}' = \mathbf{DP}$
 - $\boldsymbol{\Lambda}$ new covariance matrix (diagonal, i.e., no correlations)

^aunbiased estimate; for large n also $\frac{1}{n}\mathbf{D}^T\mathbf{D}$ ok

PCA: Dimension reduction

- Assume Λ ordered into decreasing order by $\lambda_i = \Lambda_{ii}$
- Keep only r largest $(\lambda_1, \dots, \lambda_r)$ + corresponding eigenvectors
- approximate data $\mathbf{D}' = \mathbf{DP}_r$



Example: Dimension reduction with PCA

Data D

$$\begin{bmatrix} 2 & 2 & 1 & 2 & 0 & 0 \\ 2 & 3 & 3 & 3 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 2 & 2 & 2 & 3 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 2 & 1 & 2 \end{bmatrix}$$

Mean-centered data ^a

$$\begin{bmatrix} 0.83 & 0.67 & -0.17 & 0.00 & -0.50 & -0.67 \\ 0.83 & 1.67 & 1.83 & 1.00 & -0.50 & -0.67 \\ -0.17 & -0.33 & -0.17 & -1.00 & -0.50 & -0.67 \\ 0.83 & 0.67 & 0.83 & 1.00 & 0.50 & 0.33 \\ -1.17 & -1.33 & -1.17 & -1.00 & 0.50 & 0.33 \\ -1.17 & -1.33 & -1.17 & 0.00 & 0.50 & 1.33 \end{bmatrix}$$

^arounded for the slide presentation only!

Mean vector $\approx [1.167, 1.333, 1.167, 2, 0.5, 0.667]$

Example (continued)

Covariance matrix \square^a :

$$\mathbf{C} \approx \begin{bmatrix} 0.97 & 1.13 & 0.97 & 0.6 & -0.3 & -0.53 \\ 1.13 & 1.47 & 1.33 & 0.8 & -0.4 & -0.67 \\ 0.97 & 1.33 & 1.37 & 0.80 & -0.3 & -0.53 \\ 0.60 & 0.80 & 0.80 & 0.80 & 0.00 & 0.00 \\ -0.30 & -0.40 & -0.30 & 0.00 & 0.30 & 0.40 \\ -0.53 & -0.67 & -0.53 & 0.00 & 0.40 & 0.67 \end{bmatrix}$$

^arounded for the slide presentation only!

Example (continued)

Eigenvalues λ_i : 4.43, 0.89, 0.17, 0.066, 0.014, 1.2e-17

Eigenvectors \mathbf{v}_i (column vectors) ^a:

$$\begin{bmatrix} 0.44 & 0.058 & 0.68 & -0.34 & -0.47 & 1.8e-15 \\ 0.57 & 0.027 & 0.092 & 0.18 & 0.54 & 0.58 \\ 0.53 & -0.14 & -0.71 & -0.26 & -0.35 & 1.5e-15 \\ 0.32 & -0.62 & 0.14 & 0.37 & 0.16 & -0.58 \\ -0.15 & -0.42 & 0.041 & -0.78 & 0.43 & 8.4e-17 \\ -0.26 & -0.65 & 0.052 & 0.19 & -0.38 & 0.58 \end{bmatrix}$$

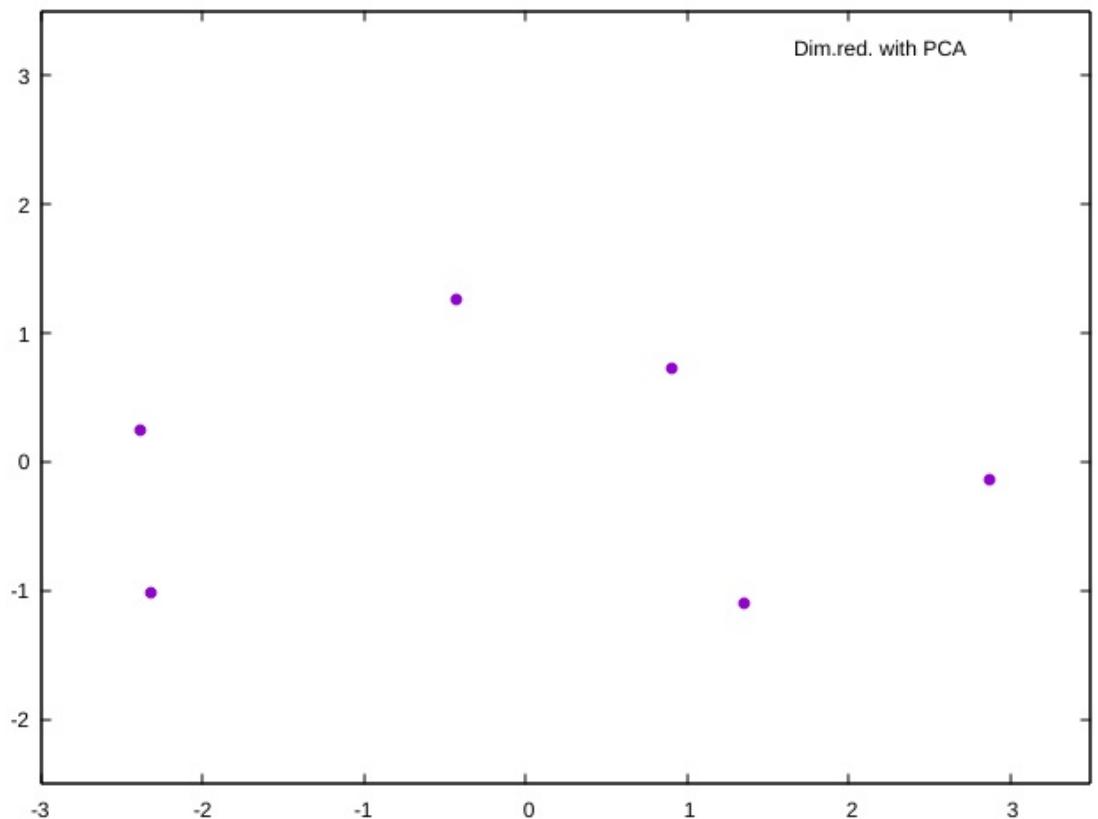
What shall we do if we want a 2-dimensional representation of data?

^aAll rounded for the presentation

Example (continued)

Since λ_1 and λ_2 largest, set $\mathbf{P}_2 = [\mathbf{v}_1, \mathbf{v}_2]$.

$$\mathbf{D}' = \mathbf{D}\mathbf{P}_2 \approx \begin{bmatrix} 0.91 & 0.73 \\ 2.87 & -0.14 \\ -0.43 & 1.26 \\ 1.35 & -1.09 \\ -2.38 & 0.25 \\ -2.32 & -1.01 \end{bmatrix}$$



When PCA can fail?

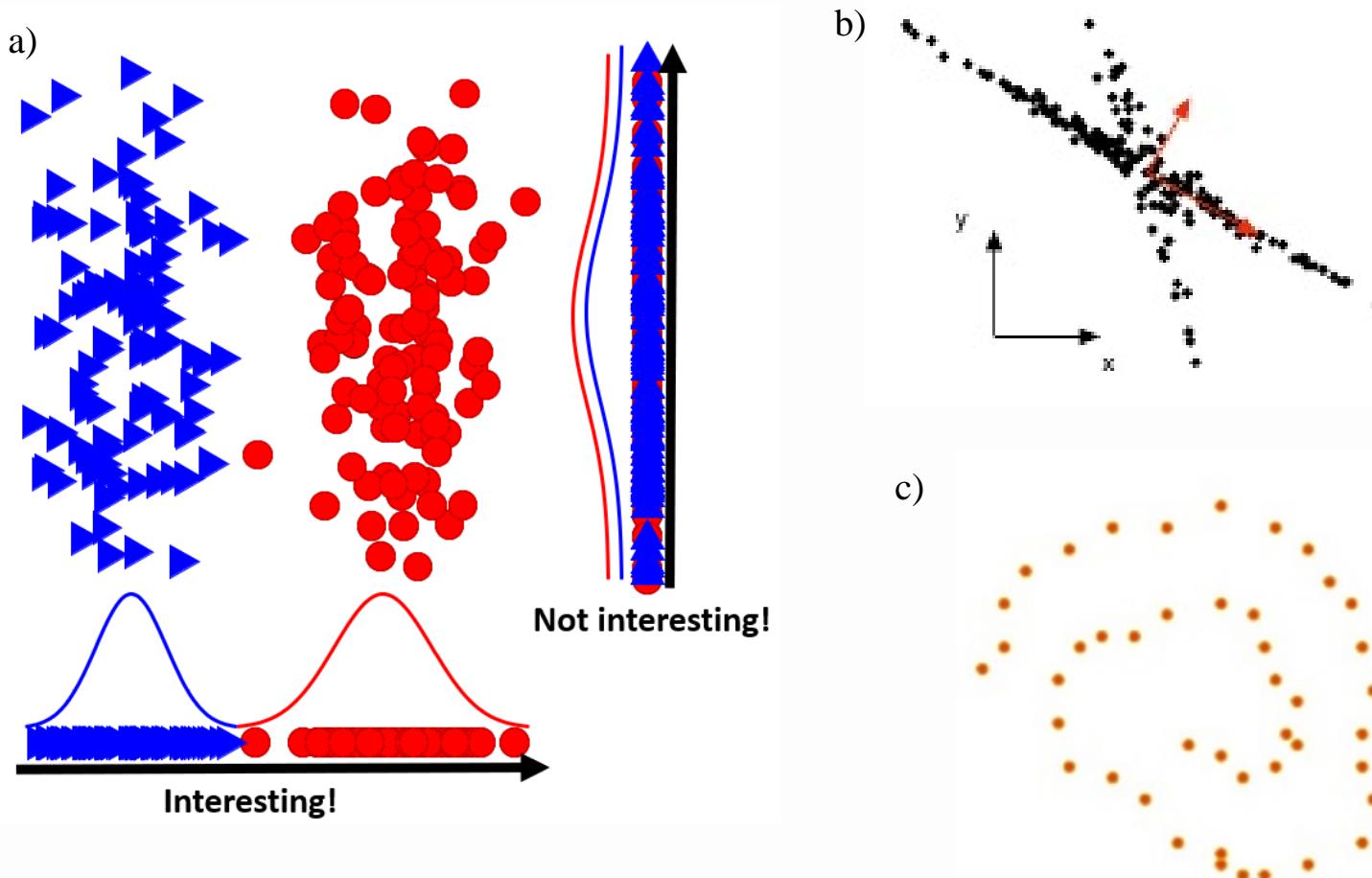


Image sources: <https://towardsdatascience.com/interesting-projections-where-pca-fails-fe64ddca73e6> and Shlen's good tutorial: <https://arxiv.org/abs/1404.1100>

Singular value decompositions (SVD)

Factorize \mathbf{D} as $\mathbf{D} = \mathbf{Q}\Sigma\mathbf{P}^T$ a

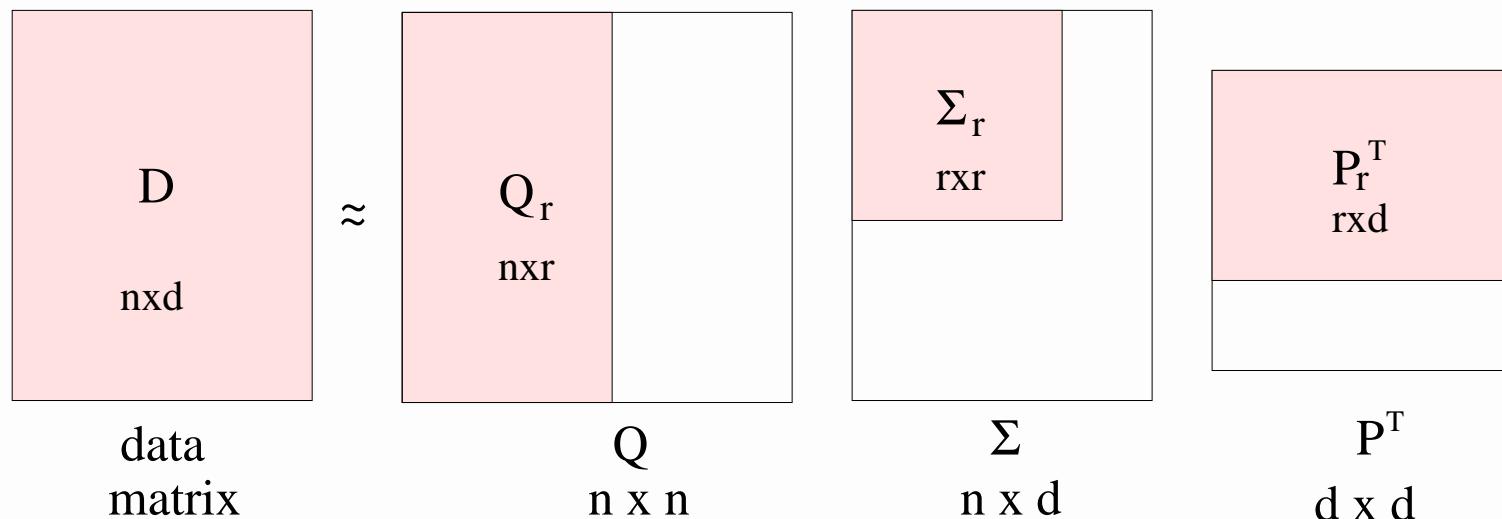
- Σ diagonal, $\Sigma_{ii} = \sigma_i$ **singular values**
- \mathbf{Q} 's columns **left singular vectors**, (orthonormal eigenvectors of \mathbf{DD}^T)
- \mathbf{P} 's columns **right singular vectors**, (orthonormal eigenvectors of $\mathbf{D}^T\mathbf{D}$)
- transformed data $\mathbf{D}' = \mathbf{DP}$
 - if \mathbf{D} mean-centered, same basis vectors as PCA b
 - mean-centering often skipped, if \mathbf{D} sparse, non-negative (e.g., document-word matrices)

^aalways possible

^bmay be opposite direction

Truncated SVD: Dimension reduction

- Assume Σ ordered into decreasing order by $\sigma_i = \Sigma_{ii}$.
- Keep only r largest $(\sigma_1, \dots, \sigma_r)$ + corresponding singular vectors of \mathbf{P}
- approximate data $\mathbf{D}' = \mathbf{D}\mathbf{P}_r$



Previous example with SVD

Let's first test without mean-centering:

```
Q
[[ -4.10936057e-01  1.74569815e-01  8.24528531e-01  2.52257347e-01 -2.39114763e-01  1.29454797e-16]
 [ -6.45804322e-01  3.14417109e-01 -5.61570935e-01  3.01160896e-01 -2.79318562e-01  1.25912076e-16]
 [ -2.31559546e-01  1.26698028e-01  3.39347806e-02 -9.93766673e-02  5.02626927e-01  8.16496581e-01]
 [ -5.62143219e-01 -2.03086484e-01  4.38362617e-02 -6.02554461e-01  3.33302743e-01 -4.08248290e-01]
 [ -9.90241264e-02 -4.56482541e-01 -2.40332995e-02 -4.03801126e-01 -6.71951112e-01  4.08248290e-01]
 [ -1.86112160e-01 -7.77813886e-01 -3.37638835e-02  5.56475872e-01  2.22626206e-01 -3.01457954e-16]]

Sigma
[8.42523943e+00 3.26119142e+00 9.87979172e-01 5.74286469e-01 2.72145612e-01 2.01264667e-17]

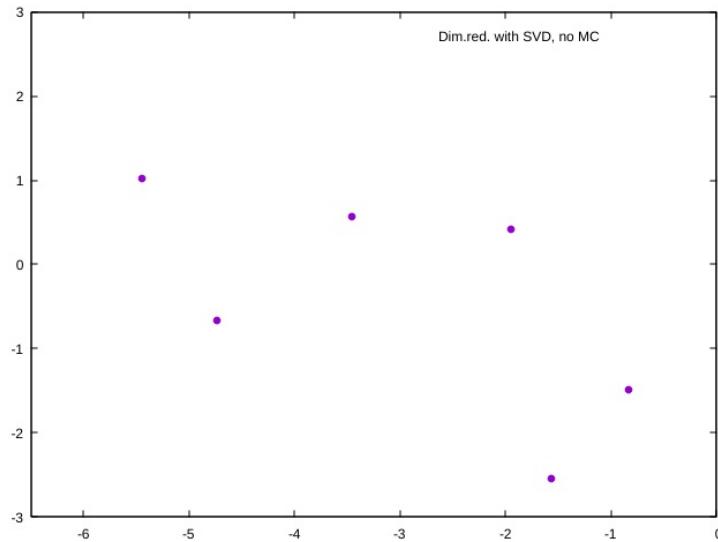
P^T
[[ -4.11777822e-01 -4.88428975e-01 -4.39654568e-01 -6.11083254e-01 -1.00564442e-01 -1.22654279e-01]
 [ 2.14185191e-01  3.10596922e-01  2.57067462e-01 -3.68663051e-01 -4.40753922e-01 -6.79259973e-01]
 [ 6.55400957e-01  8.69973391e-02 -7.47563302e-01  3.86918624e-02 -1.41307851e-02 -4.83054767e-02]
 [-3.44164653e-01  1.80244178e-01 -2.59009332e-01  3.65859132e-01 -7.83371609e-01  1.85614954e-01]
 [ 4.86378460e-01 -5.39978569e-01  3.38649460e-01 -1.48261638e-01 -4.26323839e-01  3.91716930e-01]
 [ 0.00000000e+00 -5.77350269e-01  1.66533454e-16  5.77350269e-01  9.12464548e-16 -5.77350269e-01]]
```

σ_1 and σ_2 largest $\Rightarrow \mathbf{P}_2 = \text{first two columns of } \mathbf{P}$
+ calculate $\mathbf{D}' = \mathbf{D}\mathbf{P}_2$

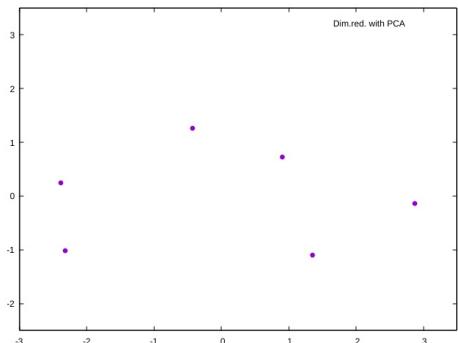
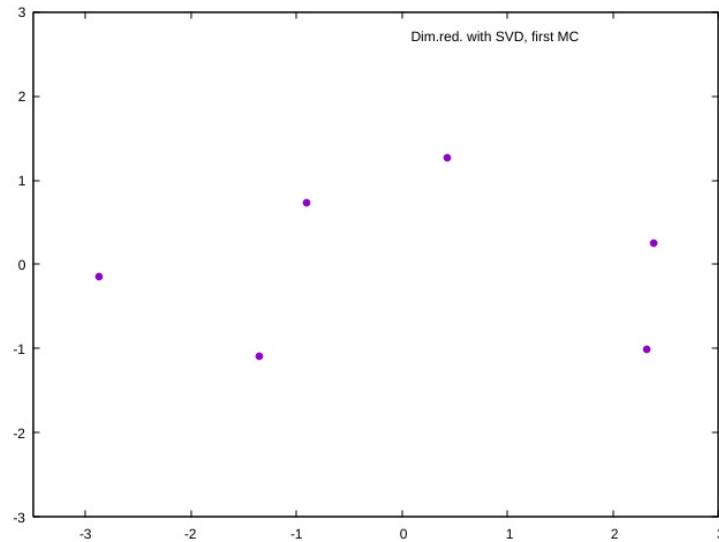
Example continued

Note: Different \mathbf{Q} , Σ , and \mathbf{P} , if mean-centered data

\mathbf{D}' without mean-centering



\mathbf{D}' with mean-centering



SVD applications

- dimension reduction: $\mathbf{D}' = \mathbf{D}\mathbf{P}_r$
- sometimes \mathbf{Q} also useful, e.g., user-item rating matrix
 - $\mathbf{D}^T\mathbf{Q}_r$ describes items by r latent components
- **Latent semantic analysis (LSA)** applies SVD on document-term matrix (e.g., tf-idf matrix)
 - often drastic dimension reduction!
 - helps with noise due to synonymous words
 - e.g., $\{(car), (truck), (flower)\} \rightarrow \{(1.3452 \cdot car + 0.2828 \cdot truck), (flower)\}$
- noise reduction: truncated SVD tends to correct inconsistencies

Example of truncated SVD (Aggarwal p. 45)

$$D = \begin{pmatrix} 2 & 2 & 1 & 2 & 0 & 0 \\ 2 & 3 & 3 & 3 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 2 & 2 & 2 & 3 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 2 & 1 & 2 \end{pmatrix} \approx Q_2 \Sigma_2 P_2^T$$
$$\approx \begin{pmatrix} -0.41 & 0.17 \\ -0.65 & 0.31 \\ -0.23 & 0.13 \\ -0.56 & -0.20 \\ -0.10 & -0.46 \\ -0.19 & -0.78 \end{pmatrix} \begin{pmatrix} 8.4 & 0 \\ 0 & 3.3 \end{pmatrix} \begin{pmatrix} -0.41 & -0.49 & -0.44 & -0.61 & -0.10 & -0.12 \\ 0.21 & 0.31 & 0.26 & -0.37 & -0.44 & -0.68 \end{pmatrix}$$
$$= \begin{pmatrix} 1.55 & 1.87 & \underline{1.67} & 1.91 & 0.10 & 0.04 \\ 2.46 & 2.98 & \underline{2.66} & 2.95 & 0.10 & -0.03 \\ 0.89 & 1.08 & 0.96 & 1.04 & 0.01 & -0.04 \\ 1.81 & 2.11 & 1.91 & 3.14 & 0.77 & 1.03 \\ 0.02 & -0.05 & -0.02 & 1.06 & 0.74 & 1.11 \\ 0.10 & -0.02 & 0.04 & 1.89 & 1.28 & 1.92 \end{pmatrix}$$

Summary

1. Factorize $\mathbf{C} = \mathbf{P}\Lambda\mathbf{P}^T$ or $\mathbf{D} = \mathbf{Q}\Sigma\mathbf{P}^T$
2. Use \mathbf{P}_r (best components of \mathbf{P}).
Reduced data $\mathbf{D}' = \mathbf{D}\mathbf{P}_r$

Only heuristics! Work well only if the underlying assumptions are true.

Hierarchical clustering: start

Watch video “Hierarchical Clustering - Fun and Easy Machine Learning” (10min) by Augmented Startups

<https://www.youtube.com/watch?v=EUQY3hL38cw>

(link in MyCourses)

Questions:

- Which linkage metric to use?
- What kinds of clusters can you find?
- Is there anything equivalent to large data?

Generic agglomerative hierarchical clustering

given D = intercluster distance (“linkage metric”)

Initialize distance matrix \mathbf{M}

Repeat until termination:

1. pick closest pair of clusters C_i and C_j ($D(C_i, C_j)$ minimal)
2. merge clusters: $C_{ij} = C_i \cup C_j$
3. update \mathbf{M}
 - remove rows and cols of C_i and C_j
 - add a new row and col for C_{ij} + their entries (distances to C_{ij})

Famous linkage metrics

Single	$\min_{\mathbf{x}_1 \in C_1, \mathbf{x}_2 \in C_2} \{d(\mathbf{x}_1, \mathbf{x}_2)\}$	elongated, straggly, also concentric clusters
Complete	$\max_{\mathbf{x}_1 \in C_1, \mathbf{x}_2 \in C_2} \{d(\mathbf{x}_1, \mathbf{x}_2)\}$	small, compact, hyperspherical, equal-sized
Average	$\frac{\sum_{\mathbf{x}_1 \in C_1, \mathbf{x}_2 \in C_2} d(\mathbf{x}_1, \mathbf{x}_2)}{ C_1 C_2 }$	quite compact; allows different sizes and densities
Minimum variance (Ward)	$SSE(C_1 \cup C_2) - SSE(C_1) - SSE(C_2)$	compact, quite well-separated, hyperspherical; not elongated or very different sized
Distance of centroids	$d(\mathbf{c}_1, \mathbf{c}_2)$	hyperspherical, equal-sized; not elongated

Famous linkage metrics

- linkage metric has a strong effect on results!
- **Warning:** most linkage metrics are sensitive to data order! \Rightarrow results may change if you shuffle data
- single linkage is not, but it is prone to “chaining effect”

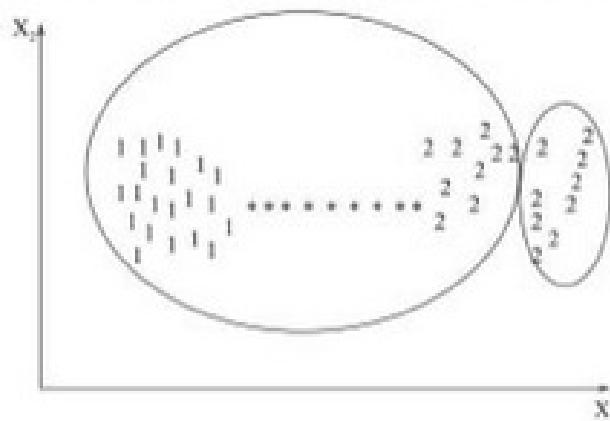


Figure 12. A single-link clustering of a pattern set containing two classes (1 and 2) connected by a chain of noisy patterns (*).

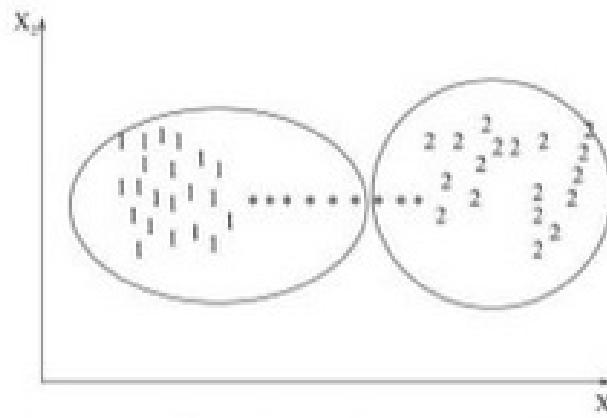


Figure 13. A complete-link clustering of a pattern set containing two classes (1 and 2) connected by a chain of noisy patterns (*).

image source <https://www.slideshare.net/KalpaGunaratna/incremental-conceptual-clustering-reading-group-discussion>

Example (Old Faithful data)

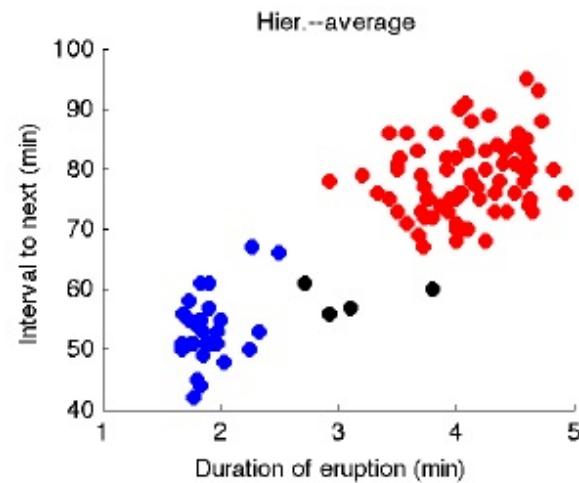
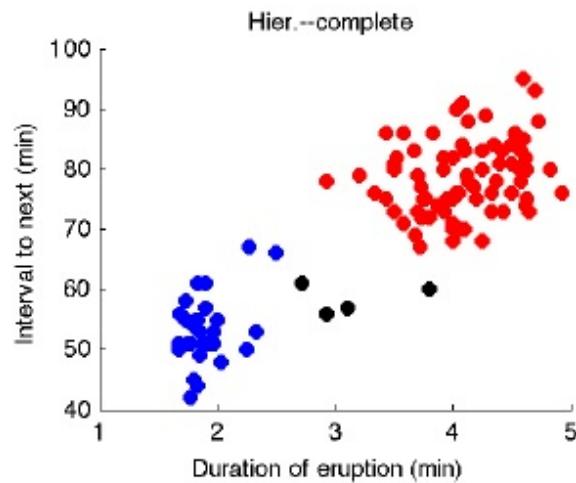
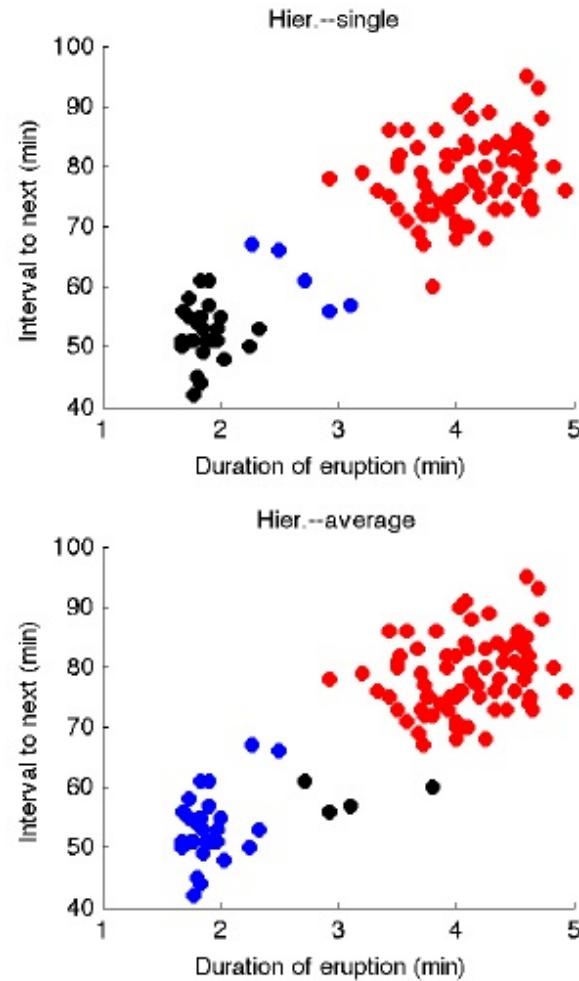
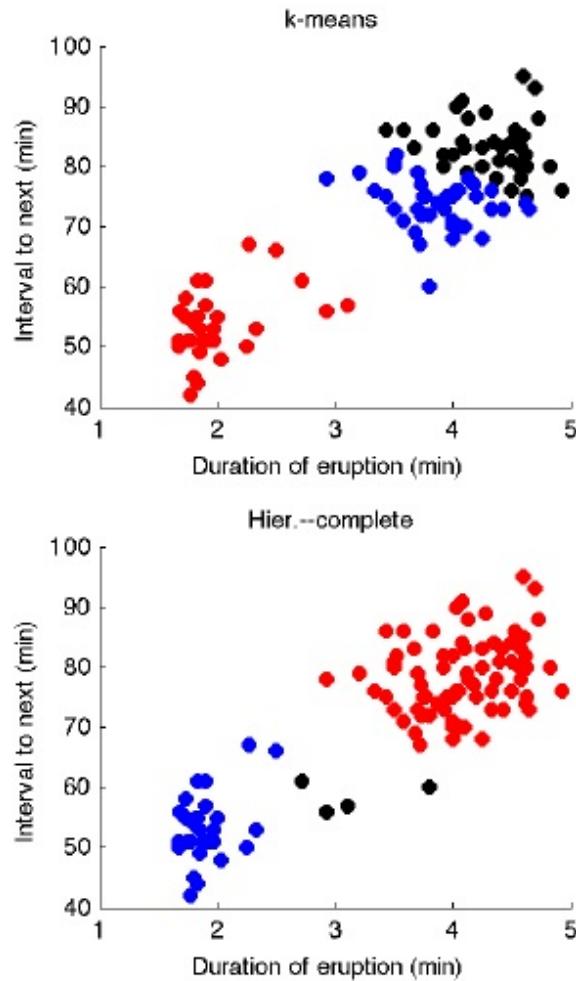


image source: Sungkyu Jung (2013) slides on clustering (STAT2221, Univ. of Pittsburgh)
https://www.stat.pitt.edu/sungkyu/course/2221Fall13/lec7_clustering.pdf

Connection to graph theory

Single linkage is related to **connected components** and complete linkage to **cliques**

Let e_1, \dots, e_m be edges of complete distance graph with weights $d_1 < d_2 < \dots < d_m$. ($m = \frac{n(n-1)}{2}$, n data points)

Single linkage

1. Initialize: Create graph \mathbf{G} without edges
 - i.e., n connected components and all data points in their own clusters
2. Repeat until one connected component
 - add new edge e_i with smallest d_i to \mathbf{G}
 - form clusters from connected components of \mathbf{G}

Connection to graph theory

Complete linkage

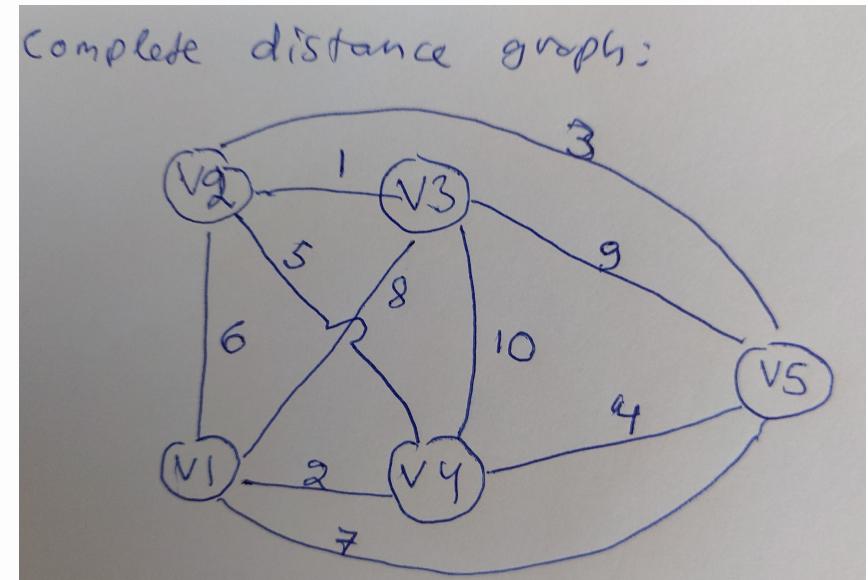
1. Initialize: Create graph \mathbf{G} without edges
 - all data points in their own clusters
2. Repeat until \mathbf{G} complete
 - add new edge e_i with smallest d_i to \mathbf{G}
 - if two of the current clusters form a clique in \mathbf{G} , merge them

Note: You are not allowed to break existing clusters, even if you would find alternative cliques

Task: graph-based single linkage clustering

Distance matrix:

	v1	v2	v3	v4	v5
v1	0	6	8	2	7
v2	6	0	1	5	3
v3	8	1	0	10	9
v4	2	5	10	0	4
v5	7	3	9	4	0



Add edges in the increasing order of weights (2, 3, ..., 7). What are the corresponding single linkage clusters? Complete linkage clusters? (hometask)

Task

**Extra: single linkage clustering from minimum spanning trees*

Begin from complete distance graph \mathbf{G} and search its minimum spanning tree (MST)

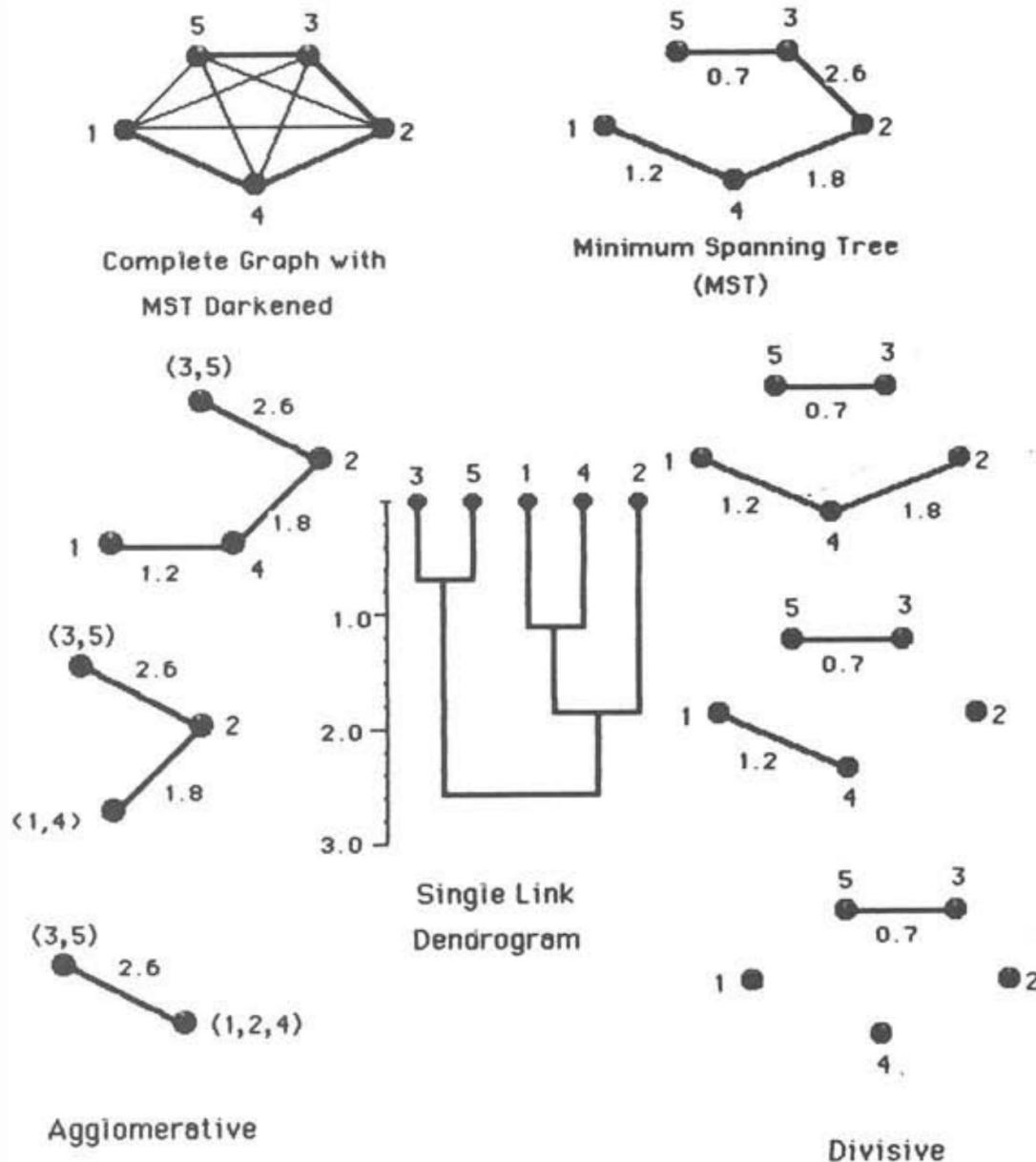
Repeat until all objects belong to one cluster:

1. Merge two clusters that are connected in the MST and have the smallest edge weight
2. Set the edge weight as ∞

Notes:

- The same can be done in a divisive manner: cut the MST edges in the descending order by weight.
- If there are no proximity ties, the result is the same as normal single linkage clustering

*Example (Jain and Dubes 1988, Fig 3.6)



Agglomerative or divisive?

- Agglomerative = bottom-up
 - cheaper and easier to implement
 - still slow, at least $O(n^2)$
 - early decisions based on local patterns, cannot cancel later
- Divisive = top-down
 - often better quality clustering
 \Leftarrow large clusters created early, based on global distribution
 - fastest $O(n^2 \log(n))$

Bisecting K-means

Idea: combine divisive hierarchical and K -means.

Given K and q =number of iterations

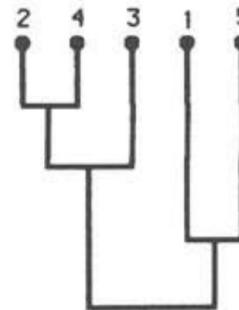
1. Initialization: put all data points into one cluster
 2. Repeat until K clusters:
 - choose cluster C to split (with largest SSE)
 - split C q times with 2-means
 - keep the best split (two new clusters)
- + efficient (like K -means)
- + good results (comparable to hierarchical)

On dendograms

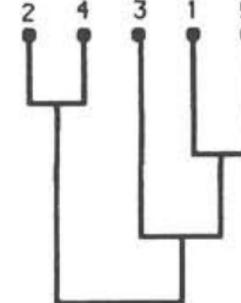
- **Threshold dendograms:** in which order the clusters were formed
- **Proximity dendograms:** at which proximities they were formed

	x_2	x_3	x_4	x_5
x_1	5.8	4.2	6.9	2.6
x_2		6.7	1.7	7.2
x_3			1.9	5.6
x_4				7.6

Threshold
Dendograms

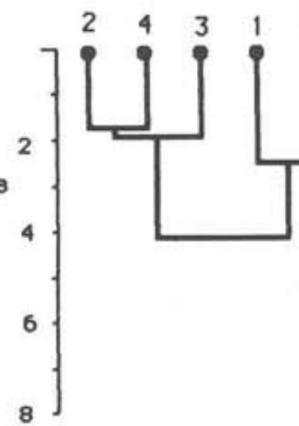


single

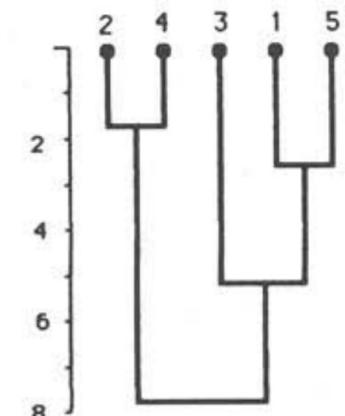


complete

Proximity
Dendograms



single



complete

image source Jain & Dubes 1988 Fig 3.5

Dendrogram example

Here real (biological) classes of data points are shown under the dendrogram

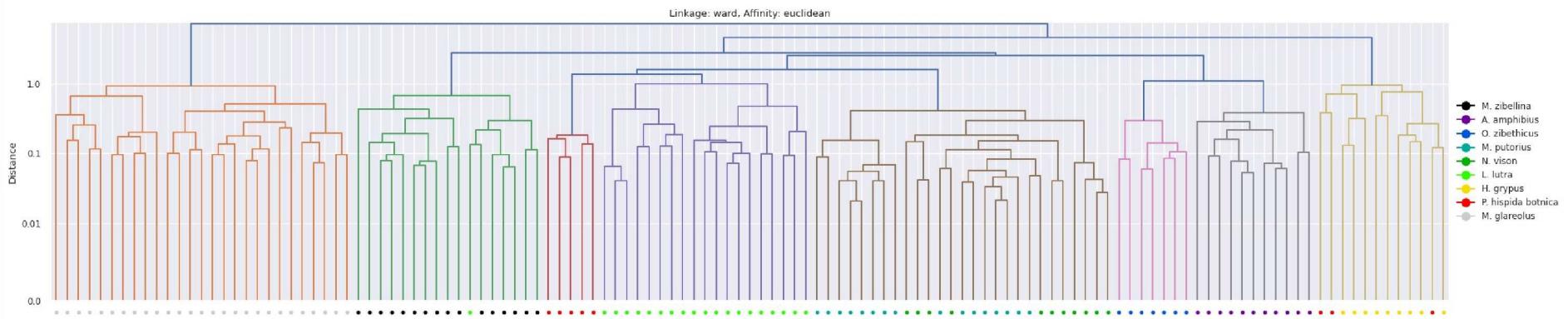


image author Lehtiniemi 2020

Another dendrogram example

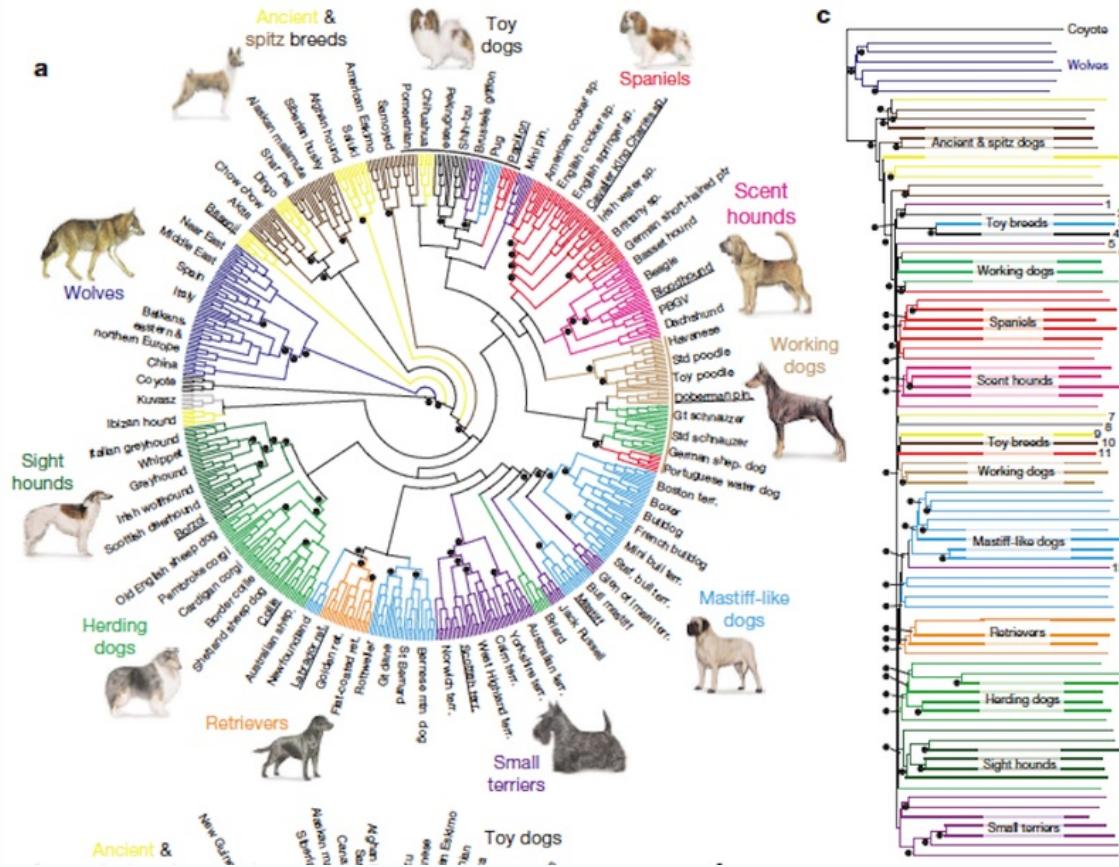


image source

<https://www.instituteofcaninebiology.org/how-to-read-a-dendrogram.html>

Summary

- useful information on clustering structure
 - dendograms!
- linkage metrics have a strong effect
 - Beware: most metrics sensitive to data order!
- connections to graph theory (single \leftrightarrow connected components, complete \leftrightarrow cliques)
- inefficient for really large data (at least $O(n^2)$ time and space)

Voluntary task: Fill a summary table!

method	data type	cluster type	benefits	drawbacks
<i>K-representatives</i>				
<i>K-means</i>				
<i>K-medoids</i>				
...				
<i>Hierarchical</i>				
<i>singe-link</i>				
...				
<i>Graph-based</i>				
<i>Density-based</i>				
<i>Probabilistic</i>				

Further reading

- Gan, Ma, Wu: Data clustering – theory, algorithms, and applications. SIAM 2007.
- Jain and Dubes: Algorithms for clustering data. Prentice-Hall 1988.

Today's lecture

1. K -representatives clustering

- Recap K -means (video)
- other members of the family

2. Hierarchical clustering

- introduction (video)
- more on linkage metrics, connections to graph theory, dendograms

Book 6.3, 6.4

Main groups of clustering methods (Aggarwal)

- Representative-based
- Hierarchical
- Probabilistic model-based
- Density-based (including grid-based)
- Graph-based
- Matrix factorization based

Representative-based: K-means

Watch video “K-means clustering: how it works” (7.5 min)
by Victor Lavrenko

https://www.youtube.com/watch?v=_aWzGGNrcic

Questions

- Why K -means is only for numerical data?
- Could we apply something similar to categorical data? or other data types?

K-means

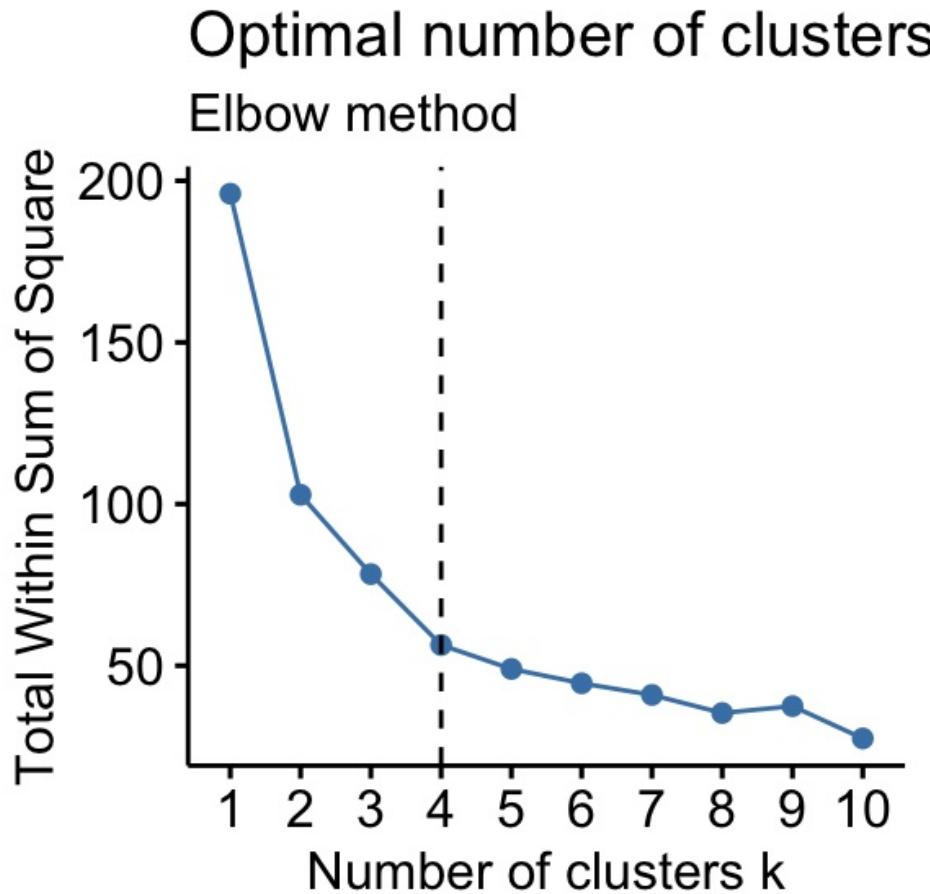
Notations: Data points $\mathbf{x}_i \in \mathcal{D}$, clusters C_1, \dots, C_K , centroids $\mathbf{c}_1, \dots, \mathbf{c}_k$, \mathbf{m} mean of data.

- objective: minimize $SSE = \sum_{j=1}^K \sum_{\mathbf{x} \in C_j} L_2^2(\mathbf{x}, \mathbf{c}_j)$
 - minimizes wc, maximizes bc, since
$$\sum_{\mathbf{x} \in \mathcal{D}} L_2^2(\mathbf{x}, \mathbf{m}) = \sum_{j=1}^K \sum_{\mathbf{x} \in C_j} L_2^2(\mathbf{x}, \mathbf{c}_j) + \sum_{j=1}^K |C_j| L_2^2(\mathbf{c}_j, \mathbf{m})$$
- tends to find **compact, hyperspherical** clusters
- **designed only for L_2** , but many K -representative variants for other distance measures
 - **warning:** if you use another distance in K -means, may not find even local optimum or converge. **Why?**
- very sensitive to the initialization of centroids!
→ **run multiple times**

K-means

- + can produce good results if clusters compact, well-separated, hyperspherical
- + easy to implement
- + quite efficient $O(nKq)$, q =number of iterations
- basic form requires L_2 measure
- sensitive to outliers
- sensitive to initialization (some improved strategies)
- converges to local optimum (not necessarily global)
- sometimes convergence can be slow
- needs parameter K

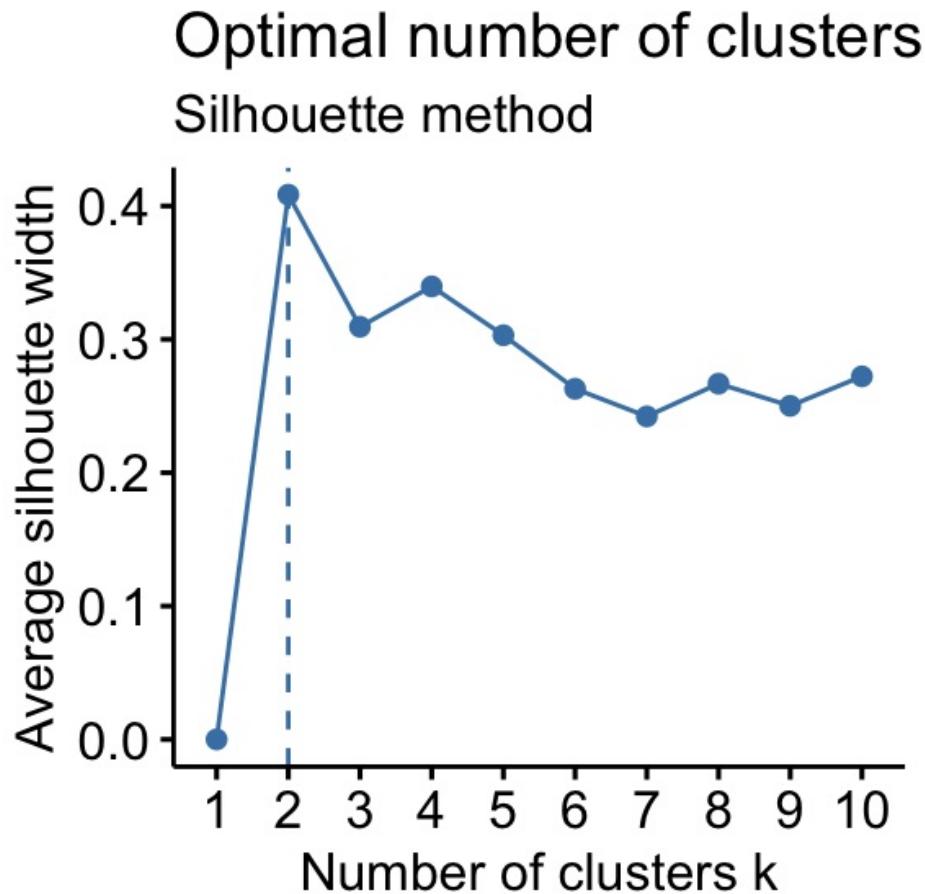
Choosing number of clusters: SSE elbow



- SSE decreases with K
- is there an elbow of the curve, where speed slows down?
- not always clear

source <https://www.datanovia.com/en/lessons/determining-the-optimal-number-of-clusters-3-must-know-methods/>

Choosing number of clusters: silhouette peak



- Silhouette tells how well an individual data point is clustered
- **Average silhouette** evaluates the entire clustering

source <https://www.datanovia.com/en/lessons/determining-the-optimal-number-of-clusters-3-must-know-methods/>

Silhouette coefficient

Silhouette of a point \mathbf{x} is

$$S(\mathbf{x}) = \begin{cases} 0 & \text{if singleton} \\ \frac{b-a}{\max\{a,b\}} & \text{otherwise} \end{cases}$$

a =mean distance of \mathbf{x} to points in the same cluster

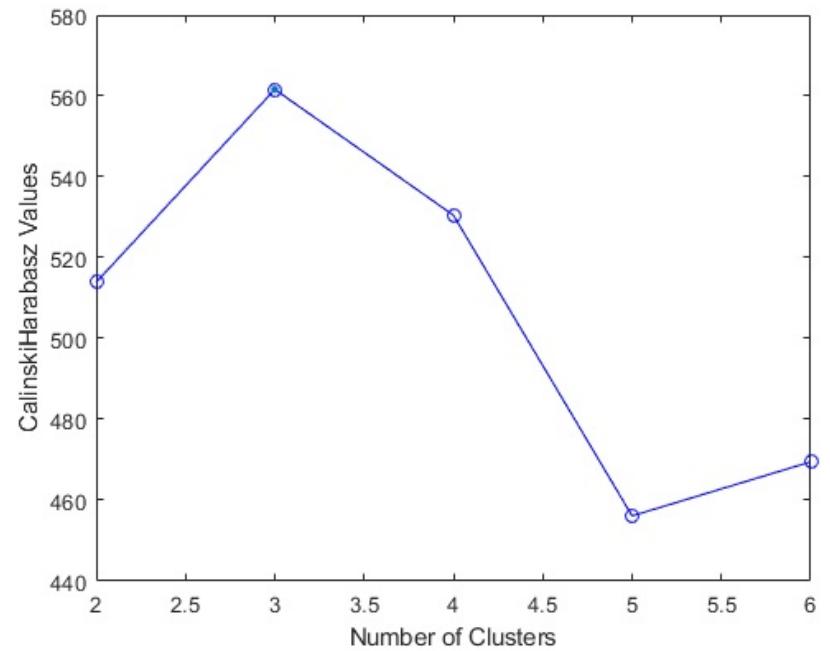
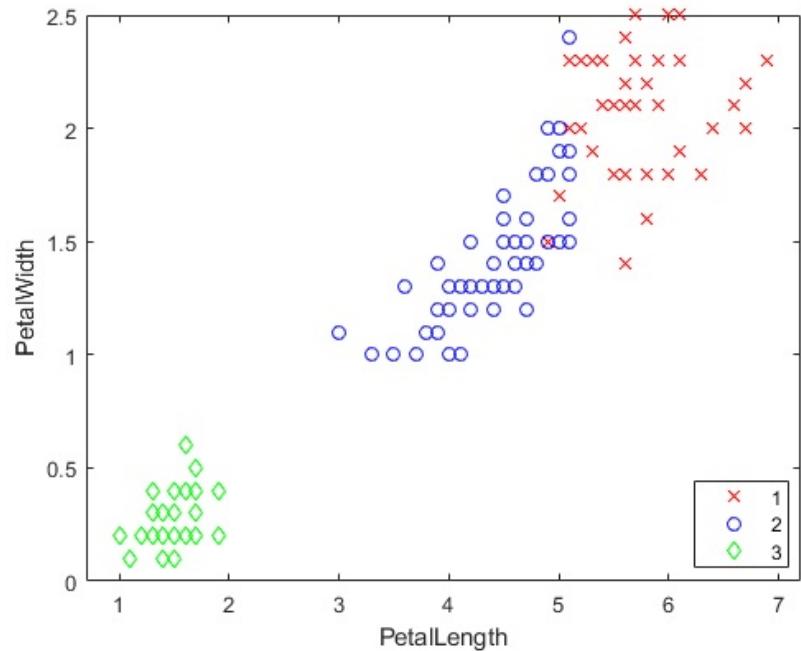
b =mean distance of \mathbf{x} to points in the closest neighbouring cluster

⇒ average Silhouette $S_{avg} = \frac{1}{n} \sum_{\mathbf{x} \in \mathcal{D}} S(\mathbf{x})$

→ More on lecture 5

Choosing number of clusters: Calinski-Harabasz

based on inter-cluster and intra-cluster variances



source <https://www.mathworks.com/help/stats/clustering.evaluation.calinskiharabaszevaluation-class.html>

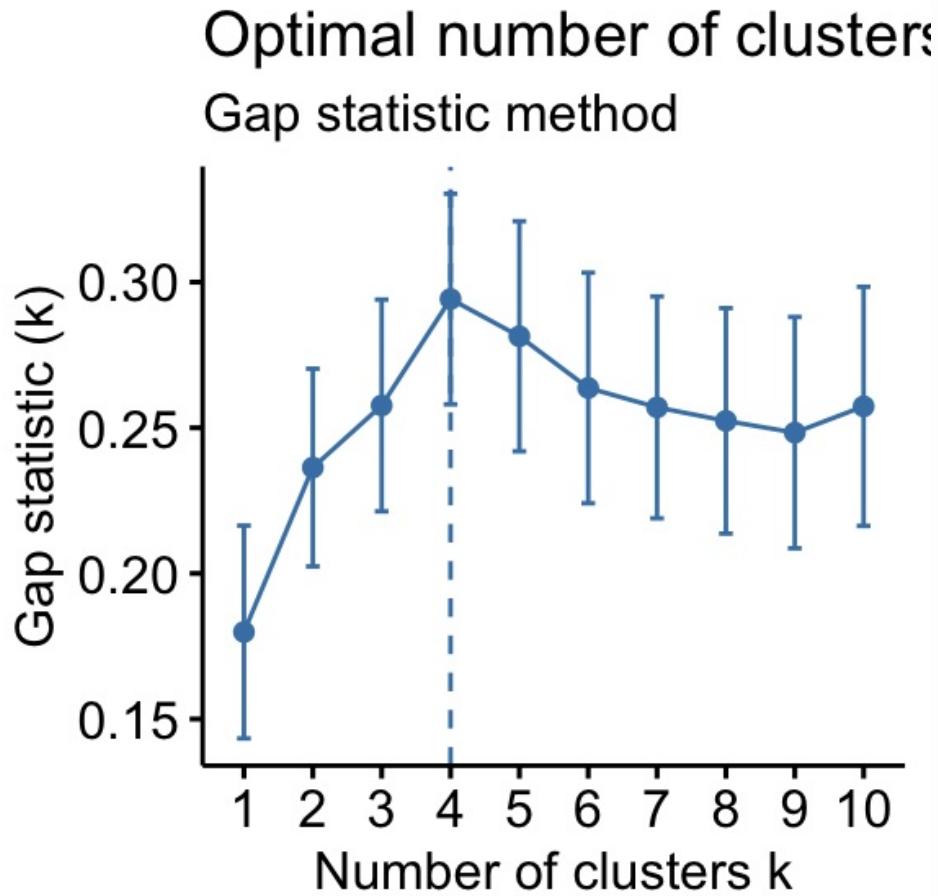
Choosing number of clusters: Calinski-Harabasz

$$S_{CH} = \frac{(n - K)B}{(K - 1)W}$$

- between-cluster variance $B = \sum_{i=1}^K |C_i| L_2^2(\mathbf{c}_i, \mathbf{m})$ (\mathbf{m} = mean of the whole data)
- within-cluster variance $W = \sum_{i=1}^K \sum_{\mathbf{x} \in C_i} L_2^2(\mathbf{x}, \mathbf{c}_i)$
- well suitable to K -means!

→ More on lecture 5

Choosing number of clusters: Gap statistic



- Cluster data and evaluate $W_K = \sum_{r=1}^K \frac{1}{2|C_r|} \sum_{\mathbf{x}, \mathbf{y} \in C_r} d(\mathbf{x}, \mathbf{y})$
- Evaluate W_K in B random data sets → W_{K1}, \dots, W_{KB}
- $Gap(K) = \frac{1}{B} \sum_{b=1}^B \log(W_{Kb}) - \log(W_K)$
- Choose $\min K$: $Gap(K) \geq Gap(K+1) - \sigma_{K+1}$

source <https://www.datanovia.com/en/lessons/determining-the-optimal-number-of-clusters-3-must-know-methods/>

Gap statistic

- σ_K = standard deviation of W_{K1}, \dots, W_{KB}
- if $d = L_2^2$, W_K estimates SSE
- + suits to any clustering method and distance d
- computationally heavy (B random simulations for all tested K)

Further reading: Tibshirani et al.: Estimating the number of clusters in a data set via the gap statistic. Journal of the Royal Statistical Society, 2001.

K-means extensions

- *K-medians*

- uses L_1 measure and medians
- determine median values along each dimension separately
- + more robust to outliers
- computationally more costly

- *K-medoids*

- medoid = the center-most **data point** in a cluster
- + more efficient (but slower than k -means)
- + allows any distance function
- + suits to any data type! (given distance function)

K-means vs. K-medoids

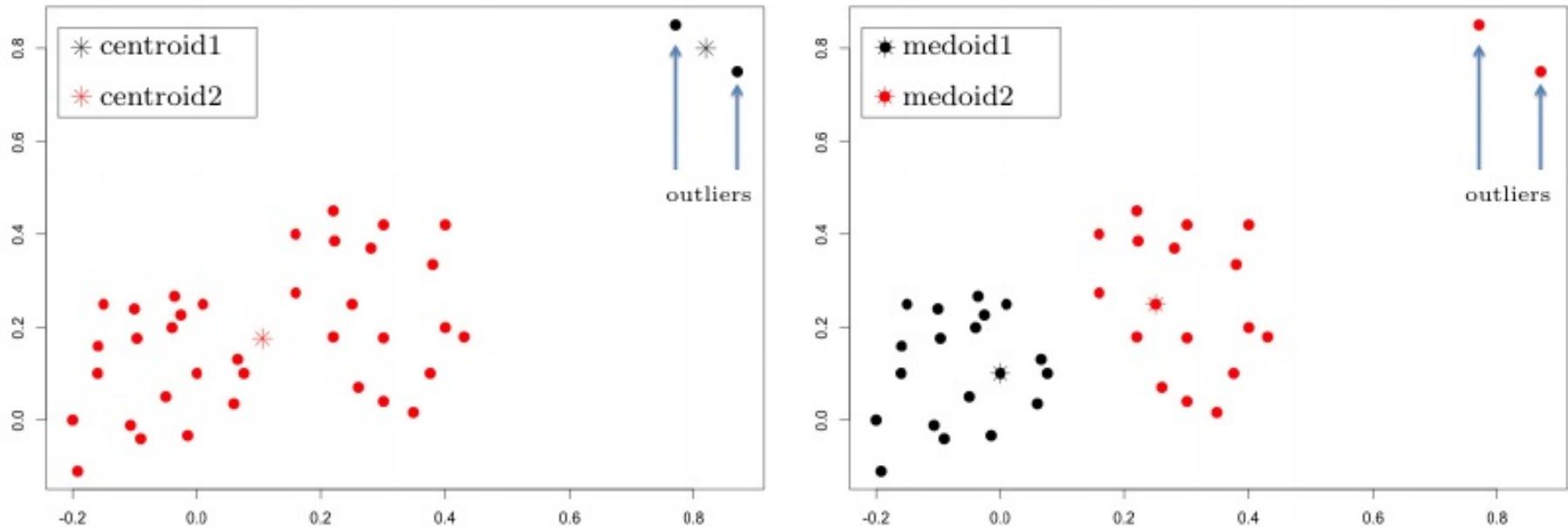


Figure 3.3: Outliers effect: *k*-means clustering (left) vs. *k*-medoids clustering (right)

image source: Soheily-Khah (2016): Generalized *k*-means based clustering for temporal data under time warp

K-modes

- for categorical data
- minimize $\sum_{\mathbf{x} \in C} \sum_{i=1}^k d_s(x_i, c_i)$, where

$$d_s(x_i, y_i) = \begin{cases} 1 & \text{if } x_i \neq y_i \\ 0 & \text{otherwise} \end{cases}$$

- “simple matching distance” = overlap distance without weights
- cluster centers \mathbf{c} are “modes” (choose most frequent values of each feature)

K-modes: example

K=3. Original centers ("modes") individuals 1, 5, 10

Individual	Q1	Q2	Q3	Q4	Q5	C1	C2	C3
1	A	B	A	B	C	0	4	2
2	A	A	A	B	B	2	4	4
3	C	A	B	B	A	4	2	4
4	A	B	B	A	C	2	5	0
5	C	C	C	B	A	4	0	5
6	A	A	A	A	B	3	5	4
7	A	C	A	C	C	2	4	3 ↗
8	C	A	B	B	C	3	3	3
9	A	A	B	C	A	4	4	3
10	A	B	B	A	C	2	5	0

Note: Many ways to choose initial “modes”.

K-modes: example

Calculate new modes:

Cluster	Q1	Q2	Q3	Q4	Q5
1 (1), (2), (6), (7), (8)	A	A	A	B	C
2 (3), (5)	C	A	B	B	A
3 (4), (9), (10)	A	B	B	A	C 

Example from "K-Modes intuition and example" by Aysan Fernandes

https://www.youtube.com/watch?v=b39_vipRkUo

K-prototypes

- for mixed data

- minimize

$$\sum_{\mathbf{x} \in C} \left(\sum_{i=1}^q (x_i - c_i)^2 + \gamma \sum_{i=q+1}^k d_s(x_i, c_i) \right), \text{ where}$$

x_1, \dots, x_q numerical values

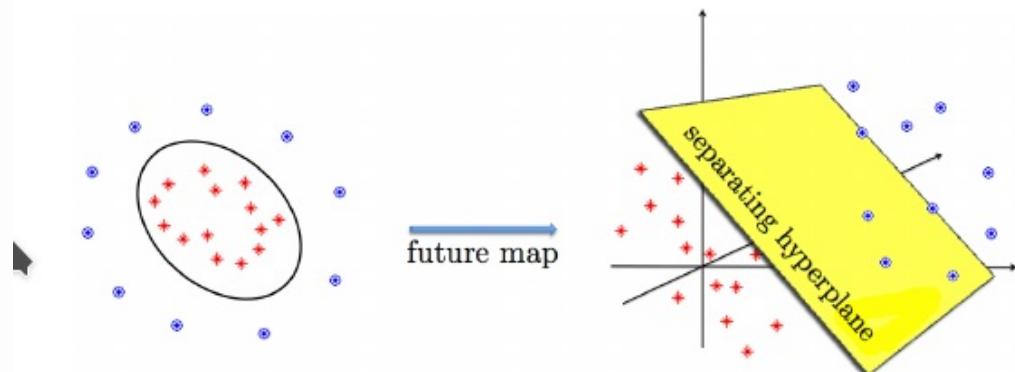
x_{q+1}, \dots, x_k categorical values

γ =balancing weight

- cluster centroids \mathbf{c} are “prototypes”

K-means extensions: Kernel-K-means

Idea: map data implicitly to a higher dimensional space and perform K -means there



The kernel trick - complex in low dimension (left), simple in higher dimension (right)

- + robust
- + can detect arbitrary shapes
- expensive

image source Soheily-Khah (2016): Generalized k-means based clustering for temporal data under time warp

Summary

- Basic idea of K -representatives method
 - K -means, K -medians, K -medoids, K -modes, K -prototypes
- Techniques to choose K
 - SSE elbow, Silhouette peak, Calinski-Harabasz, Gap statistic

Further reading:

- Gan, Ma, Wu: Data clustering – theory, algorithms, and applications. SIAM 2007.
- Jain and Dubes: Algorithms for clustering data. Prentice-Hall 1988.

Clustering validation

Is there any real clustering? How good is it?

- Book: Chapter 6.9
- External material: Halkidi et al. (2002): Cluster Validity Methods: Part I. ACM SIGMOD Record 31(2): 40–45.

<https://doi.org/10.1145/565117.565124>

Three similar problems

1. Clustering tendency: is there any clustering in data presented with certain features?
2. Determining number of clusters (or other parameters)
3. Evaluating goodness of clustering
 - compare different methods
 - compare against classification

All three depend on the **clustering objective!**

- assumptions on clusters (e.g., compactness, shape)
- separation between clusters

Evaluating goodness of clustering

1. Internal criteria

- validity indices, similar to objective functions
- do not work, if clustering had a different objective!
- can be used to i) evaluate a single clustering or ii) compare clusterings (as **relative indices**)

2. External criteria

- compare clustering to a predefined classification
- classes may not reflect natural clusters

3. Statistical hypothesis testing

- maybe the most sound approach, but computationally demanding

Internal validity indices

- indices assume some clustering objective → reward methods with the same objective
 - even a good clustering can get a bad score if a different objective!
 - many indices assume/favor spherical or convex clusters
- best for comparing similar algorithms and tuning parameters
- Some popular indices:
 - **Average silhouette**
 - **Calinski-Harabasz index**
 - **Davies-Bouldin index**

Silhouette index

Silhouette of a point \mathbf{x} is

$$S(\mathbf{x}) = \begin{cases} 0 & \text{if } \mathbf{x} \text{ a cluster of its own} \\ \frac{b-a}{\max\{a,b\}} & \text{otherwise} \end{cases}$$

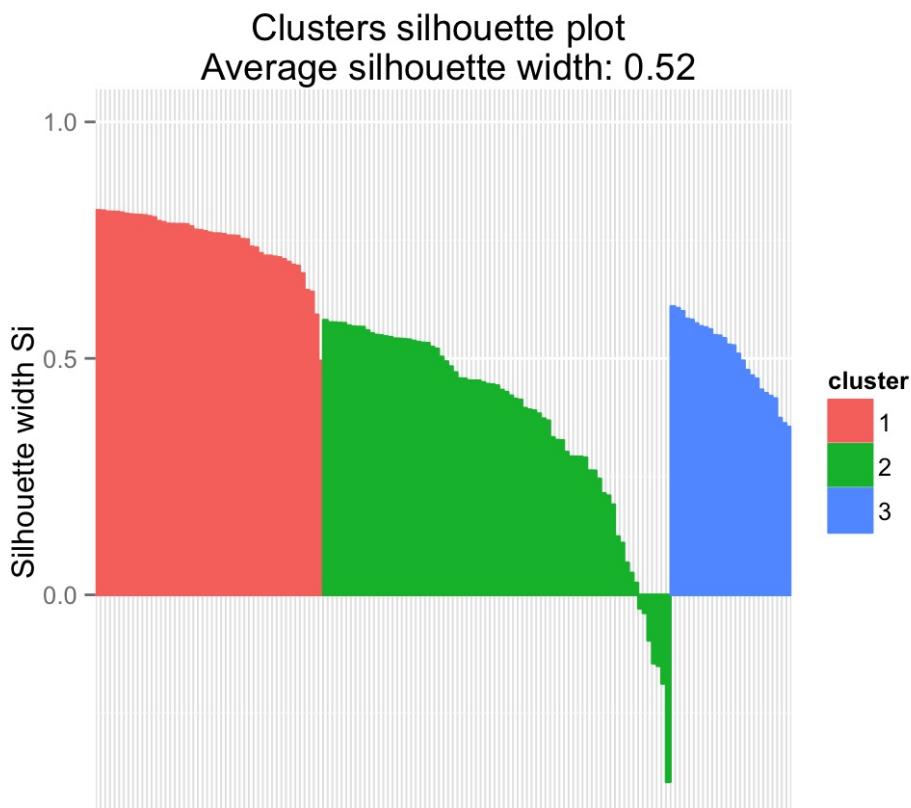
$$a = \text{avg}\{d(\mathbf{x}, \mathbf{y}) \mid \mathbf{x} \in C, \mathbf{y} \in C\}$$

$$b = \min_q \text{avg}\{d(\mathbf{x}, \mathbf{y}) \mid \mathbf{x} \in C, \mathbf{y} \in C_q, C \neq C_q\}$$

≈ how closely \mathbf{x} matches its own cluster and how loosely the neighbouring cluster

- $S(\mathbf{x}) \in [-1, 1]$, **high values good**
- **Average silhouette** describes goodness of entire clustering
- flexible: any distance function d

Example: Silhouette of points



What negative values mean?

$$S(\mathbf{x}) = \begin{cases} 0 & \text{if singleton} \\ \frac{b-a}{\max\{a,b\}} & \text{otherwise} \end{cases}$$

$$a = \text{avg}\{d(\mathbf{x}, \mathbf{y}) \mid \mathbf{x} \in C, \mathbf{y} \in C\}$$

$$b = \min_q \text{avg}\{d(\mathbf{x}, \mathbf{y}) \mid \mathbf{x} \in C, \mathbf{y} \in C_q, C \neq C_q\}$$

image source [http://www.sthda.com/
english/wiki/wiki.php?id_contents=7952](http://www.sthda.com/english/wiki/wiki.php?id_contents=7952)

Calinski-Harabasz index

$$S_{CH} = \frac{(n - K)B}{(K - 1)W}$$

- **between-cluster variance** $B = \sum_{i=1}^K |C_i| L_2^2(\mathbf{c}_i, \mathbf{m})$, where \mathbf{m} is the mean of the whole data
- **within-cluster variance** $W = \sum_{i=1}^K \sum_{\mathbf{x} \in C_i} L_2^2(\mathbf{x}, \mathbf{c}_i)$
- requires $K \geq 2$
- range $[0, \infty[$, **high values good**
- When could you get value 0?

Calinski-Harabasz index (cont'd)

$$S_{CH} = \frac{(n - K)B}{(K - 1)W} = \frac{(n - K) \sum_{i=1}^K |C_i| L_2^2(\mathbf{c}_i, \mathbf{m})}{(K - 1) \sum_{i=1}^K \sum_{\mathbf{x} \in C_i} L_2^2(\mathbf{x}, \mathbf{c}_i)}$$

Note: $W = SSE(\mathbf{C})$. K -means criterion minimizes $W \Rightarrow$ maximizes B , because

$$\sum_{\mathbf{x} \in \mathcal{D}} L_2^2(\mathbf{x}, \mathbf{m}) = \sum_{i=1}^K \sum_{\mathbf{x} \in C_i} L_2^2(\mathbf{x}, \mathbf{c}_i)^2 + \sum_{i=1}^K |C_i| L_2^2(\mathbf{c}_i, \mathbf{m})$$

$\Rightarrow S_{CH}$ favours especially K -means!

Important: need to use L_2 in clustering!

Davies-Bouldin index

$$S_{DB} = \frac{1}{K} \sum_{i=1}^K \max_{j \neq i} \frac{S_i + S_j}{D_{ij}} \quad , \text{ where}$$

- $S_i = \left(\frac{1}{|C_i|} \sum_{\mathbf{x} \in C_i} L_p^q(\mathbf{x}, \mathbf{c}_i) \right)^{\frac{1}{q}}$ measures dispersion of C_i
 - usually $q = 2$ (stdev of distances)
 - if $q = 1$, average distances
- $D_{ij} = L_p(\mathbf{c}_i, \mathbf{c}_j)$ measures separation between C_i and C_j
- max: for each C_i , evaluate relation to most problematic C_j
- possible to take avg instead of max

Important: use the same L_p as the clustering algorithm!

Davies-Bouldin index (cont'd)

$$S_{DB} = \frac{1}{K} \sum_{i=1}^K \max_{j \neq i} \frac{S_i + S_j}{D_{ij}} \quad , \text{ where}$$

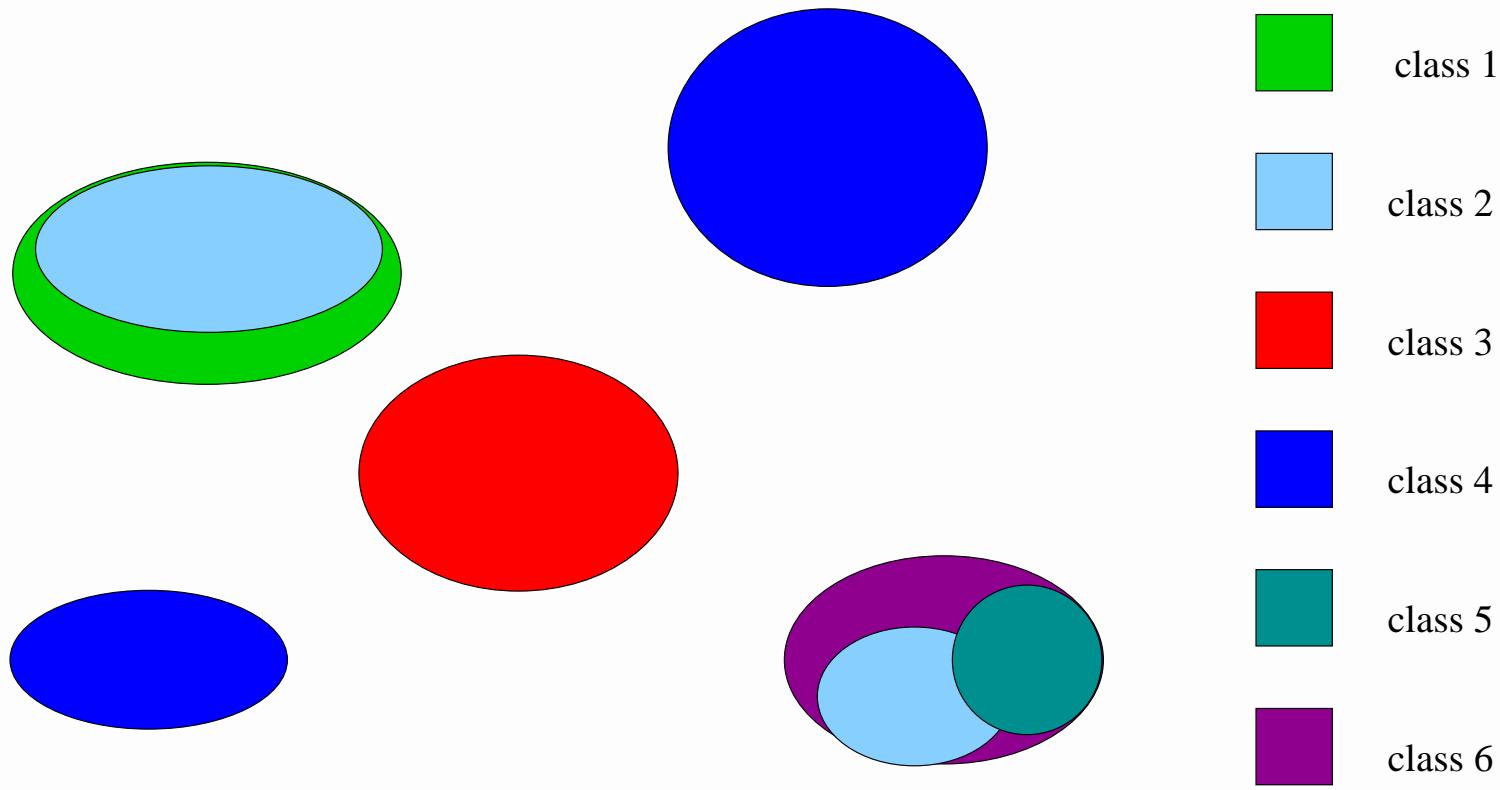
$$S_i = \left(\frac{1}{|C_i|} \sum_{\mathbf{x} \in C_i} L_p^q(\mathbf{x}, \mathbf{c}_i) \right)^{\frac{1}{q}} \text{ and } D_{ij} = L_p(\mathbf{c}_i, \mathbf{c}_j)$$

- range $[0, \infty[$, **small values good**
- When could you get value 0?

Possible strategies when S_{DB} used to determine K :

- restrict number of singletons (e.g., 0 or a few)
- define $S_i = a$ for some large a , when $|C_i| = 1$

External validation: Compare clustering against predefined classification



A confusion matrix: clustering vs. classification

	Class 1	Class 2	Class 3	
Cluster 1	n_{11}	n_{12}	n_{13}	m_1
Cluster 2	n_{21}	n_{22}	n_{23}	m_2
Cluster 3	n_{31}	n_{32}	n_{33}	m_3
	c_1	c_2	c_3	n

image source Cunningham <https://slideplayer.com/slide/14318989/>

External validation

Given clustering C_1, \dots, C_K and classification D_1, \dots, D_q .
Many validation indices! E.g.,

- **purity**

$$Pur(C) = \frac{1}{n} \sum_{i=1}^K \max_j |C_i \cap D_j|$$

- be careful! (increases with K)
- **normalized mutual information** NMI (robust, independent of K)
- **Rand index**

Normalized mutual information

Normalized mutual information by Strehl and Ghosh (2003):

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

mutual information $I = \sum_{C_i \in C} \sum_{D_j \in D} P(C_i, D_j) \log \frac{P(C_i, D_j)}{P(C_i)P(D_j)}$

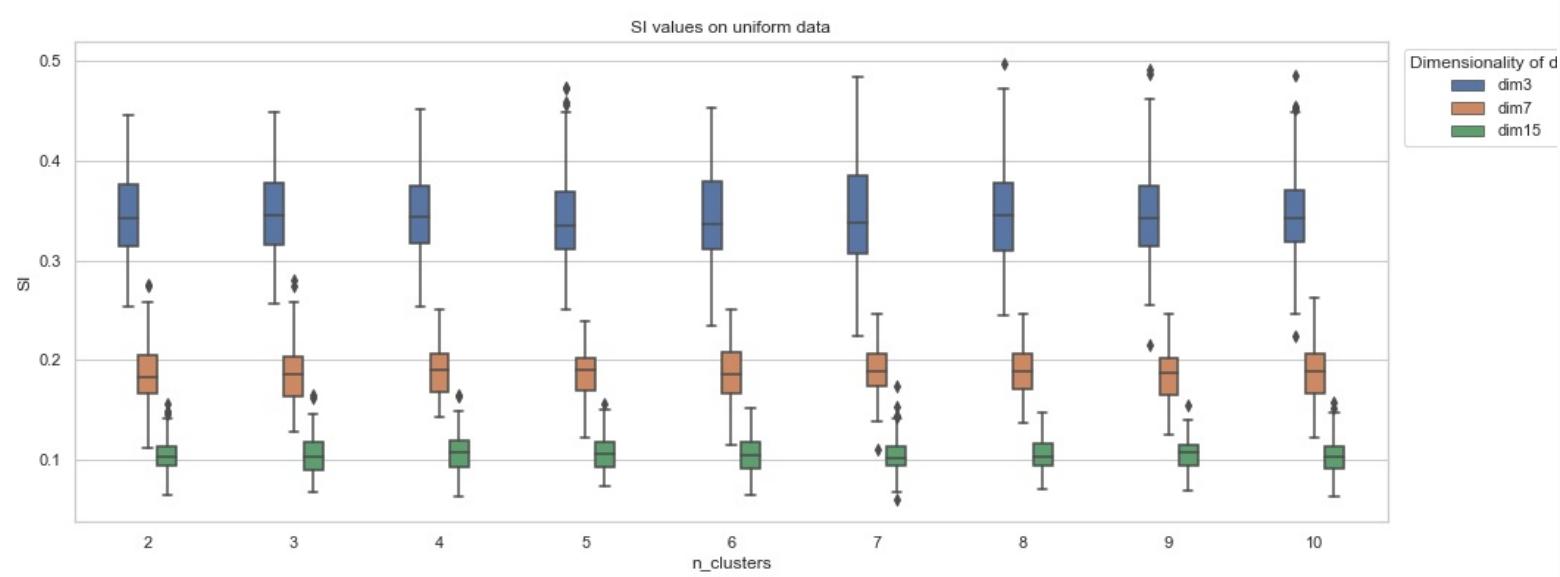
entropy $H(C) = -\sum_{C_i \in C} P(C_i) \log P(C_i)$

- + does not depend on the number of clusters
- many singleton clusters can cause problems

Note: Also other variants of normalized mutual information, give always equation and/or reference what you use!

Statistical hypothesis testing: motivation

SI can be pretty good even for random data!



- each feature generated independently from uniform distribution
- 100 randomizations
- K -means repeated 100 times → best result for each K

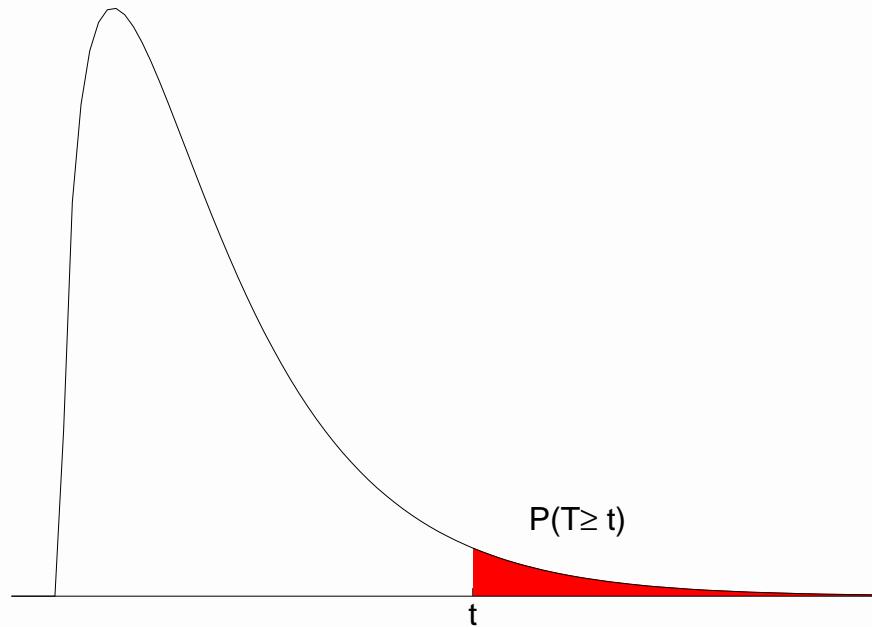
Experiment by Georgy Ananov for MDM 2023

Statistical hypothesis testing

Procedure:

1. decide a **null hypothesis** H_0 to test
 - describes the state where there isn't any clustering
 - e.g., H_0 : All sets of n locations in certain region are equally likely.
2. decide a **test statistic** T
 - may be a validity index
3. What is the probability to obtain at least as good test statistic values as in data (where $T = t$) if H_0 was true?

Statistical hypothesis testing



Assume that large T value good

Idea: If $P(T \geq t)$ very small
⇒ unlikely that the observed clustering had occurred by chance

- $P(T \geq t)$ is the **p-value** that can be used as a significance measure

Statistical hypothesis testing

Problem: How to evaluate p -value? (T 's distribution seldom known!)

- often by Monte Carlo experiments (randomization tests):
 - generate random data sets fulfilling H_0 , cluster them and evaluate T
 - p -value \approx proportion of random sets that obtained $T \geq t$ (if large T good)
- computationally demanding (a lot of simulations!)
- many alternatives for H_0 s and T s

Other evaluation: What the clustering reveals?

- Look at cluster sizes (e.g., C_1 : $n - 2$ data points and C_2 : 2 points – likely outliers!)
- How do the clusters differ? (selected and external features)
 - e.g., rats clustered by body measurements (weight, tail and body length, organ weights)
 - 2 clusters: big and small rats
 - vs. 3 clusters: C_1 : young or sick rats, C_2 : pregnant or nursing females, C_3 : other adults
- Are all clusters clear? (e.g., C_1 and C_3 intermingled, C_2 separate)

Summary

- Remember validation, but be cautious!
 - even random data can produce clusterings, but they seldom pass validation
 - problem: indices biased or do not reflect the underlying clustering
 - try always more than one validation technique
- Objective, distance measure, clustering method and validation should match!

Sources and further reading

- Halkidi et al. (2001): On clustering validation techniques, Journal of Intelligent Information Systems 17: 107–145. https://www.researchgate.net/publication/2500099_On_Clustering_Validation_Techniques
- Jain and Dubes (1988): Algorithms for clustering data, Ch 4.
- Gan, Ma, Wu (2007): Data clustering - theory, algorithms, and applications, Ch 17, https://www.researchgate.net/publication/220694937_Data_Clustering_Theory_Algorithms_and_Applications

Sources and further reading

- Vargha, Bergman, Takacs: Performing Cluster Analysis Within a Person-Oriented Context: Some Methods for Evaluating the Quality of Cluster Solutions. *Journal of Person-Oriented Research*, 2: 78-86, 2016.

Spectral clustering

Contents:

- Matrices from the similarity graph
- 1D spectral embedding & clustering
- Unnormalized and normalized spectral clustering
- Important choices

Book: Sections 2.4.4.3, 6.7, 19.3.4

Recommended external material:

von Luxburg (2007): A Tutorial on Spectral Clustering.

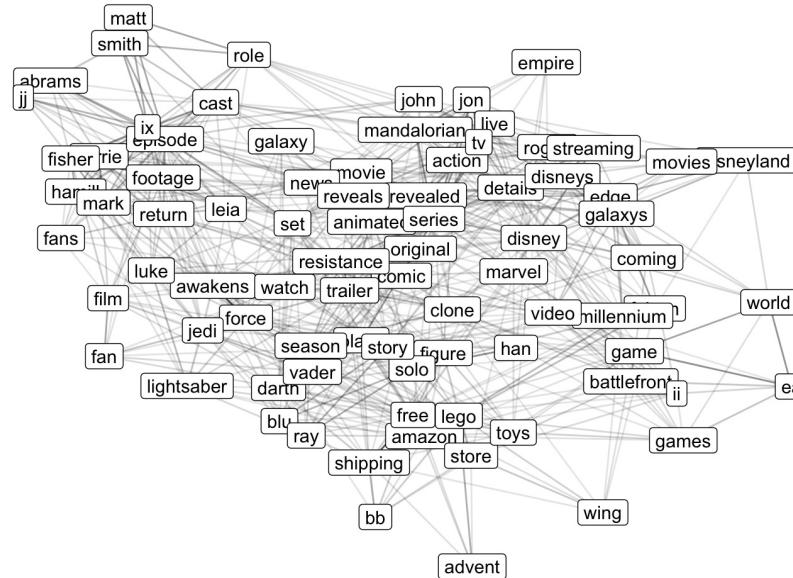
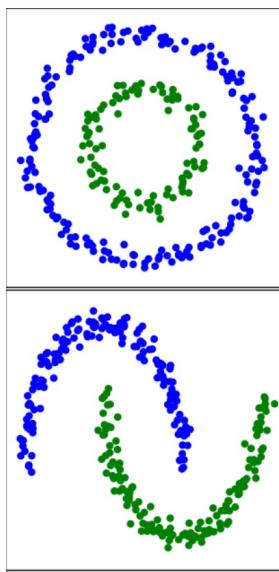
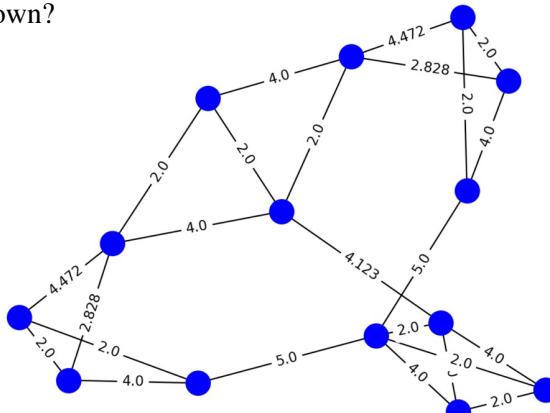
Presemo: <https://presemo.aalto.fi/mdm2023>

Recap: How could you cluster these?

Arbitrary shapes?



Only similarity graphs known?



Images: White (2019) <https://www.markhw.com/blog/word-similarity-graphs>, Cooper (2021) <https://spin.atomicobject.com/2021/09/07/spectral-clustering/>, Park & Kim (2020) <https://doi.org/10.1115/DETC2020-22642>, Scikit-learn documentation https://ogrisel.github.io/scikit-learn.org/sklearn-tutorial/auto_examples/cluster/plot_cluster_comparison.html

General idea of graph-based clustering

1. Present data as a similarity (neighbourhood) graph \mathbf{G}
 2. Cluster nodes of G with a network clustering or community detection algorithm
- + can detect arbitrary-shaped clusters
 - + even varying cluster densities (given k nearest neighbour similarity graph)
 - + for any data type (if pairwise similarity/distance defined)
 - computationally costly
 - many parameter choices

Spectral clustering: Idea

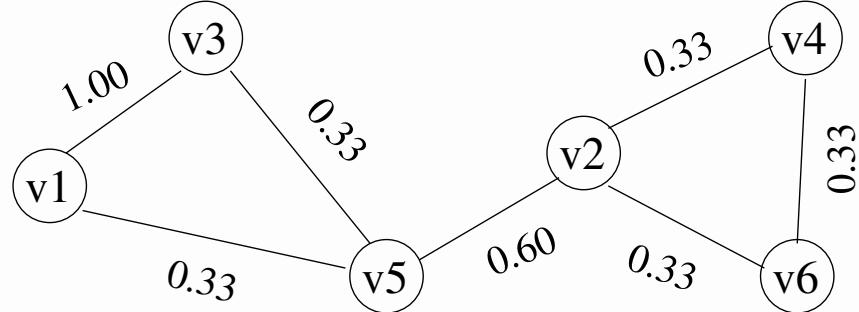
1. Create similarity graph \mathbf{G}
 - node v_i for the i th data point ($i = 1, \dots, n$)
 - edge weight w_{ij} = similarity between nodes v_i and v_j
2. Present data in (low-dimensional) vector space (i.e., find vectors $\mathbf{y}_1, \dots, \mathbf{y}_n$) such that local similarity/clustering structure is preserved
 - idea: choose \mathbf{Y} to minimize $cost(\mathbf{G}, \mathbf{Y}) = \sum \sum w_{ij} L_2^2(\mathbf{y}_i, \mathbf{y}_j)$
 - intuition: large w_{ij} tends to produce small $d(\mathbf{y}_i, \mathbf{y}_j)$
 - → easy after reformulation with a Laplacian matrix
3. Cluster \mathbf{y}_i s with K -means (etc.)

What is needed?

From \mathbf{G} derive:

1. weight matrix \mathbf{W}
2. diagonal degree matrix Λ
3. Laplacian matrix $\mathbf{L} = \Lambda - \mathbf{W}$
4. normalized Laplacian matrices \mathbf{L}_{rw} , \mathbf{L}_{sym} if desired)

Similarity graph and weight matrix \mathbf{W}



$$\begin{bmatrix} 0.00 & 0.00 & 1.00 & 0.00 & 0.33 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.33 & 0.60 & 0.33 \\ 1.00 & 0.00 & 0.00 & 0.00 & 0.33 & 0.00 \\ 0.00 & 0.33 & 0.00 & 0.00 & 0.00 & 0.33 \\ 0.33 & 0.60 & 0.33 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.33 & 0.00 & 0.33 & 0.00 & 0.00 \end{bmatrix}$$

- \mathbf{W} adjacency matrix of a weighted graph
- $W_{ij} = w_{ij}$ (similarity between nodes v_i and v_j)
- if unweighted graph, use weights 1 (edge) or 0

Diagonal degree matrix Λ ($\Lambda_{ii} = \sum_{j=1}^n W_{ij}$)

$$\Lambda = \begin{bmatrix} 1.33 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1.26 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1.33 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.66 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.26 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.66 \end{bmatrix}$$

$$W = \begin{bmatrix} 0.00 & 0.00 & 1.00 & 0.00 & 0.33 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.33 & 0.60 & 0.33 \\ 1.00 & 0.00 & 0.00 & 0.00 & 0.33 & 0.00 \\ 0.00 & 0.33 & 0.00 & 0.00 & 0.00 & 0.33 \\ 0.33 & 0.60 & 0.33 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.33 & 0.00 & 0.33 & 0.00 & 0.00 \end{bmatrix}$$

(Unnormalized) Laplacian matrix $\mathbf{L} = \mathbf{\Lambda} - \mathbf{W}$

$$\mathbf{L} = \begin{bmatrix} 1.33 & 0.00 & -1.00 & 0.00 & -0.33 & 0.00 \\ 0.00 & 1.26 & 0.00 & -0.33 & -0.60 & -0.33 \\ -1.00 & 0.00 & 1.33 & 0.00 & -0.33 & 0.00 \\ 0.00 & -0.33 & 0.00 & 0.66 & 0.00 & -0.33 \\ -0.33 & -0.60 & -0.33 & 0.00 & 1.26 & 0.00 \\ 0.00 & -0.33 & 0.00 & -0.33 & 0.00 & 0.66 \end{bmatrix}$$

⇒ normalized Laplacian matrices:

- Random-walk Laplacian $\mathbf{L}_{rw} = \mathbf{\Lambda}^{-1}\mathbf{L}$
- Symmetric Laplacian $\mathbf{L}_{sym} = \mathbf{\Lambda}^{-0.5}\mathbf{L}\mathbf{\Lambda}^{-0.5}$

Idea of 1D spectral embedding & clustering

Goal: find embedding $\mathbf{y} = (y_1, \dots, y_n)^T$, where each y_i corresponds v_i and $cost(G, \mathbf{y})$ minimal.

$$cost(G, \mathbf{y}) = \sum \sum w_{ij}(y_i - y_j)^2 = 2\mathbf{y}^T \mathbf{L} \mathbf{y}$$

- we want to avoid trivial solution $\forall i : y_i = 0 \rightarrow$
- scaling constraint (e.g.) $\mathbf{y}^T \mathbf{y} = 1$ (i.e., $\sum_i y_i^2 = 1$)
- \mathbf{L} is positive semidefinite (eigenvalues λ_i real, $\lambda_i \geq 0$)
- solution smallest non-trivial eigenvector of \mathbf{L}

Extra: Why eigenvectors y of L would be the solution?

Task: Find y such that $2y^T Ly$ minimal given constraint
 $y^T y = 1$

Method of Lagrange multipliers:

1. Reformulate as a Lagrangian function
$$\mathcal{L}(y, \lambda) = y^T Ly - \lambda(y^T y - 1)$$
2. Set the partial derivatives (with respect to y and λ) as 0
3. Reduces to $Ly = \lambda y$ **Eigenvalue & -vector definition!**

Idea of 1D spectral embedding & clustering

- solution smallest non-trivial eigenvector \mathbf{y} of \mathbf{L}
- $cost = 2\mathbf{y}^T \mathbf{Ly} = 2\mathbf{y}^T \lambda \mathbf{y} = 2\lambda(y_1^2 + \dots + y_n^2) = 2\lambda$ (λ eigenvalue)
- $cost$ minimal, when λ minimal (recall $\mathbf{y}^T \mathbf{y} = 1$)
- but **skip trivial solution** $\lambda = 0$ with \mathbf{y} (proportional to)
 $\mathbf{1} = (1, \dots, 1)^T$
 - exists always when \mathbf{G} connected
- → optimal solution **eigenvector corresponding to the 2nd smallest λ**
- cluster elements of \mathbf{y} with K -means

Example

Unnormalized Laplacian L

$$\begin{bmatrix} 1.33 & 0.00 & -1.00 & 0.00 & -0.33 & 0.00 \\ 0.00 & 1.26 & 0.00 & -0.33 & -0.60 & -0.33 \\ -1.00 & 0.00 & 1.33 & 0.00 & -0.33 & 0.00 \\ 0.00 & -0.33 & 0.00 & 0.66 & 0.00 & -0.33 \\ -0.33 & -0.60 & -0.33 & 0.00 & 1.26 & 0.00 \\ 0.00 & -0.33 & 0.00 & -0.33 & 0.00 & 0.66 \end{bmatrix}$$

Eigenvalues:

$\approx 0, 0.20, 0.99, 0.99, 1.99, 2.33^*$

Second smallest eigenvector:

$(0.48, -0.19, 0.48, -0.48, 0.19, -0.48)^T$

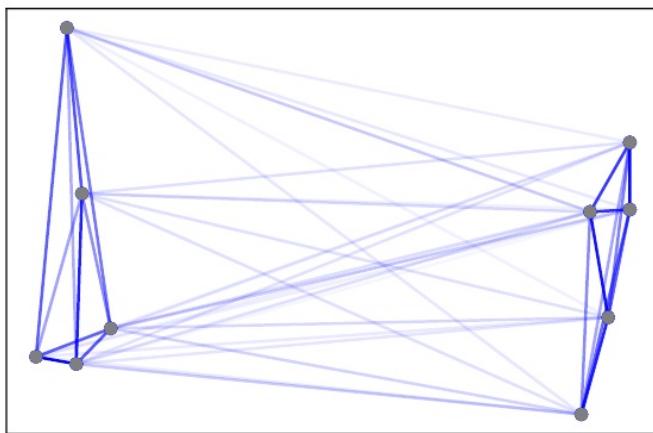
The new representation can be clustered by K -means:



* 1st eigenvalue 1.9e-16 due to imprecision (should be 0)

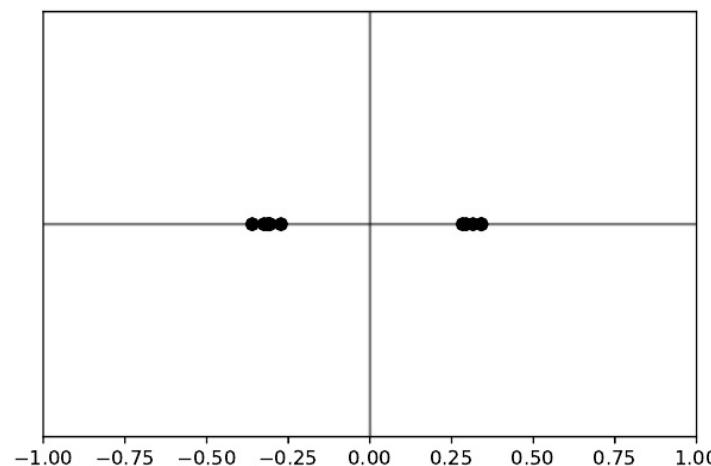
Another example with 1D embedding

Fully connected weighted graph.



Eigenvector:

$$(0.32, 0.34, 0.28, 0.34, 0.29, -0.32, -0.27, -0.31, -0.36, -0.31)^T$$



$$n = 10$$

Example by Bruno Ordozgoiti, MDM 2020

Generalization with multidimensional embedding

Unnormalized spectral clustering

Input: Graph \mathbf{G} with adjacency matrix \mathbf{W} , number of clusters K .

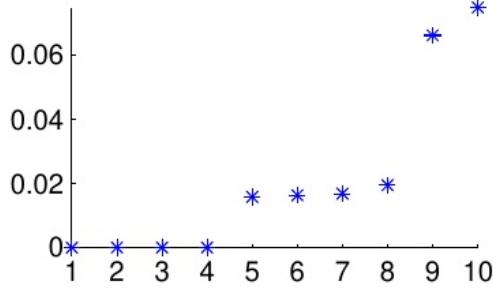
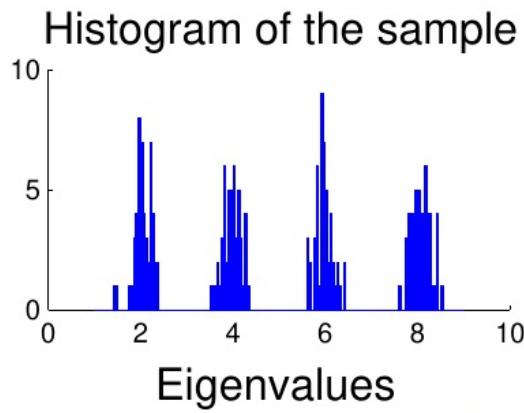
1. Compute the Laplacian $\mathbf{L} = \mathbf{\Lambda} - \mathbf{W}$
2. Compute the **eigenvectors** $\mathbf{y}_1, \dots, \mathbf{y}_k$ of \mathbf{L} corresponding to the k smallest eigenvalues (excluding $\lambda = 0$)
3. Present the data as matrix \mathbf{Y} whose columns are $\mathbf{y}_1, \dots, \mathbf{y}_k$.
4. Cluster \mathbf{Y} with K -means.

Note: Usually $k = K$ or $k < K$. Eigengap $|\lambda_{k+1} - \lambda_k|$ can be used to choose k .

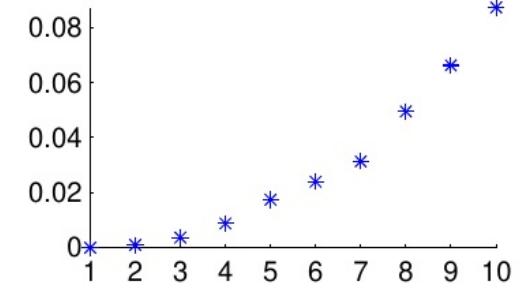
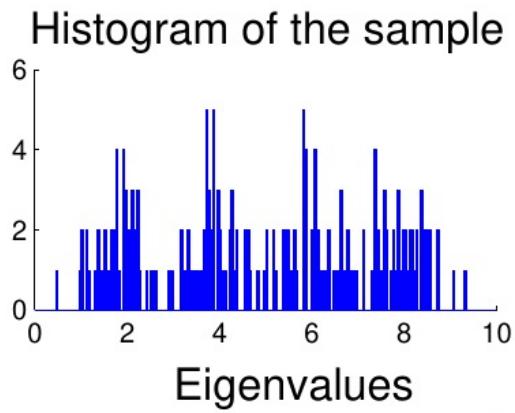
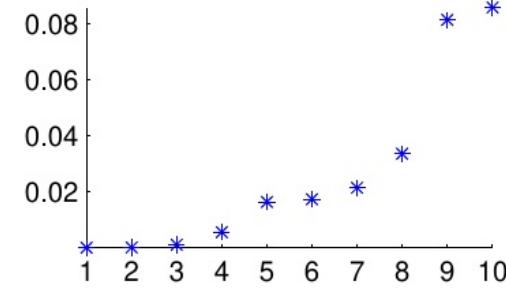
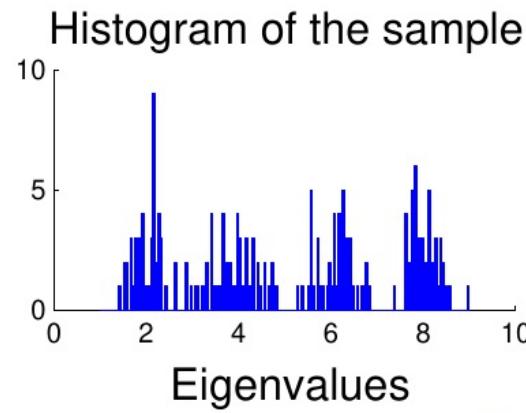
Eigengap heuristic for choosing k

Choose k such that $\lambda_1, \dots, \lambda_k$ small but λ_{k+1} relatively large.

clear gap



no gap



Normalized spectral clustering using random walk Laplacian \mathbf{L}_{rw}

Input: Graph \mathbf{G} with adjacency matrix \mathbf{W} , number of clusters K .

1. Compute the **random walk Laplacian** $\mathbf{L}_{rw} = \mathbf{\Lambda}^{-1} \mathbf{L}$
2. Compute the right eigenvectors $\mathbf{y}_1, \dots, \mathbf{y}_k$ of \mathbf{L}_{rw} corresponding to the k smallest eigenvalues (excluding $\lambda = 0$)
3. Present the data as matrix \mathbf{Y} whose columns are $\mathbf{y}_1, \dots, \mathbf{y}_k$.
4. Normalize the columns of \mathbf{Y} to unit norm.
5. Cluster \mathbf{Y} with K -means.

Normalized spectral clustering using symmetric normalized Laplacian \mathbf{L}_{sym}

Input: Graph \mathbf{G} with adjacency matrix \mathbf{W} , number of clusters K .

1. Compute the **symmetric normalized Laplacian**
$$\mathbf{L}_{sym} = \Lambda^{-1/2} L \Lambda^{-1/2}$$
2. Compute the eigenvectors $\mathbf{y}_1, \dots, \mathbf{y}_k$ of \mathbf{L}_{sym} corresponding to the k smallest eigenvalues (excluding $\lambda = 0$)
3. Present the data as matrix \mathbf{Y} whose columns are $\mathbf{y}_1, \dots, \mathbf{y}_k$.
4. Normalize the rows of \mathbf{Y} to unit norm.
5. Cluster \mathbf{Y} with K -means.

Important choices

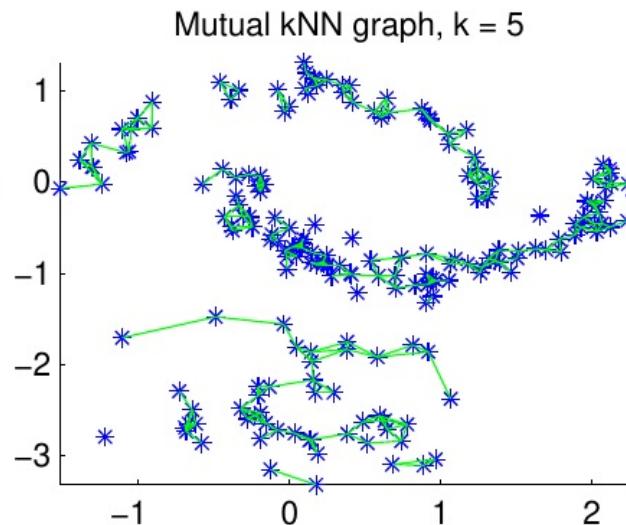
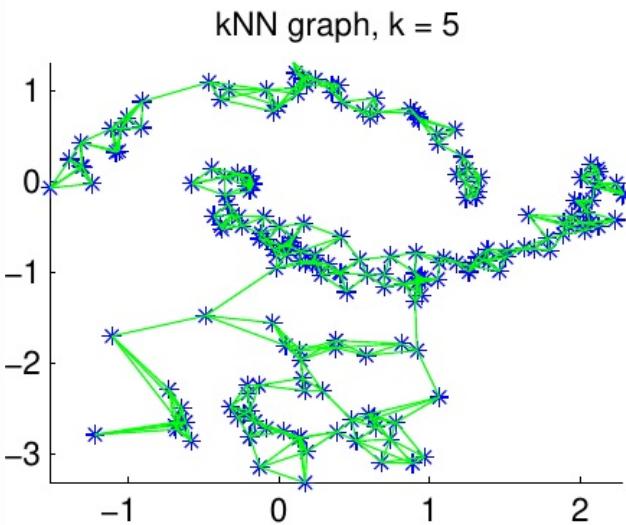
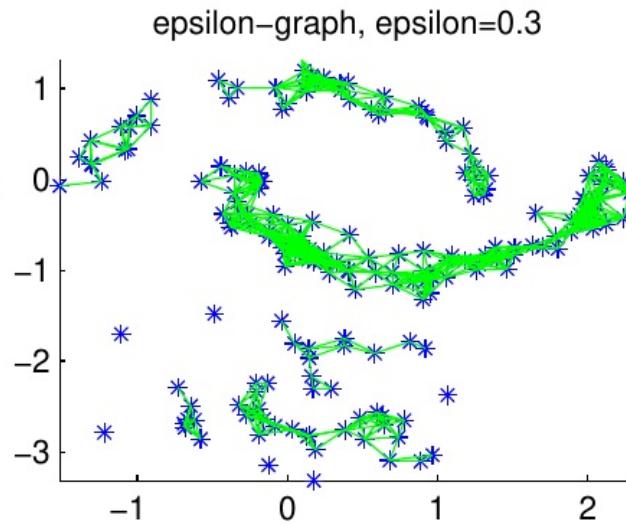
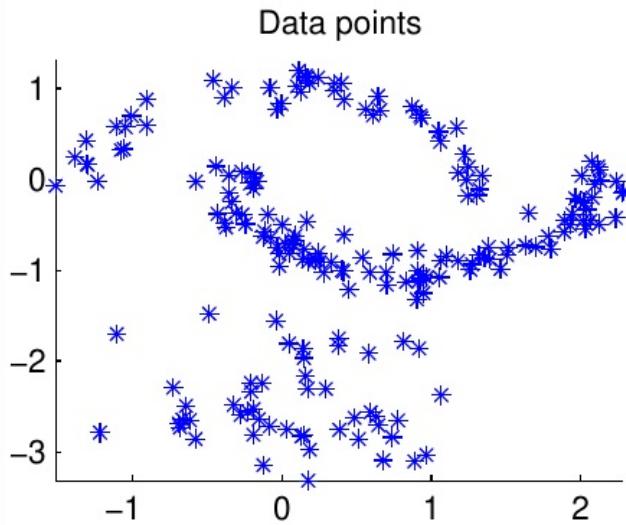
- **Method:** Unnormalized, random walk or symmetric normalized?
 - Usually normalization helps. Suggestion: try random walk first.
- **Similarity measure**
 - should measure local similarity reliably (close neighbours)
 - for numeric data, Gaussian similarity $\exp\left(\frac{-\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right)$ often used
- **Similarity graph and its parameters**
 - this has a strong effect on results!

Common choices for the similarity graph

General goal: sparse but connected graph (or number of connected components $\ll K$)

1. **ϵ -neighbourhood** graph: keep only $w_{ij} \geq \epsilon$
 - problems if clusters of different densities
2. **k -nearest neighbour** graph: v_i among k nearest neighbours of v_j **or** vice versa
 - often a good first choice
 - can break the graph into disconnected components
3. **mutual k -nearest neighbour** graph: v_i among k nearest neighbours of v_j **and** vice versa

Similarity graph examples (von Luxburg, Fig 3)



Similarity graph (cont)

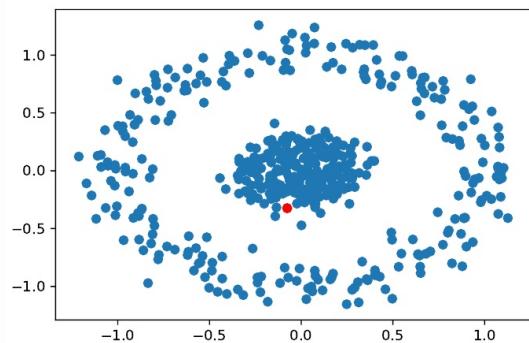
4. fully connected graph

- often with Gaussian similarity $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(\frac{-\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right)$ (radial basis function, RBF)
- how to choose σ ?
- Note: in scikitlearn parameter $\gamma = \frac{1}{2\sigma^2}$
- graph not sparse → heavy computation

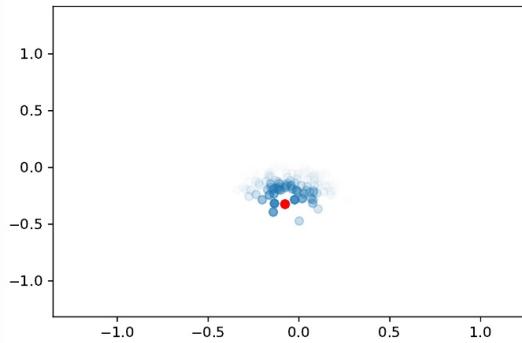
Choice of parameters (ϵ, k, σ) affects a lot, too!

Example: neighbourhood with $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(\frac{-\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right)$

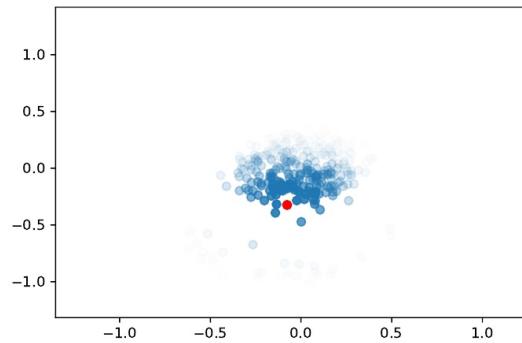
Data, query point red



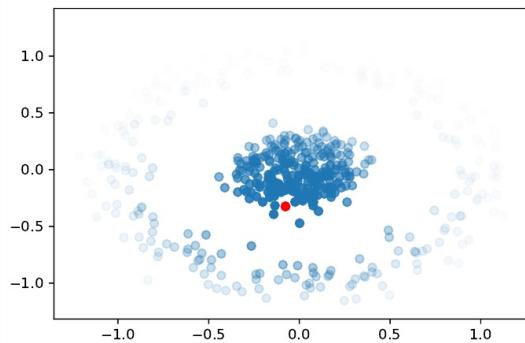
$\sigma = 0.1$



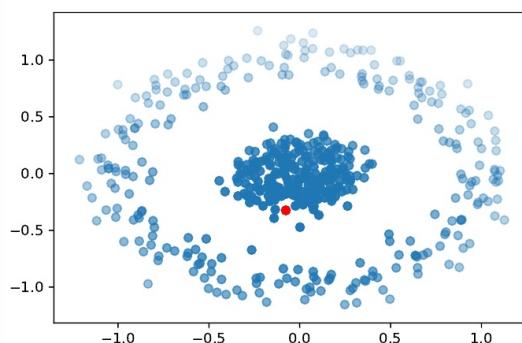
$\sigma = 0.2$



$\sigma = 0.4$



$\sigma = 0.8$



$\sigma = 1.6$

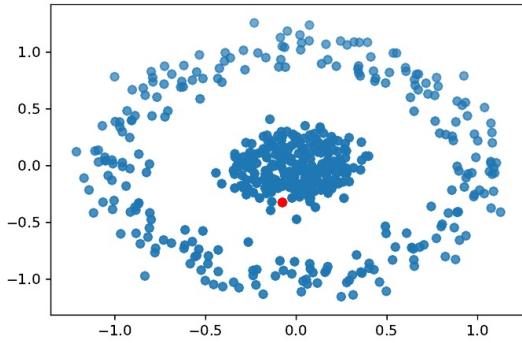
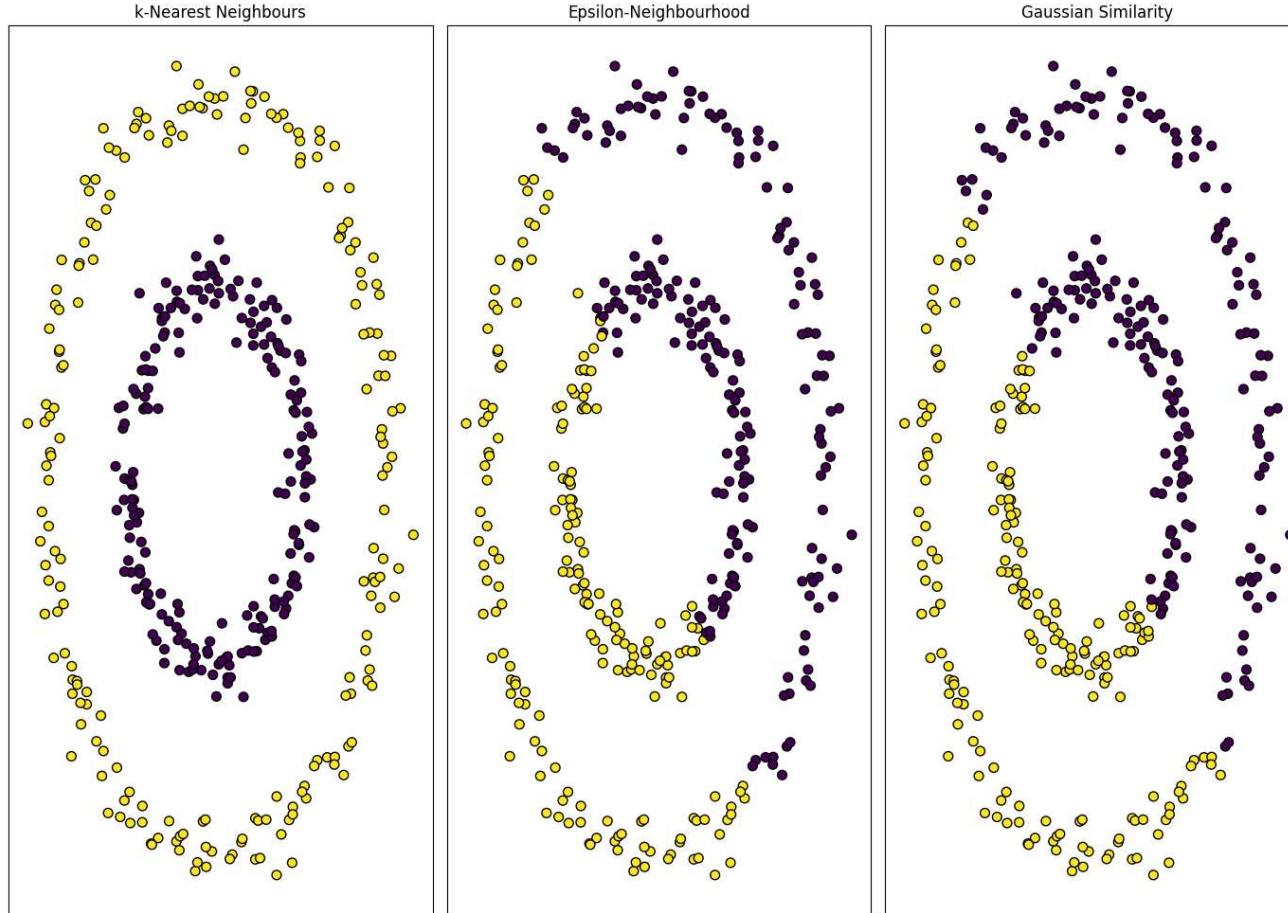


Image source: Bruno Ordozgoiti, MDM 2020 slides

Example: Different clustering results with different similarity graphs



Parameters: $k = 2$, $\epsilon = 0.3$, RBF with $\gamma = 10$ (i.e., $\sigma = \sqrt{5}$)

Experiment by Lai Khoa for MDM 2023

Summary

Idea: similarity graph → low-dimensional VS presentation
(eigenvectors) → clustering (K -means etc.)

- + very powerful (virtually any datatype, arbitrary shapes)
- computationally expensive
 - creating similarity graph $O(n^2)$, spectral decomposition $O(n^3)$
- many important parameter choices

Further reading

von Luxburg: A Tutorial on Spectral Clustering. Statistics and Computing, vol. 17, pp. 395–416, 2007.

Reading guide: Sec 2 overview, 2.2, Sec 3 overview + definitions of Laplacian matrices from 3.1-3.2, Sec 4, Sec 5 overview, (possibly Sec 6 overview), Sec 8. ^a

^asection overview = text before subsections

Mining association patterns (Part 1)

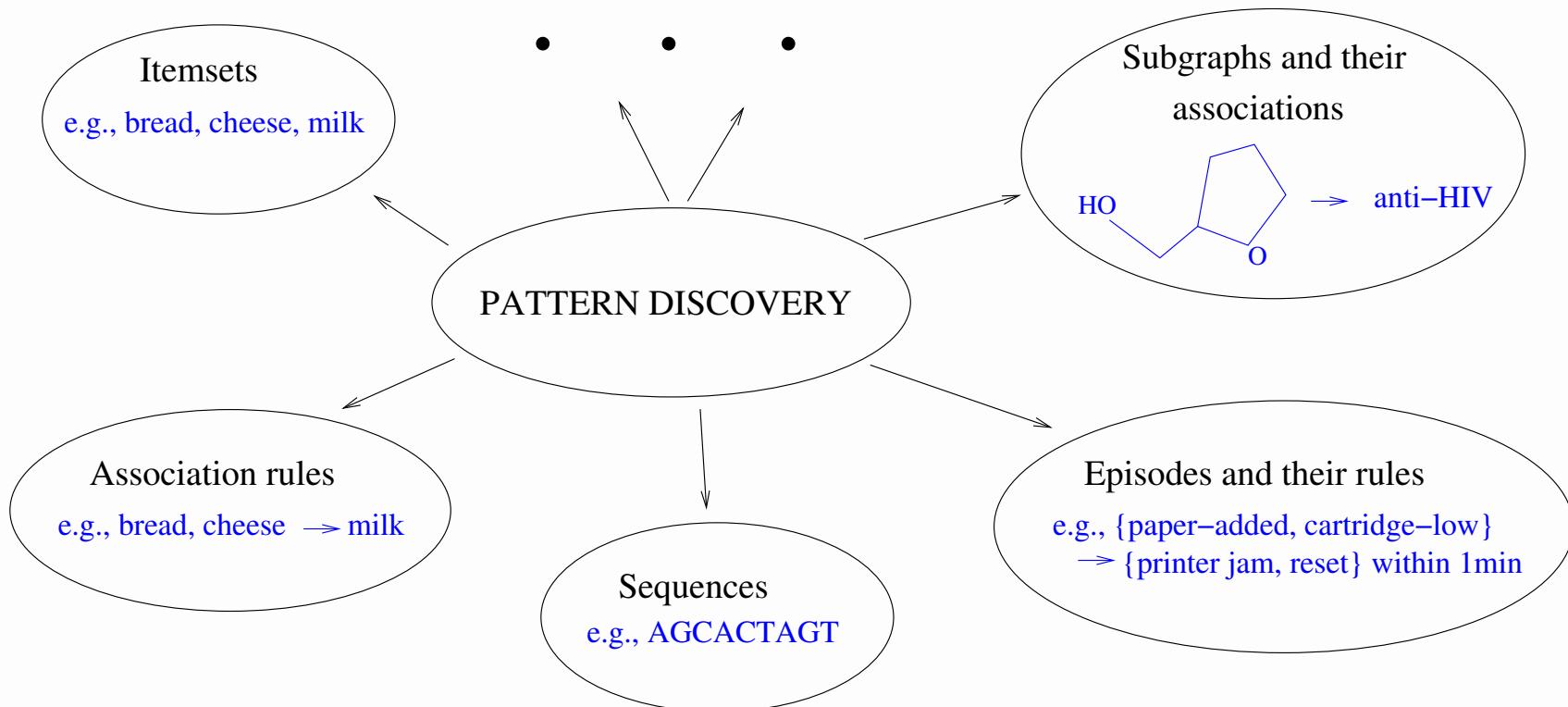
milk, cheese and bread
are often bought together

genes g1, g2, g3 and g4
are often over-expressed
in DLBC lymphomas

occurrence of certain insect species
makes it more likely to meet the
threatened white-backed woodpecker



Pattern discovery: search for all sufficiently good/top-K patterns of a certain type



Many variants of pattern types!

Contents for the next lectures

- Overview
 - pattern types, notions of association
- Frequent sets and association rules
 - Pruning the search space (monotonicity, Apriori)
 - Condensed representations
- Search for statistically significant association rules
 - Measures and algorithms, filtering redundant associations
- Advanced topics
 - Computational strategies, generic Apriori, etc.

1. Why associative pattern mining?

- really efficient way of discovering associations in large data sets!
 - sometimes globally optimal solutions to NP -hard optimization problems!
 - can handle even 20 000 attributes and millions of samples in a few minutes
 - for binary data – other data types should be binarized
- numerous applications!
- dependency analysis often a first step of data modelling \Rightarrow helps to choose methods (and features)
- as a subroutine of other methods
 - e.g., associative classifiers, clustering \square^a

^ae.g., Li & Zaiane 2017, Zimek et al. 2014, Zimmermann & Nijssen 2014

2. Data: typically occurrence data

	milk	juice	bread	cheese	oranges
basket1	1	0	1	1	0
basket2	1	1	0	0	1
basket3	0	1	1	1	0
basket4	1	0	1	1	1
basket5			:		

- items in market baskets
- species in ecological sites
- over-expressed or under-expressed genes in samples
- feature extraction for other data types

Data presented as **transactions** listing only 1-valued attributes: {milk, bread, cheese}, {milk, juice, oranges}, {juice, bread, cheese}, {milk, bread, cheese, oranges}, ...

Data: Binarization and discretization

Categorical: Create a new binary attribute for each value

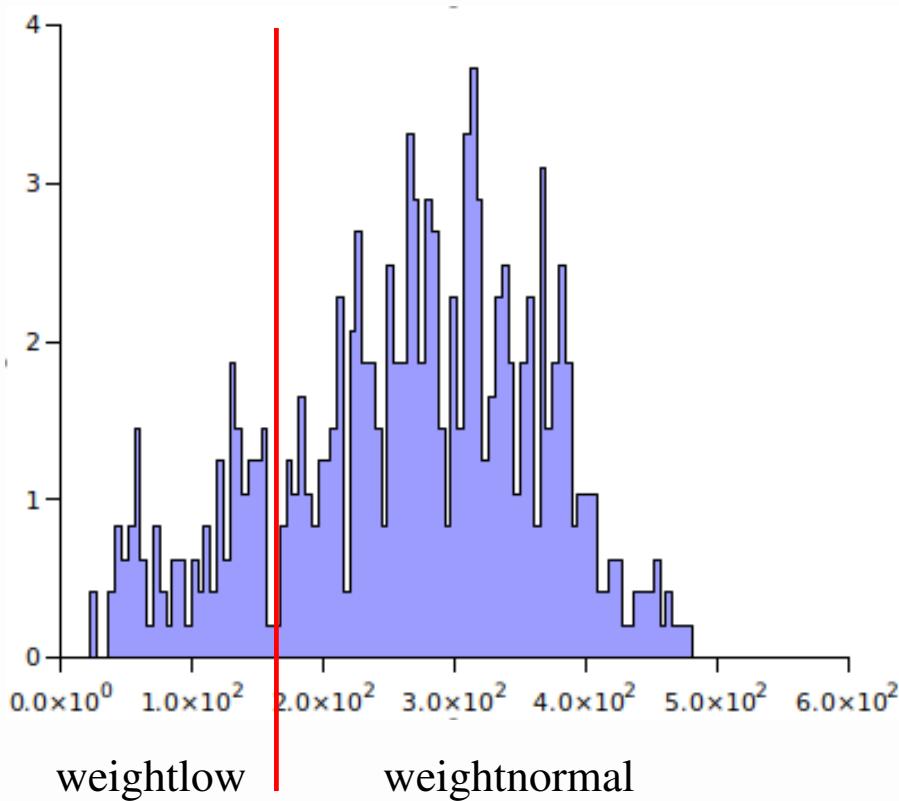
- $\text{Colour}=\{\text{red, blue, green}\} \Rightarrow \text{attributes } C_{\text{red}}, C_{\text{blue}}, C_{\text{green}}$
- usually needed also for binary features!
 $\text{Gender}=\{\text{F, M}\} \Rightarrow \text{attributes } F, M$
- no information loss!

Numerical: Discretize into bins and create a new attribute for each (interesting) bin

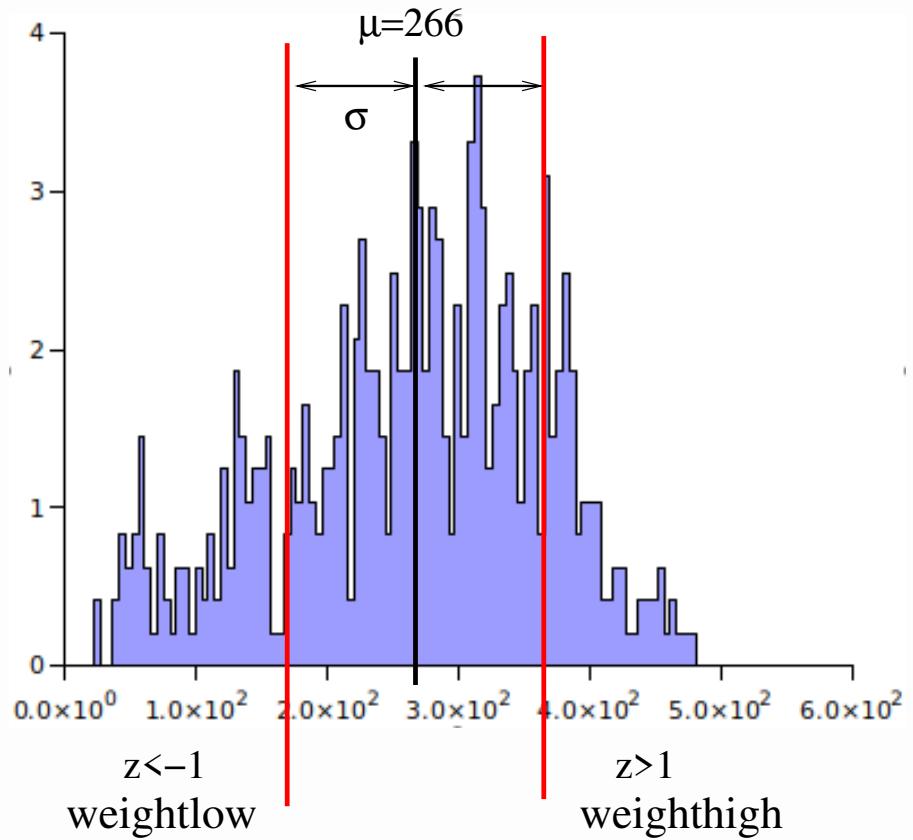
- Many approaches!
- static vs. dynamic, disjoint vs. overlapping, entire range vs. distribution tails
- loses some information, but also reduces noise

Example: 4 ways to discretize the rat's weight

1. One visually determined cut-off
entire range, two attributes



2. z-score discretization
attributes only for extreme values



3. Equi-width or 4. Equi-depth discretization of range [20, 500]

3. Types of association patterns

Given binary occurrence data

$\mathbf{R} = \{A_1, \dots, A_k\}$ set of binary attributes ($\text{Dom}(A_i) = \{0, 1\}$)

$\mathcal{D} = \{d_1, \dots, d_n\}$, $d_i \in \text{Dom}(A_1) \times \dots \times \text{Dom}(A_k)$, data

Association patterns can be

1. **sets** $\wedge A_{i_j}=a_{i_j}$ ($A_{i_j} \in \mathbf{R}$, $a_{i_j} \in \{0, 1\}$) such that all $A_{i_j}=a_{i_j}$ are associated
e.g., *milk=1, cheese=1, bread=1* (these items occur often together in a market basket), or
2. **rules** $\wedge A_{i_j}=a_{i_j} \rightarrow C=c$, $A_{i_j}, C \in \mathbf{R}$, $a_{i_j}, c \in \{0, 1\}$, such that rule condition and consequence are associated
e.g., *cheese=1 → bread=1* (people who buy cheese tend to buy bread, too)

Simplification

Usually we are interested in patterns containing only 1-valued attributes \Rightarrow Simplified notations:

- sets $X \subseteq R$ (simply list elements of set)
e.g., *milk, cheese, bread*
- rules $X \rightarrow C$, where $A_i = 1$ for all $A_i \in X$
e.g., *cheese → bread*

Notes:

- if needed, you can create new attributes A_{neg} for $\neg A$
- we concentrate on single attribute consequences
(could be a set, too)

What association means??

Statistics

- = **statistical dependence** between categorical variables
- defined by the opposite, statistical independence
- measures for the strength of association
- statistical significance: is the observed association spurious?

Frequent pattern mining

- = **frequent co-occurrence** of attributes (given some minimum frequency)
- extra criteria to filter interesting patterns
- statistical measures and tests can also be applied (post-processing)

Statistical independence and dependence

- Events $A=a$ and $B=b$ are **statistically independent**, if
 $P(A=a, B=b) = P(A=a)P(B=b)$
- Variables A and B are **statistically independent**, if for all
 $a \in Dom(A), b \in Dom(B), P(A=a, B=b) = P(A=a)P(B=b)$
- If A and B binary, these conditions are equivalent!
- **Leverage** δ and **lift** γ measure the strength of
dependence ^a 

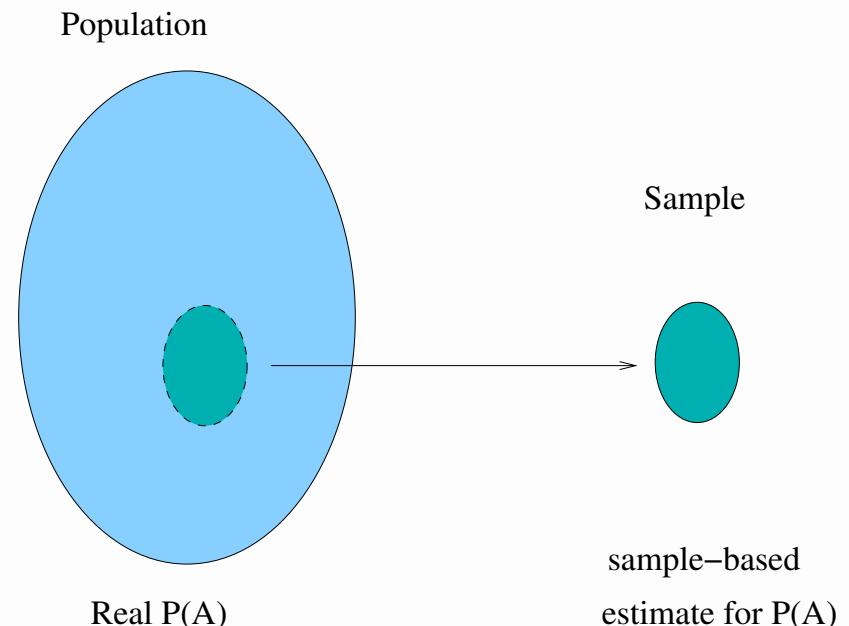
$$\delta(A=a, B=b) = P(A=a, B=b) - P(A=a)P(B=b)$$

$$\gamma(A=a, B=b) = \frac{P(A=a, B=b)}{P(A=a)P(B=b)}$$

^aAlso notations $\delta(\mathbf{X} \rightarrow C)$, $\gamma(\mathbf{X} \rightarrow C)$

Note: Simplified notation $P = \tilde{P}$

- in the definition of statistical independence, P refers to real (but unknown) probability
- its maximum likelihood estimate is **relative frequency** in data, $\tilde{P}(A) = \frac{fr(A)}{n}$, where $fr(A)$ =number of rows containing A , n =number of rows in data
- Here we use $P = \tilde{P}$



Statistical (in-)dependence in rules $\mathbf{X} \rightarrow C=c$

\mathbf{X} lists true-valued attributes, e.g., $\mathbf{X} = A, B, D$.

Notate $C=1$ by C and $C=0$ by $\neg C$.

- \mathbf{X} and C are **independent**, if $P(\mathbf{X}, C) = P(\mathbf{X})P(C)$ ($\delta = 0$, $\gamma = 1$) ("independence rule")
- \mathbf{X} and C are **positively associated**, if $P(\mathbf{X}, C) > P(\mathbf{X})P(C)$ ($\delta > 0$, $\gamma > 1$) (rule $\mathbf{X} \rightarrow C$)
- \mathbf{X} and C are **negatively associated**, if $P(\mathbf{X}, C) < P(\mathbf{X})P(C)$ ($\delta < 0$, $\gamma < 1$). Now \mathbf{X} and $\neg C$ positively associated!
(rule $\mathbf{X} \rightarrow \neg C$ 

^aCustomary to present positive associations, if both C and $\neg C$ are allowed

Note on confidence (precision)

Strength of frequent association rules is often measured with “confidence” (precision) ϕ (cf) + required $\phi \geq \min_{cf}$

$$\phi(\mathbf{X} \rightarrow C) = P(C|\mathbf{X}) = \frac{P(\mathbf{X}C)}{P(\mathbf{X})}$$

But high ϕ does not guarantee statistical dependence!

	Coffee	$\overline{\text{Coffee}}$	
Tea	15	5	20
$\overline{\text{Tea}}$	75	5	80
	90	10	100

(image from Tan et al. 2018)

e.g., $\min_{cf} = 0.70$
 $\phi(\text{tea} \rightarrow \text{coffee}) = 0.75$
 $P(\text{coffee}) = 0.90$
 $\gamma(\text{tea} \rightarrow \text{coffee}) = 0.75/0.90$
 $= 0.83 < 1$ **neg. association**

Statistical independence in sets X

Let $\mathbf{X} = A_1, \dots, A_m$ and $a_i \in \{0, 1\}$.

Variables $A_i \in \mathbf{X}$ are **mutually independent**, if for all $a_i \in \{0, 1\}$

$$P\left(\bigwedge_{A_i \in \mathbf{X}} A_i = a_i\right) = \prod_{A_i \in \mathbf{X}} P(A_i = a_i)$$

Equivalently, \mathbf{X} is an “**independence set**”, if for all $\mathbf{Y} \subseteq \mathbf{X}$

$$P(\mathbf{Y}) = \prod_{A_i \in \mathbf{Y}} P(A_i = 1)$$

Note

It is possible that $P(\mathbf{X}) = \prod_{A_i \in \mathbf{X}} P(A_i=1)$, even if \mathbf{X} is not an independence set!

E.g., *bread, cheese, juice*:

$$P(B, C, J) = 0.06 = 0.4 \cdot 0.3 \cdot 0.5 = P(B)P(C)P(J), \text{ but}$$

$$P(B, C) = 0.25 > 0.4 \cdot 0.3 \text{ (positive association)}$$

$$P(J|BC) = 0.24 < P(J) \text{ (negative association)}$$

⇒ less misleading to report only *bread, cheese*?

Set dependencies

Many choices for dependency sets (=“correlated” sets):

- **Minimal** sets \mathbf{X} that express positive dependence between true-valued attributes: $P(\mathbf{X}) > \prod_{A_i \in \mathbf{X}} P(A_i=1)$
- Sets \mathbf{X} that express bipartition dependence: for **some** $\mathbf{Q} \subsetneq \mathbf{X}$: $P(\mathbf{X}) > P(\mathbf{X} \setminus \mathbf{Q})P(\mathbf{Q})$
i.e., association $\mathbf{X} \setminus \mathbf{Q} \rightarrow \mathbf{Q}$
- Self-sufficient sets $\overset{a}{\square} \mathbf{X}$ where for **all** $\mathbf{Q} \subsetneq \mathbf{X}$, $\mathbf{Q} \neq \emptyset$:
 $P(\mathbf{X}) > P(\mathbf{X} \setminus \mathbf{Q})P(\mathbf{Q})$ (+ some extra criteria)

^aWebb and Vreeken, 2014

Extra: Extensions of leverage and lift to sets

- Leverage

$$\delta_S(\mathbf{X}) = P(\mathbf{X}) - \prod_{A_i \in \mathbf{X}} P(A_i)$$

- Lift

$$\gamma_S(\mathbf{X}) = \frac{P(\mathbf{X})}{\prod_{A_i \in \mathbf{X}} P(A_i)}$$

Problem: $\mathbf{Y} \subsetneq \mathbf{X}$ may express association (i.e., \mathbf{X} is not independence set), even if $\delta_S(\mathbf{X}) = 0$ and $\gamma_S(\mathbf{X}) = 1$!

4. Search problems

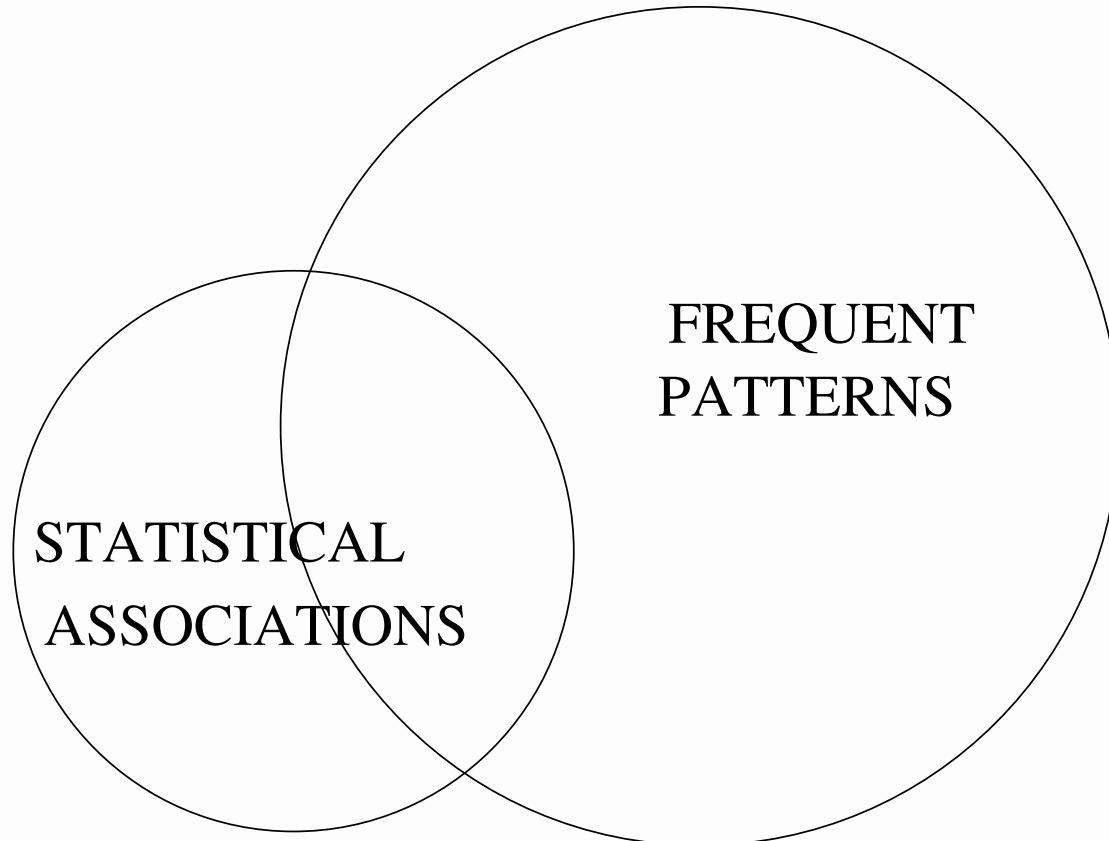
1. **Enumeration problem:** Search **all** patterns whose measure values are **sufficiently good** (given thresholds), e.g.,
 - all frequent sets given minimum frequency \min_{fr}
 - all frequent and confident rules, given \min_{fr} and \min_{cf}
 - all rules whose mutual information is at least \min_{MI}
2. **Optimization problem:** Find the **best K** patterns with a given goodness measure (+ possible extra constraints)
 - top-100 frequent rules with largest lift (given \min_{fr})
 - top-100 rules with mutual information

5. Main approaches

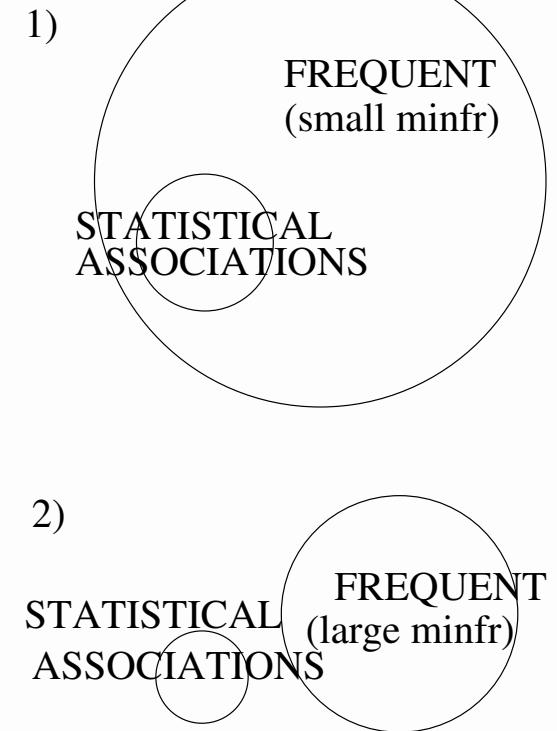
- 1. Frequency-based search**
 - i) search frequent sets
 - ii) construct rules from sets (if wanted)
 - iii) filter with statistical measures
 - iv) possibly test statistical significance
- 2. Direct search of statistical associations** with measures for the strength and/or statistical significance of association (for rules or sets)

Approach 1 is easier to implement, but often fails to find everything we want (and finds a lot that we don't want!)

Two approaches find generally different patterns!



TWO EXTREMES:



If sufficiently small min_{fr} is feasible, you can filter statistical associations afterwards (but this can be heavy!)

Empirical comparison: frequent vs. statistical associations

Proportions of top-100 statistically significant rules with χ^2 that could be found with Apriori, when min_{fr} as small as computationally possible (given 256GB memory!):

data	n	k	tlen	min_{fr}	discovered %
Mushroom	8124	119	23.0	0.01	100%
Chess	3196	75	37.0	0.20	1%
T10I4D100K	100000	870	10.1	0.00007	100%
T40I10D100K	100000	942	39.6	0.01	0%
Accidents	340183	468	33.8	0.25	14%
Pumsb	49046	2113	74.0	0.45	27%
Retail	88162	16470	10.3	0.000085	100%

Lesson to learn

Check always what is the **definition of association** of a given algorithm!

- What kind of patterns do you find precisely?
- Do you find what you want?
- If not, consider alternatives or try to tailor the algorithm to find your target patterns

6. Algorithm for frequent association mining

Let $\mathbf{R} = \{A_1, \dots, A_k\}$ binary attributes.

Focus: How to find **frequent sets** i.e., $\mathbf{X} \subseteq \mathbf{R}$ such that $P(\mathbf{X}) \geq min_{fr}$?

- rules are easy to derive: check $\mathbf{X} \setminus \{C\} \rightarrow C$ for all $C \in \mathbf{X}$
- postprocessing may still take time...

Contents:

- 6.1 Pruning the search space (monotonicity)
- 6.2 Apriori algorithm
- 6.3 Constructing rules (task)

6.1 Pruning the search space

Problem: Search space has **exponential size!**

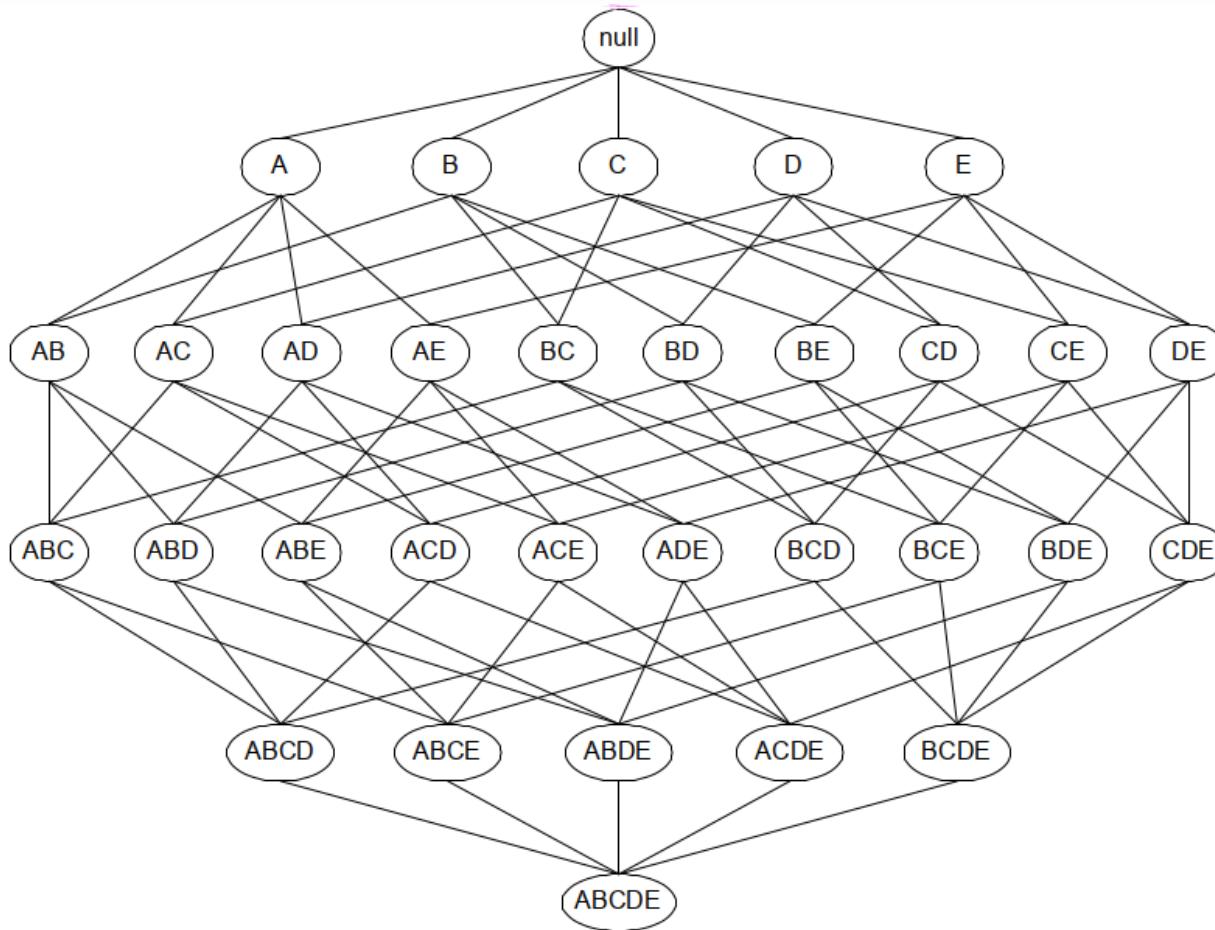
Given $\mathbf{R} = \{A_1, \dots, A_k\}$, there are

- $\sum_{i=1}^k \binom{k}{i} = 2^k - 1$ non-empty sets $\mathbf{X} \subseteq \mathbf{R}$
- $\sum_{i=2}^k i \binom{k}{i} = \sum_{i=2}^k \frac{i \cdot k!}{i!(k-i)!} = \sum_{i=2}^k \frac{k \cdot (k-1)!}{(i-1)!(k-i)!} = k(2^{k-1} - 1)$
possible rules $\mathbf{X} \setminus \{C\} \rightarrow C$

(e.g., $2^{20} \approx 10^6$, $2^{100} \approx 10^{30}$)

How to find frequent ones?

Search space as a grid for $R = \{A, B, C, D, E\}$



(image from Tan et al. 2018)

Key idea: Monotonicity of frequency

$fr(\mathbf{X})$ = absolute frequency of \mathbf{X}

$P(\mathbf{X})$ = relative frequency of \mathbf{X}

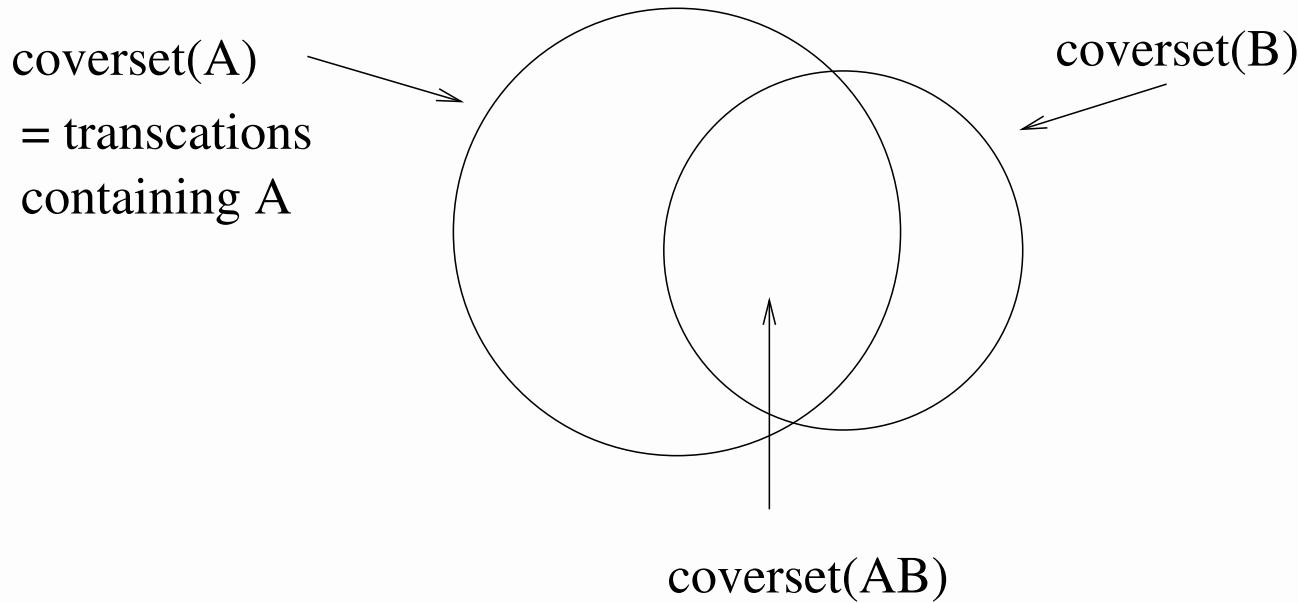
(both can be called “support”)

Frequency is **monotone** property: For all \mathbf{X}, \mathbf{Y} :

$$\mathbf{Y} \subseteq \mathbf{X} \Rightarrow fr(\mathbf{Y}) \geq fr(\mathbf{X})$$

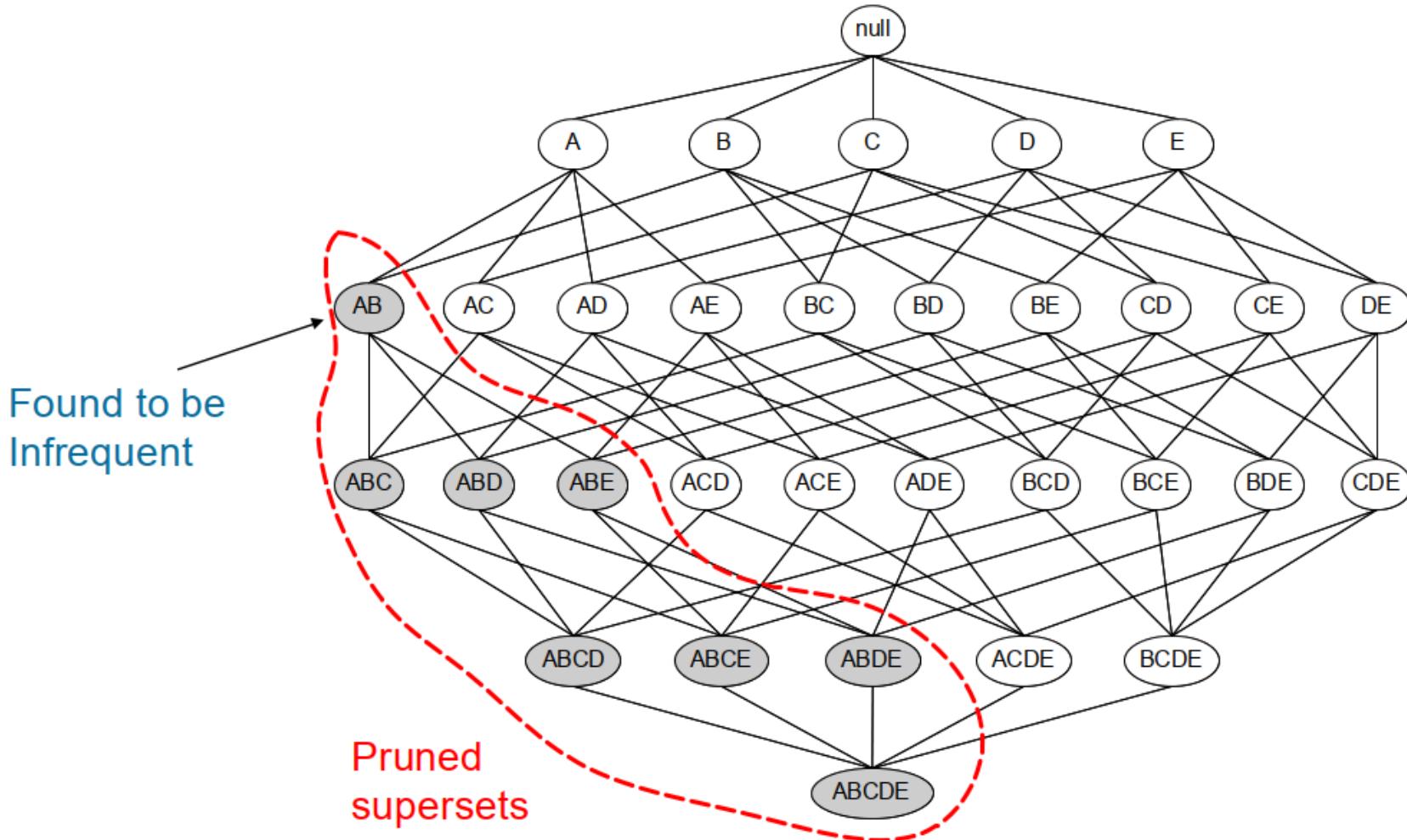
Key idea: Monotonicity of frequency

E.g., $X = \{A, B\}$ and $Y = \{A\}$. $fr(A) \geq fr(AB)$:



Consequence: If Y is infrequent ($P(Y) < min_{fr}$), then all $X \supseteq Y$ are infrequent ($P(X) < min_{fr}$)

Pruning by monotonicity



(image from Tan et al. 2018)

6.2 *Apriori* algorithm (given \mathbf{R} , \mathcal{D} and min_{fr})

\mathcal{F}_i = frequent i -itemsets, C_i = candidate i -itemsets

$i=1$

$$\mathcal{F}_1 = \{A_i \in \mathbf{R} \mid P(A_i) \geq min_{fr}\}$$

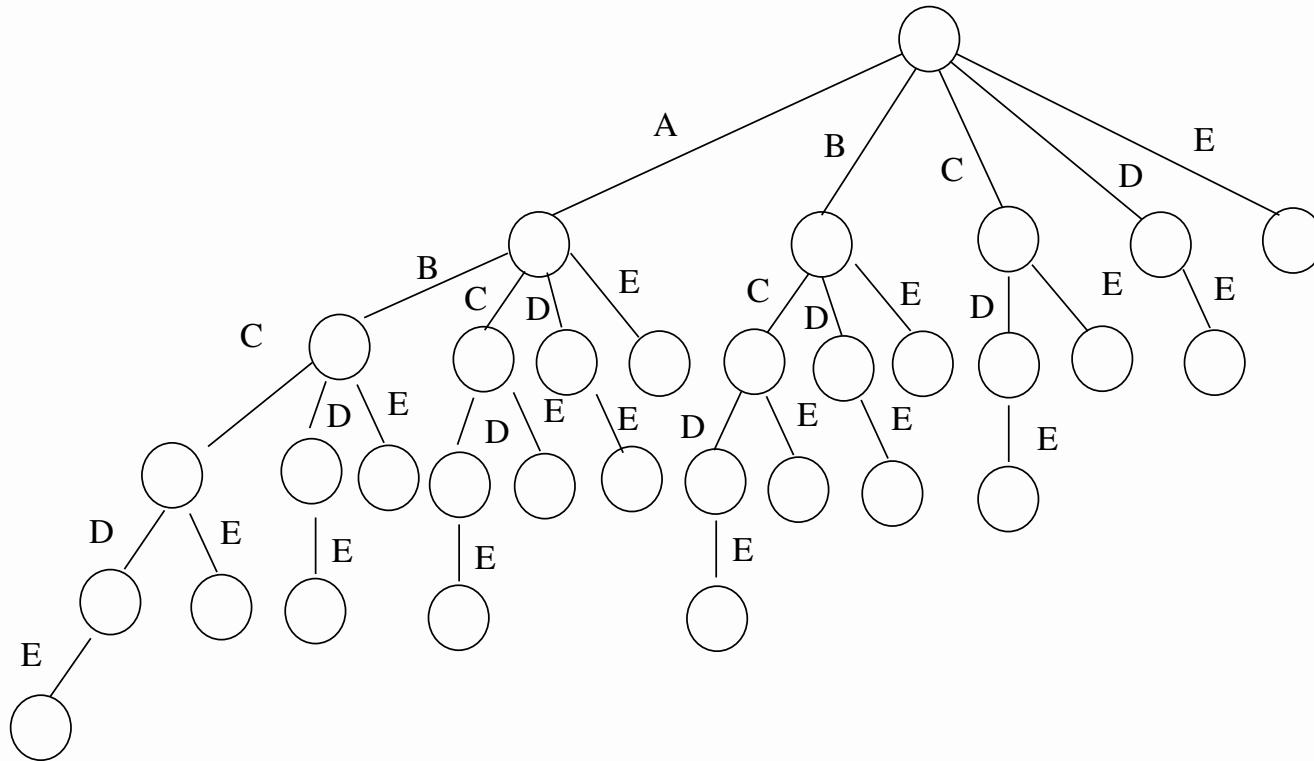
while $\mathcal{F}_i \neq \emptyset$:

- Generate candidates C_{i+1} from \mathcal{F}_i
 - Prune $\mathbf{X} \in C_{i+1}$ if $\exists \mathbf{Y} \subsetneq \mathbf{X}, |\mathbf{Y}| = i, \mathbf{Y} \notin \mathcal{F}_i$
 - Count frequencies $fr(\mathbf{X}), \mathbf{X} \in C_{i+1}$
 - Set $\mathcal{F}_{i+1} = \{\mathbf{X} \in C_{i+1} \mid P(\mathbf{X}) \geq min_{fr}\}$
 - $i = i + 1$
- } (monotonicity)

Return $\cup_i \mathcal{F}_i$

Useful data structure: enumeration tree

- Idea: each root–node path corresponds an itemset
- A complete tree has 2^k nodes ($k = |R|$) \Rightarrow construct only as much as you need!

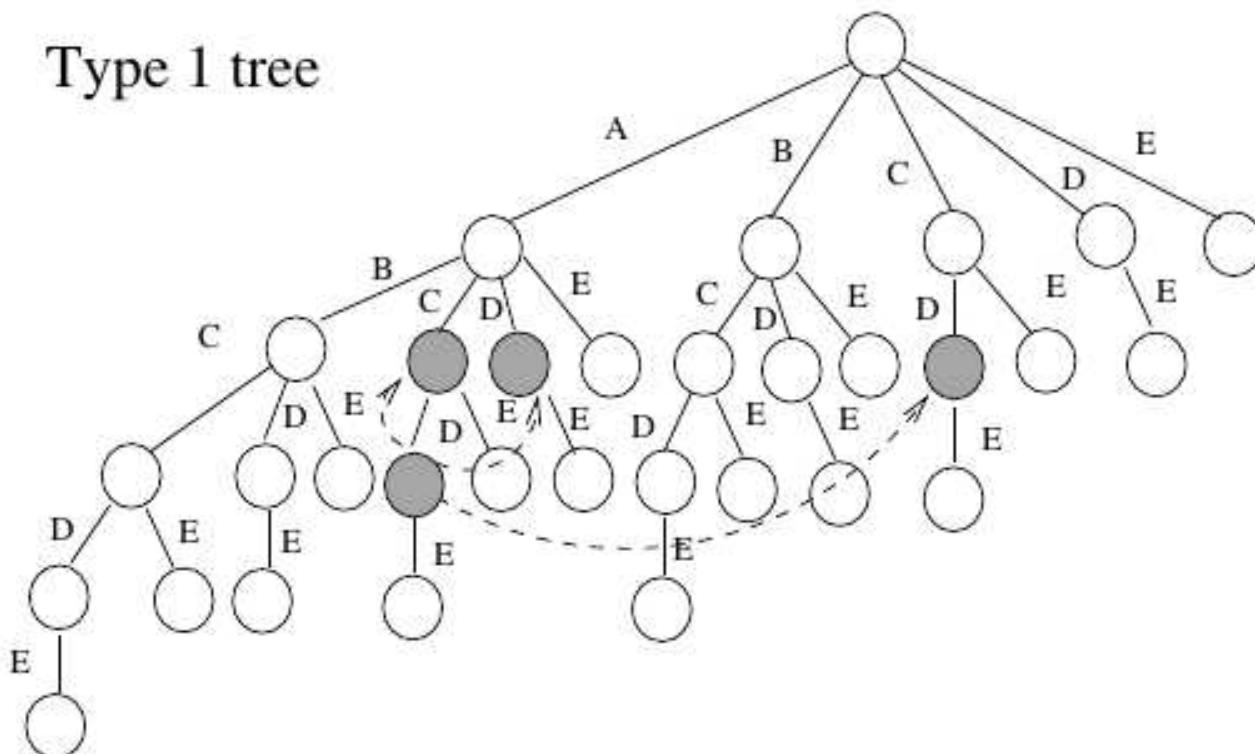


Terminology: a node may have many “parents”

parent of node X = node presenting $Y \subsetneq X$, $|X| = |Y| + 1$

Monotonicity: if any parent is infrequent, the child is infrequent

e.g., ACD has 3 parents:



Simulating Apriori

$$\mathbf{R} = \{A, B, C, D, E\}, n = 6, min_{fr} = 2/6 = 0.33$$

Transactions:

A, C, D

A, B, E

B, C, D

A, C, D, E

B, C

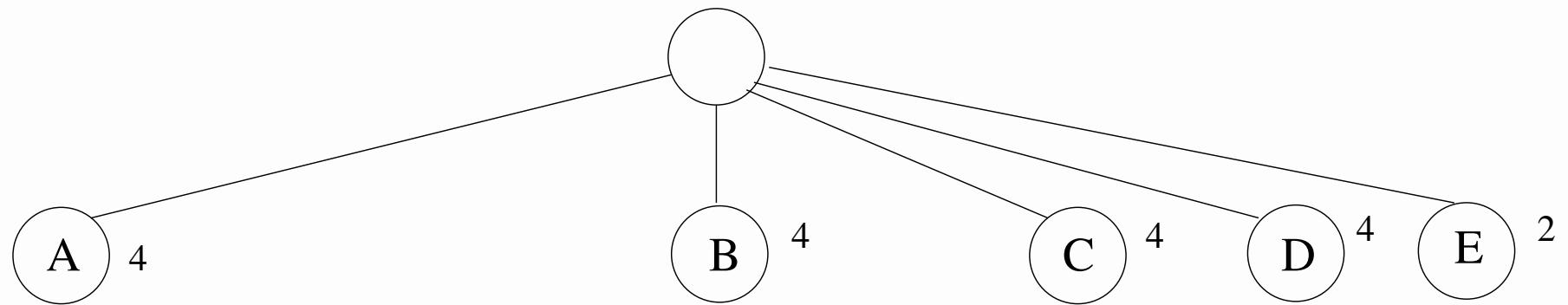
A, B, D

$$fr(A) = fr(B) = fr(C) = fr(D) = 4, fr(E) = 2$$

$A=\text{milk}$, $B=\text{juice}$, $C=\text{bread}$, $D=\text{cheese}$, $E=\text{oranges}$

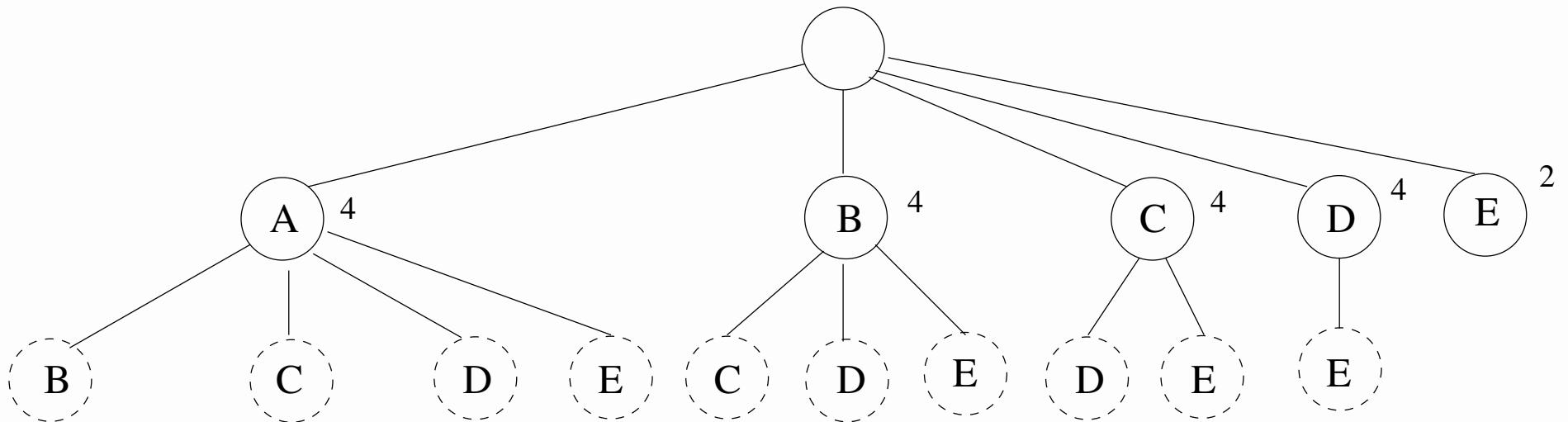
Simulation: Level $i = 1$

Check frequencies of all 1-sets and add to tree if frequent



All 1–sets frequent

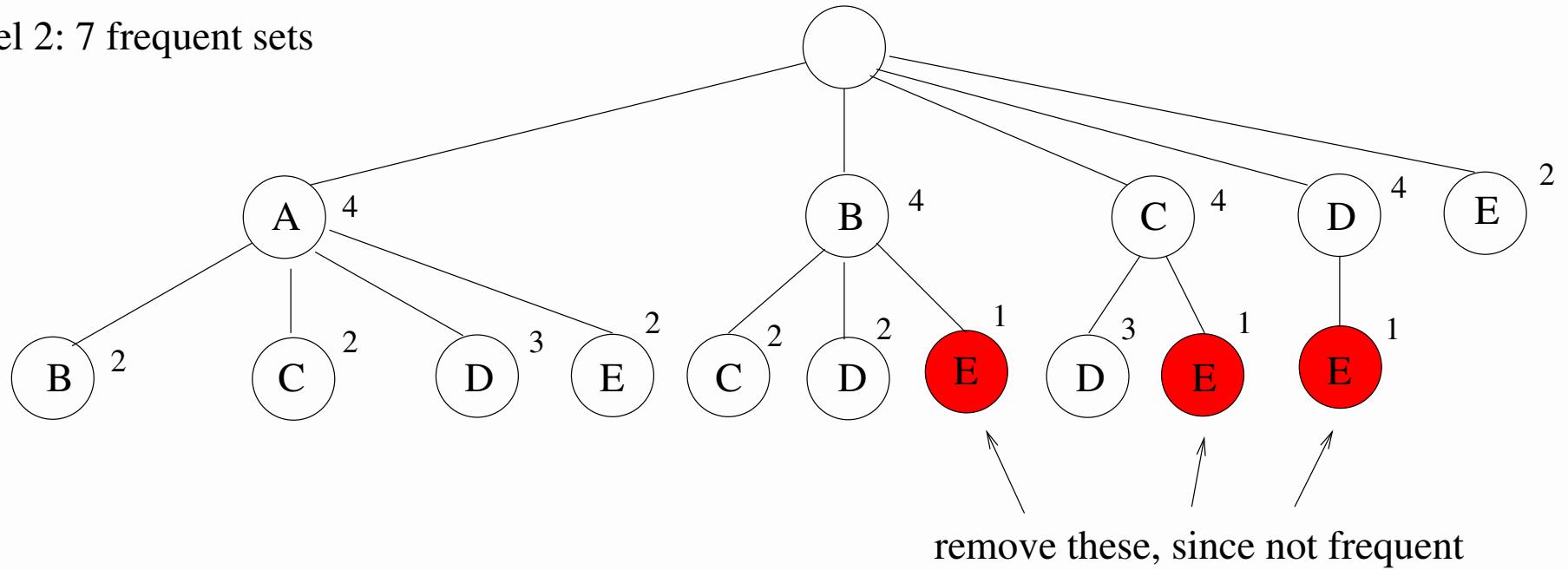
Simulation: $i = 2$ Candidate generation



Since all 1–sets frequent, check all possible 2–sets

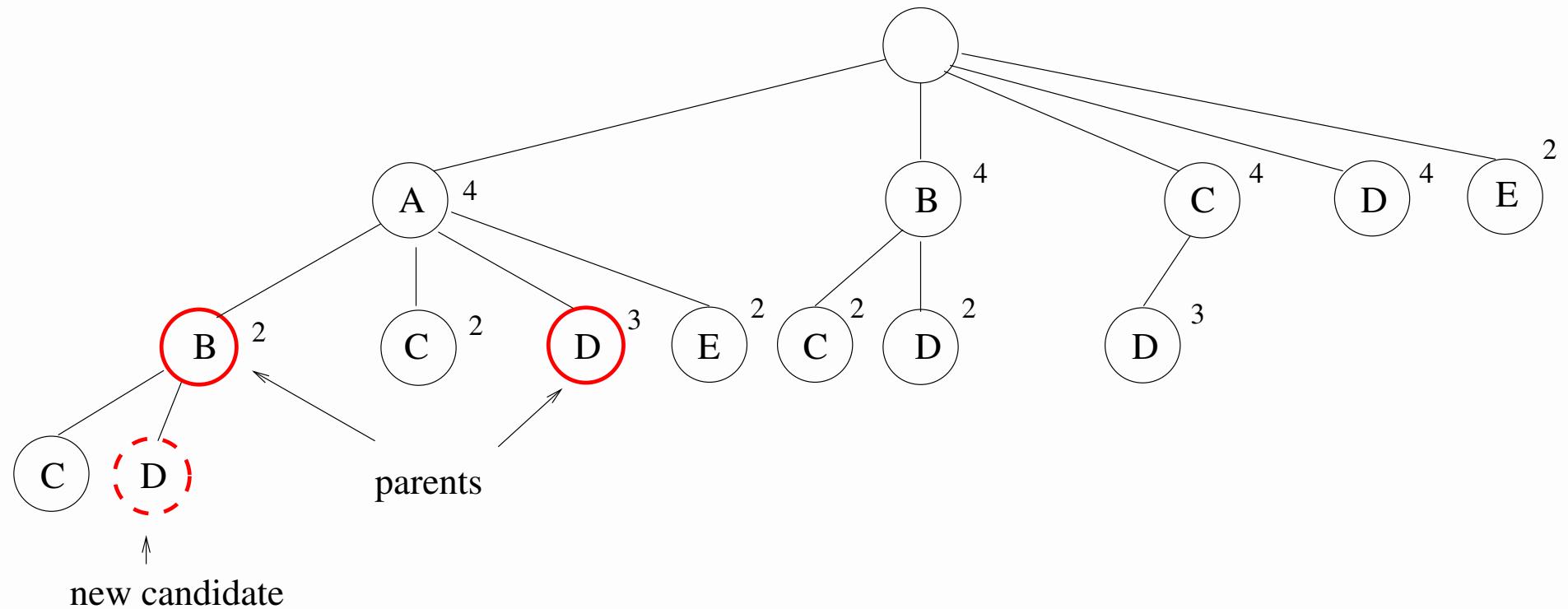
Simulation: $i = 2$ Frequency counting

Level 2: 7 frequent sets



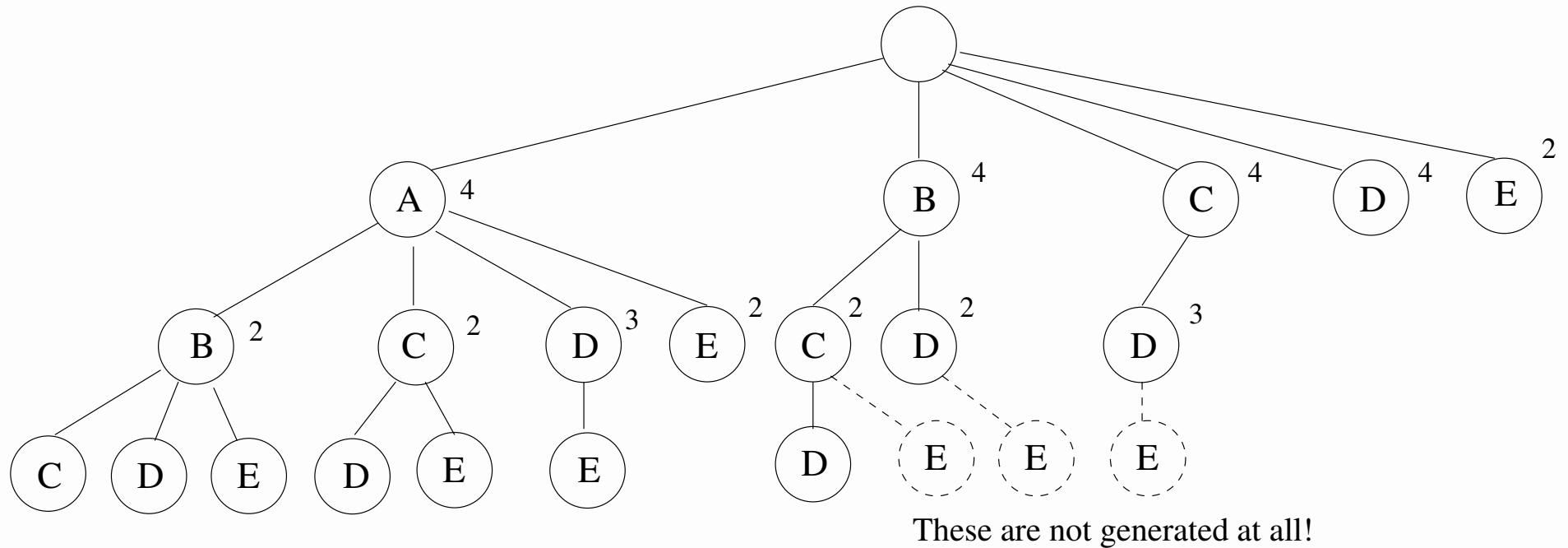
Simulation: $i = 3$ Candidate generation

Idea: Given parents $\mathbf{Y}_1, \mathbf{Y}_2 \in \mathcal{F}_i$, such that $|\mathbf{Y}_1 \cap \mathbf{Y}_2| = i - 1$, generate $(i+1)$ -candidate $\mathbf{X} = \mathbf{Y}_1 \cup \mathbf{Y}_2$. E.g.,



Simulation: $i = 3$ Candidate generation

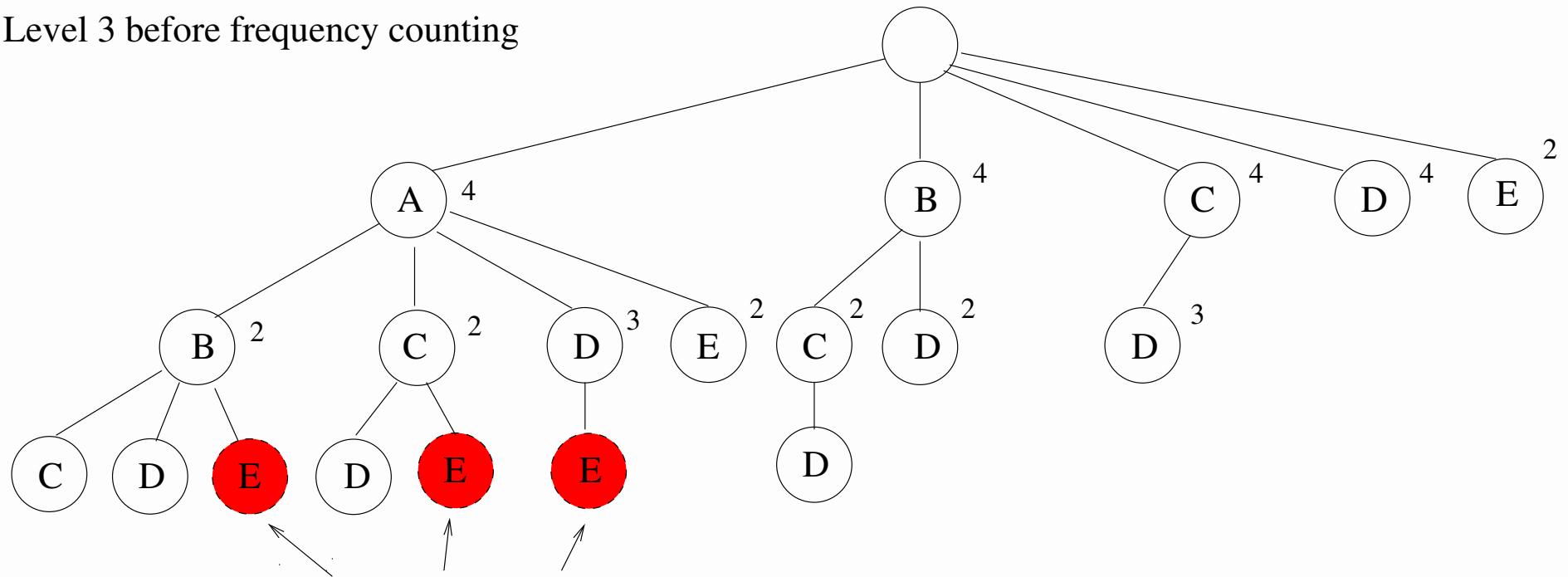
Possible 3-candidates (no pruning yet)



Simulation: $i = 3$ Candidate generation

Prune with monotonicity

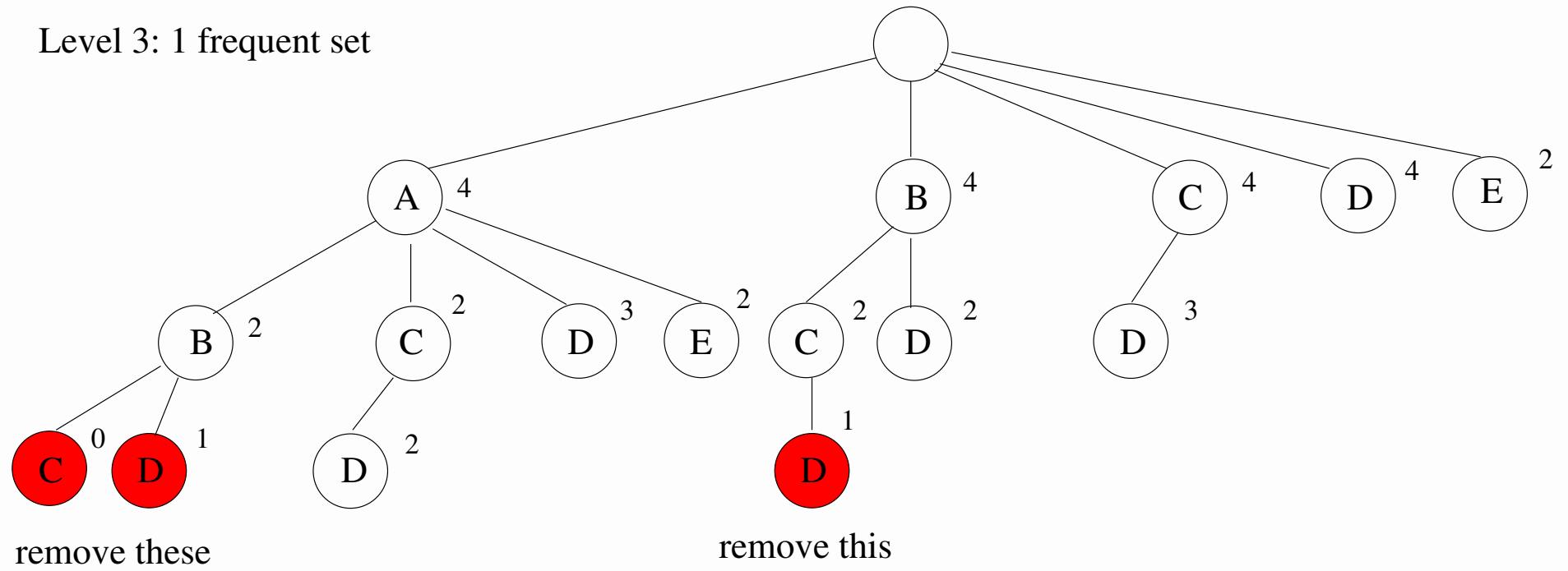
Level 3 before frequency counting



These are immediately removed because some parents not frequent

Simulation: $i = 3$ Frequency counting

Level 3: 1 frequent set



Simulation: $i = 4$ algorithm stops

Cannot create any 4-itemset candidates!

Return frequent sets (abs freq. in parenthesis):

A (4), B (4), C (4), D (4), E (2),
 AB (2), AC (2), AD (3), AE (2),
 BC (2), BD (2), CD (3), ACD (2)

6.3 Constructing rules from frequent sets

Given frequent sets $\mathcal{F} = \cup_i \mathcal{F}_i$. For all $\mathbf{X} \in \mathcal{F}$, $|\mathbf{X}| \geq 2$, and all $C \in \mathbf{X}$, evaluate $\mathbf{X} \setminus \{C\} \rightarrow C$

- confidence/precision: $\phi(\mathbf{X} \setminus \{C\} \rightarrow C) = P(C|\mathbf{X} \setminus \{C\})$
- statistical dependence:
 - lift $\gamma(\mathbf{X} \setminus \{C\} \rightarrow C) = \frac{P(\mathbf{X})}{P(\mathbf{X} \setminus \{C\})P(C)}$ or
 - leverage $\delta(\mathbf{X} \setminus \{C\} \rightarrow C) = P(\mathbf{X}) - P(\mathbf{X} \setminus \{C\})P(C)$
 - minimum requirement: $\gamma > 1$ or $\delta > 0$
- statistical significance p_F, χ^2 , mutual information, ...
- optional: prune out redundant/overfitted rules
Does $\mathbf{X} \setminus \{C\} \rightarrow C$ improve $\mathbf{Y} \setminus \{C\} \rightarrow C$ for all $\mathbf{Y} \subsetneq \mathbf{X}$?

Task: Construct rules from frequent sets

What are confident rules, if $\min_{cf} = 0.6$? Which of them express positive statistical dependence? ($n = 6$)

A (4), B (4), C (4), D (4), E (2),
 AB (2), AC (2), AD (3), AE (2),
 BC (2), BD (2), CD (3), ACD (2)

$$\phi(\mathbf{X} \rightarrow C) = \frac{P(\mathbf{X}, C)}{P(\mathbf{X})} \quad \gamma(\mathbf{X}, C) = \frac{P(\mathbf{X}, C)}{P(\mathbf{X})P(C)} = \frac{n \cdot fr(\mathbf{X}, C)}{fr(\mathbf{X})fr(C)} = \frac{P(C|\mathbf{X})}{P(C)}$$

Extra task (at home): What would be negative dependencies (form $\mathbf{X} \rightarrow \neg A_i$) with these constraints?

Task: Confident rules with $\min_{cf} = 0.6$

A (4), B (4), C (4), D (4), E (2), AB (2), AC (2), AD (3), AE (2),
 BC (2), BD (2), CD (3), ACD (2)

$$\phi(A \rightarrow B) = \frac{2}{4}$$

$$\phi(B \rightarrow A) = \frac{2}{4}$$

$$\phi(A \rightarrow C) = \frac{2}{4}$$

$$\phi(C \rightarrow A) = \frac{2}{4}$$

$$\phi(A \rightarrow D) = \frac{3}{4}$$

$$\phi(D \rightarrow A) = \frac{3}{4}$$

$$\phi(A \rightarrow E) = \frac{2}{4}$$

$$\phi(E \rightarrow A) = \frac{2}{2}$$

$$\phi(B \rightarrow C) = \frac{2}{4}$$

$$\phi(C \rightarrow B) = \frac{2}{4}$$

$$\phi(B \rightarrow D) = \frac{2}{4}$$

$$\phi(D \rightarrow B) = \frac{2}{4}$$

$$\phi(C \rightarrow D) = \frac{3}{4}$$

$$\phi(D \rightarrow C) = \frac{3}{4}$$

$$\phi(AC \rightarrow D) = \frac{2}{2}$$

$$\phi(AD \rightarrow C) = \frac{2}{3}$$

$$\phi(CD \rightarrow A) = \frac{2}{3}$$

Task: Confident rules with $\min_{cf} = 0.6$

rule	ϕ
$A \rightarrow D$ (or $D \rightarrow A$)	0.75
$E \rightarrow A$	1.00
$C \rightarrow D$ (or $D \rightarrow C$)	0.75
$AD \rightarrow C$	0.67
$CD \rightarrow A$	0.67
$AC \rightarrow D$	1.00

Note: $A \rightarrow D$ and $D \rightarrow A$ express the same association
(here also ϕ happens to be the same).

Task: Statistical dependence, $\gamma > 1$?

A (4), B (4), C (4), D (4), E (2), AB (2), AC (2), AD (3), AE (2),
 BC (2), BD (2), CD (3), ACD (2)

$$\gamma(\mathbf{X}, C) = \frac{n \cdot fr(\mathbf{X}, C)}{fr(\mathbf{X})fr(C)} = \frac{P(C|\mathbf{X})}{P(C)}$$

$$\gamma(A \rightarrow D) = \frac{6 \cdot 3}{4 \cdot 4} = 1.125$$

$$\gamma(E \rightarrow A) = \frac{6 \cdot 2}{2 \cdot 4} = 1.5$$

$$\gamma(C \rightarrow D) = \frac{6 \cdot 3}{4 \cdot 4} = 1.125$$

$$\gamma(AC \rightarrow D) = \frac{6 \cdot 2}{2 \cdot 4} = 1.5$$

$$\gamma(AD \rightarrow C) = \frac{6 \cdot 2}{3 \cdot 4} = 1$$

$$\gamma(CD \rightarrow A) = \frac{6 \cdot 2}{3 \cdot 4} = 1$$

Task: Statistical dependence, $\gamma > 1$?

rule	ϕ	γ	
$A \rightarrow D$ (or $D \rightarrow A$)	0.75	1.13	
$E \rightarrow A$	1.00	1.50	
$C \rightarrow D$ (or $D \rightarrow C$)	0.75	1.13	
$AD \rightarrow C$	0.67	1.00	→ prune out!
$CD \rightarrow A$	0.67	1.00	→ prune out!
$AC \rightarrow D$	1.00	1.50	

Strongest positive rules with lift:

oranges (E) \rightarrow milk (A)

milk (A), bread (C) \rightarrow cheese (D)

Many negative rules! e.g., $\gamma(ACD \rightarrow \neg B) = 3.0$

7. Pattern explosion and condensed representations

Pattern explosion a big problem!

- small min_{fr} \Rightarrow too many frequent patterns (worst case: $O(2^k)$ patterns!)
- but large min_{fr} not good (trivial patterns, interesting missed)

\Rightarrow condensed representations = **representatives** of all frequent sets

- faster to search
- all frequent sets can be derived from them (but **very costly!**)
- **bad shortcuts:** use only condensed representations \Rightarrow How to find statistical or significant associations??

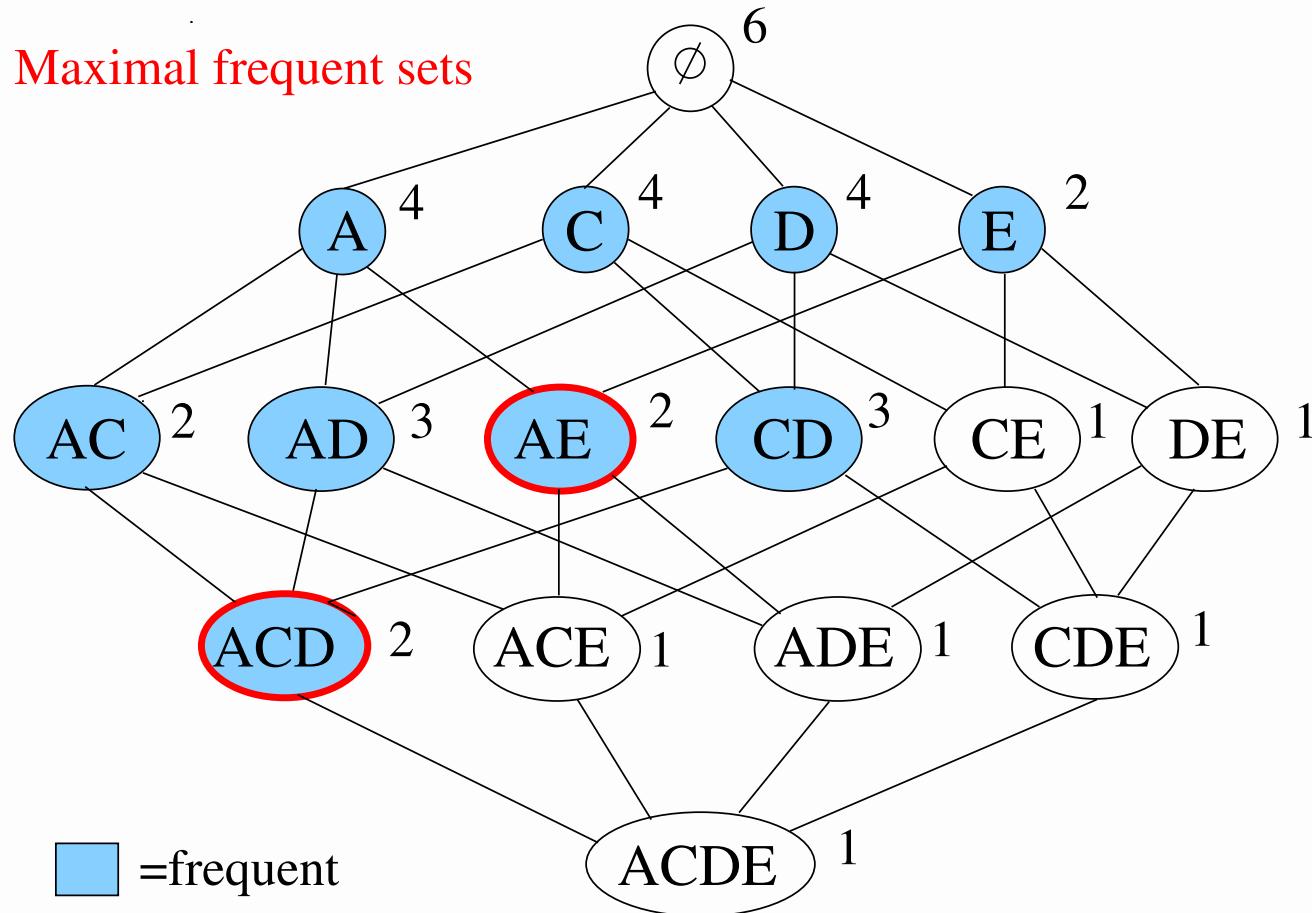
Maximal, closed and free sets

Frequent set \mathbf{X} , $P(\mathbf{X}) \geq min_{fr}$, is

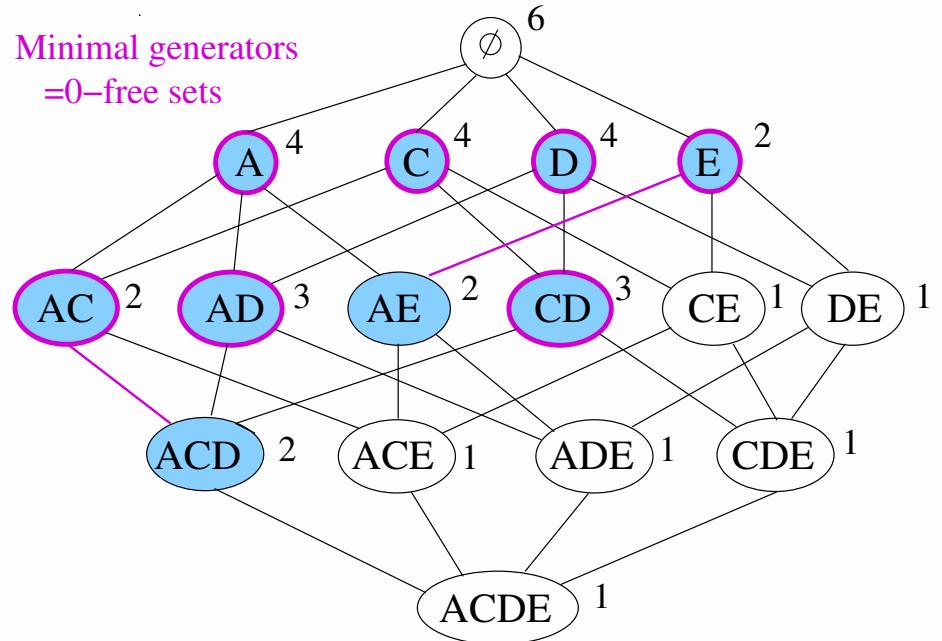
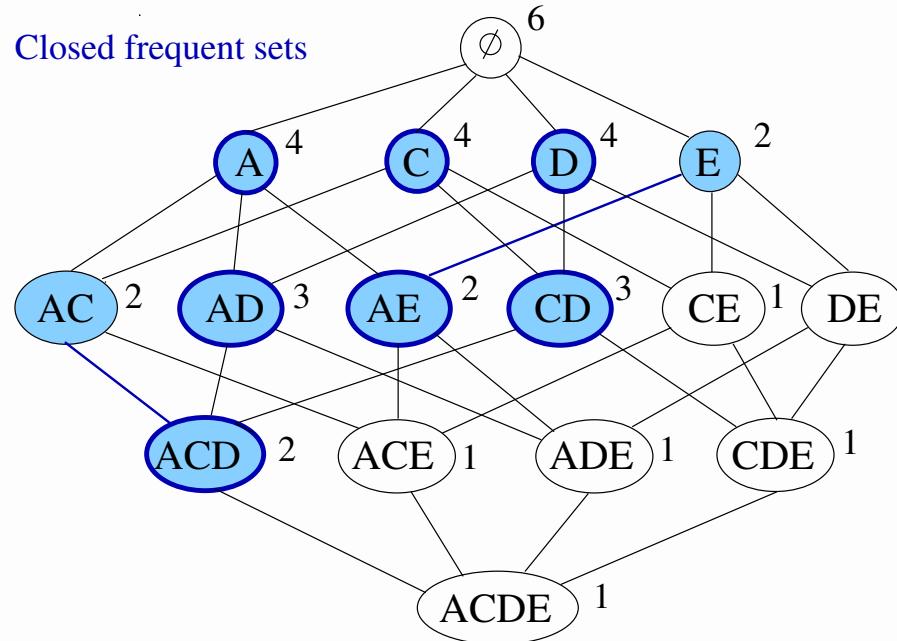
- **maximal frequent set**, if for all $\mathbf{Y} \supsetneq \mathbf{X}$: $P(\mathbf{Y}) < min_{fr}$ (most complex sets that are frequent)
- **closed set**, if for all $\mathbf{Y} \supsetneq \mathbf{X}$: $P(\mathbf{Y}) < P(\mathbf{X})$ (most specific set as a representative of **nested** sets with the same fr ; e.g., $fr(A) = fr(AC) = fr(ACD) \Rightarrow ACD$ closed)
- **0-free set = minimal generator**, if for all $\mathbf{Y} \subsetneq \mathbf{X}$: $P(\mathbf{Y}) > P(\mathbf{X})$ (most general set as a representative, e.g., A above)

Worst case: all sets closed and free!

Example: maximal sets ($\text{min}_{fr} = 2/6 = 0.33$)



Example: closed and 0-free sets



Summary

- **know what you find!** (and what you miss)
 - frequent rules \neq statistical associations
 - constraints and goodness measures
- exponential search space, but **very scalable algorithms**
- **monotonicity of frequency** very useful!
- dilemma: large \min_{fr} misses significant associations, but small \min_{fr} causes **pattern explosion**
- be **aware** when the algorithm uses **condensed representations!**

Question: How to utilize monotonicity when you search for statistical associations?

Further reading

- Aggarwal: Data mining – The textbook, Springer 2015, chapters 4–5.
- Leskovec et al.: Mining of Massive Datasets, Cambridge University Press 2014, chapter 6.
- Tan et al.: Introduction to Data Mining, Pearson, 2019, chapters 5–6.
- Hämäläinen and Webb: A tutorial on statistically sound pattern discovery. *Data Mining and Knowledge Discovery* 33(2):325-377, 2019.

<https://doi.org/10.1007/s10618-018-0590-x>

Other references

- Li and Zaiane: Exploiting statistically significant dependent rules for associative classification. *Intelligent Data Analysis*, 21(5):1155-1172, 2017.
- Webb and Vreeken: Efficient discovery of the most interesting associations. *Transactions on Knowledge Discovery from Data* 8(3):15:1-15:31, 2014.
- Zimek et al.: Frequent pattern mining algorithms for data clustering. Chapter 16 in *Frequent Pattern Mining*, 2014.
- Zimmermann and Nijssen: Supervised pattern mining and applications to classification. Chapter 17 in *Frequent Pattern Mining*, 2014.

Image sources

- Tan et al. slides for the book, 2018,
<https://www-users.cs.umn.edu/~kumar001/dmbook/>
- Animal pictures: https://commons.wikimedia.org/wiki/File:Dendrocopos_leucotos_NAUMANN.jpg and
<https://www.upmmetsa.fi/tietoa-ja-tapahtumia/tietoartikkelit/metsiemme-kovakuoriaisia/>

Mining association patterns (Part 2)

milk, cheese and bread
are often bought together

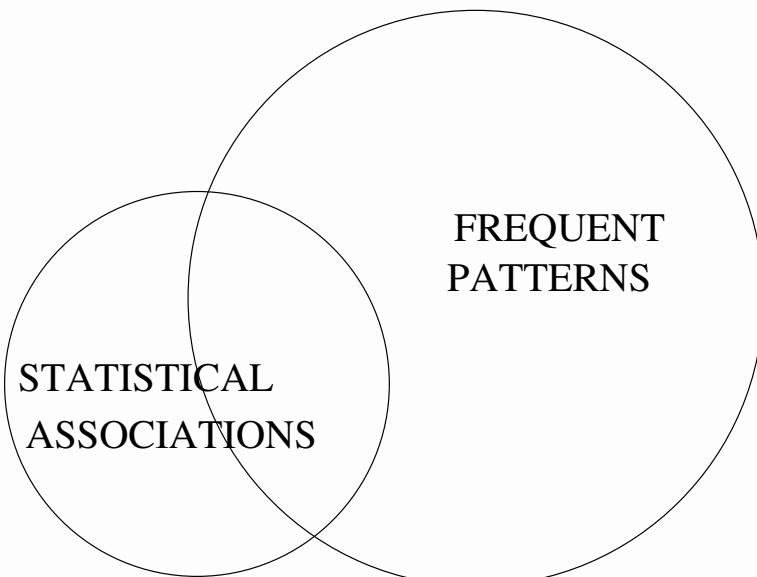
genes g1, g2, g3 and g4
are often over-expressed
in DLBC lymphomas

occurrence of certain insect species
makes it more likely to meet the
threatened white-backed woodpecker



How to find statistical association rules efficiently?

Problem: 2-step approach (frequents sets + postprocessing) very slow and often incomplete.



Recall experiments:

data	discovered %
Mushroom	100%
Chess	1%
T10I4D100K	100%
T40I10D100K	0%
Accidents	14%
Pumsb	27%
Retail	100%

Contents

1. Properties of statistical association rules
2. Pruning the search space
3. Algorithms

1. Properties of statistical association rules

Goal: Find association rules $\mathbf{X} \rightarrow C=c$ such that

1. rule expresses **statistical dependence**

$$P(\mathbf{X}, C=c) > P(\mathbf{X})P(C=c)$$

2. discovery is **statistically significant** (not spurious/due to chance) \Rightarrow likely to hold in future data
3. pattern is specialized only if improvement is statistically significant \Rightarrow **overfitting avoidance**
4. (optional) pattern is **not otherwise specious** (misleading)
 - prune out associations that definitely do not present any causal relationship

**Recall: Negative dependence between X and $C =$
positive dependence between X and $\neg C$**

	C	$\neg C$	Σ
X	$fr(\mathbf{X}C) =$ $fr(\mathbf{X})P(C) + n\delta$	$fr(\mathbf{X}\neg C) =$ $fr(\mathbf{X})P(\neg C) - n\delta$	$fr(\mathbf{X})$
$\neg X$	$fr(\neg \mathbf{X}C) =$ $fr(\neg \mathbf{X})P(C) - n\delta$	$fr(\neg \mathbf{X}\neg C) =$ $fr(\neg \mathbf{X})P(\neg C) + n\delta$	$fr(\neg \mathbf{X})$
Σ	$fr(C)$	$fr(\neg C)$	n

⇒ search rules of form $\mathbf{X} \rightarrow C$ or $\mathbf{X} \rightarrow \neg C$ expressing positive dependence between condition and consequent

$$\delta = \delta(\mathbf{X}, C) = P(\mathbf{X}, C) - P(\mathbf{X})P(C)$$

1.1 Could we use δ or γ for search?

1. Should we use δ or γ ?
 ⇒ variable-based or value-based interpretation?
2. Is high δ or γ enough to guarantee good patterns?
3. How do we perform the search?
 - statistical dependence is **not a monotonic property!**
 - $AB \rightarrow C$ can express significant dependence, even if $A \rightarrow B$, $A \rightarrow C$ and $B \rightarrow C$ expressed independence ^a

^aVoluntary home task: invent an example where this holds!

Value-based and variable-based interpretation of association $X \rightarrow C = c$

Let I_X be an indicator variable:

$I_X = 1$, if X holds, and $I_X = 0$, otherwise.

Important: Are we interested in association between values $I_X=1$ and $C=c$ or between binary variables I_X and C ?

⇒ different goodness measures and different results!

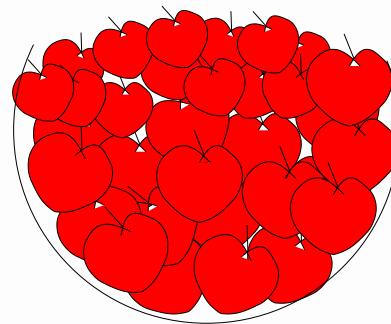
- **lift** γ measures strength of association between values
- **leverage** δ measures also strength of association between variables

Example: classifying apples by taste

Which rule should you choose? It depends whether you want a strong association between values or variables!

red → sweet

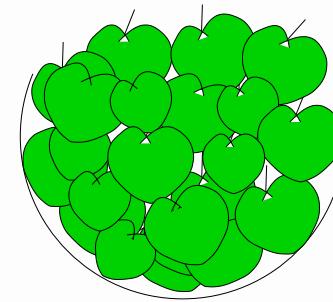
$\delta=0.22, \gamma=1.67$



Basket 1

60 red apples

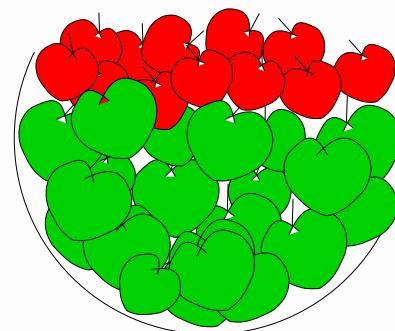
92% sweet



Basket 2

40 green apples

100% bitter



Basket 1

40 large red apples

100% sweet

Basket 2

40 green + 20 small red apples

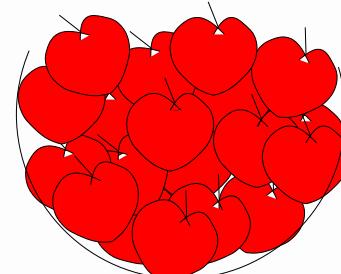
75% bitter

red ∧ big → sweet

$\delta=0.18, \gamma=1.82$

$n = 100$

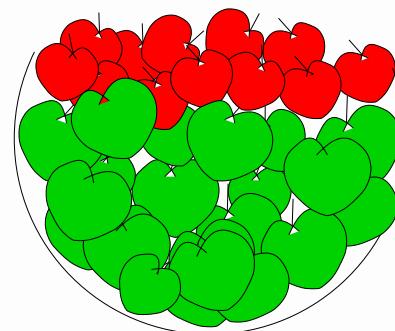
$fr(\text{sweet})=55$



Basket 1

40 large red apples

100% sweet



Is statistical dependence enough? Example

1000 students participated Mega Party. Data tells what each of them consumed in the party and what happened to them.

num	rule	fr_X	frc	fr_{XC}
1	<i>peppermint tea, sushi, chili sauce, sour cream → corona</i>	1	1	1
2	<i>vodka, sauerkraut, salmon → headache</i>	1	100	1
3	<i>cake → exam failure</i>	500	500	270
4	<i>magic mushrooms → intoxication</i>	20	20	20
5	<i>vodka → headache</i>	100	100	80
6	<i>vodka, salmon → headache</i>	40	100	30
7	<i>alcohol → exam failure</i>	333	500	300
8	<i>alcohol → cake</i>	333	500	200

Problem: High lift can be misleading

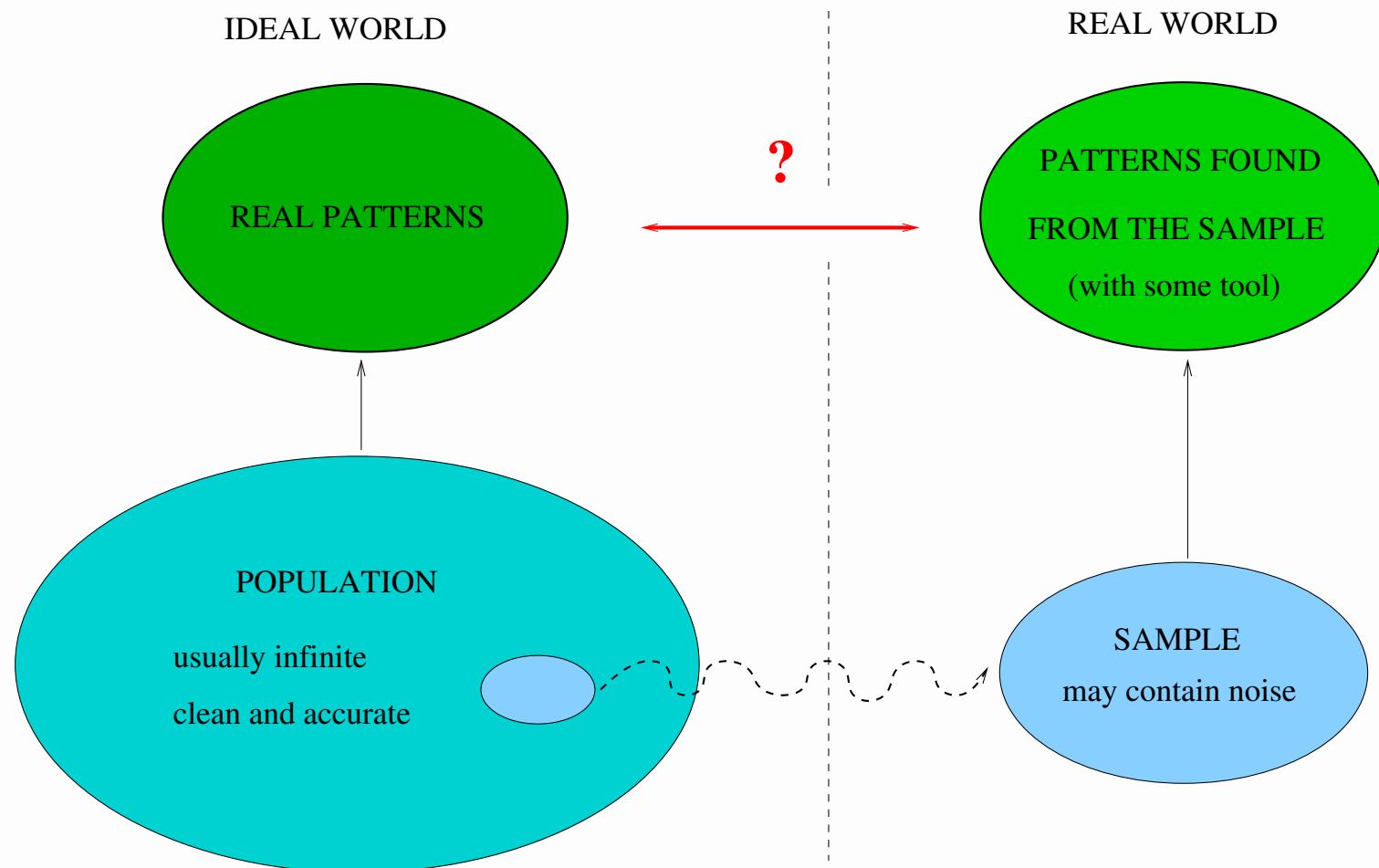
- a) 1 student got **corona** ($=C$)! The student was the only one who had combination \mathbf{X} = **peppermint tea, sushi, chili sauce, sour cream**. $\gamma(\mathbf{X}, C) = \frac{P(\mathbf{X}C)}{P(\mathbf{X})P(C)} = \frac{1}{P(C)} = n$.
- b) 100 students had **headache** ($=C$), including one with unique combination \mathbf{X} = **vodka, sauerkraut, salmon**.
 $\gamma(\mathbf{X}, C) = \frac{1}{P(C)} = \frac{1}{0.1} = 10$.
- Lift favors rare rules: $\gamma(\mathbf{X}, C) \leq \frac{1}{\max\{P(\mathbf{X}), P(C)\}}$

Problem: Leverage does not tell significance

Which rule is more significant?

- a) 500 students had chocolate cake (\mathbf{X}) and 500 failed the next day exam (C), including 270 cake eaters.
$$\delta(\mathbf{X}, C) = \frac{1}{n^2}(n \cdot 270 - 500 \cdot 500) = 0.02.$$
 - b) 20 students tried magic mushrooms (\mathbf{X}) and only them got a serious intoxication (C).
$$\delta(\mathbf{X}, C) = \frac{1}{n^2}(n \cdot 20 - 20 \cdot 20) = 0.0196.$$
- $\delta(\mathbf{X}, C) \leq \min\{P(\mathbf{X})P(\neg C), P(\neg \mathbf{X})P(C)\}$
 - Leverage favors rules where $P(\mathbf{X}) \approx P(C) \approx 0.5$
- ⇒ What is the probability of observing such strong associations by chance?

1.2 Statistical significance: problem



Statistical significance

Idea: Given pattern, estimate probability of observing at least such a strong association, if the pattern actually expressed independence.

- if probability (p -value) is very small, the pattern is likely true
- many ways to estimate the probability
 - analytically (frequentist and Bayesian approaches)
 - empirically (randomization testing)

Further reading: Hämäläinen & Webb (2019): A tutorial on statistically sound pattern discovery, Section 2.3

Significance measures for rules $\mathbf{X} \rightarrow C=c$

1. Fisher's exact p -value:

$$p_F = \sum_{i=1}^J \frac{\binom{fr(\mathbf{X})}{fr(\mathbf{X}C=c)+i} \binom{fr(\neg\mathbf{X})}{fr(\neg\mathbf{X}C\neq c)+i}}{\binom{n}{fr(C=c)}},$$

where $J = \min\{fr(\mathbf{X}C\neq c), fr(\neg\mathbf{X}C=c)\}$

- very robust! use when possible
- often $\ln(p_F)$ more convenient in programs
- remember: small values good (for p_F and $\ln(p)$)

cake → exam failure $p_F = 6.8e-3$ ($\ln(p) = -5.0$)

mushrooms → intoxication $p_F = 3.0e-42$ ($\ln(p) = -95.6$)

Significance measures for rules $\mathbf{X} \rightarrow C=c$

2. Mutual information:

$$MI = \log \frac{P(\mathbf{X}C)^{P(\mathbf{X}C)} P(\mathbf{X}\neg C)^{P(\mathbf{X}\neg C)} P(\neg\mathbf{X}C)^{P(\neg\mathbf{X}C)} P(\neg\mathbf{X}\neg C)^{P(\neg\mathbf{X}\neg C)}}{P(\mathbf{X})^{P(\mathbf{X})} P(\neg\mathbf{X})^{P(\neg\mathbf{X})} P(C)^{P(C)} P(\neg C)^{P(\neg C)}}$$

- now large values good
- you can get p -values for G statistic: $G = 2n \cdot MI$ (base e) or $G = 2n \cdot MI / \log_2(e)$ (base 2)

3. χ^2 -measure:

$$\chi^2 = \frac{n(P(\mathbf{X}, C) - P(\mathbf{X})P(C))^2}{P(\mathbf{X})P(\neg\mathbf{X})P(C)P(\neg C)} = \frac{n\delta^2(\mathbf{X}, C)}{P(\mathbf{X})P(\neg\mathbf{X})P(C)P(\neg C)}$$

- you can look p -values from the χ^2 distribution
- sensitive to accuracy of assumptions

Multiple hypothesis testing problem

Problem: The more patterns we test, the more will pass significance tests by chance.

E.g., if threshold $\alpha = 0.05$ and we test 10 000 spurious patterns, then about 500 patterns will incorrectly pass the test.

- ⇒ p -values need to be very small in DM!
 - ⇒ Many strategies
-

Further reading: Hämäläinen & Webb (2019): A tutorial on statistically sound pattern discovery, Section 6

1.3 Problem: Overfitted rules can be misleading

- Since r5 vodka → headache strong, no surprise that r6 vodka, salmon → headache is strong!
- If you knew only r6, you might think that vodka alone is safe!
- Possible that salmon and headache are **conditionally independent** given vodka or salmon may even prevent vodka headache (=negative conditional dependence)

Q and C are **conditionally independent**, given $X \Leftrightarrow$

$$P(Q, C|X) = P(Q|X)P(C|X) \Leftrightarrow$$

$$P(X, Q, C) = P(X, Q)P(C|X)$$

Problem: Specious associations are misleading

What about r3 **cake → failure** ($C \rightarrow F$)? ($fr = 270$)

- r7 **alcohol → failure** ($A \rightarrow F$) very strong, $P(F|A) = 0.9$ vs. $P(F|\neg A) = 0.3$
- If C independent of F given A , $P(F|CA) = 0.9$, so $CA \rightarrow F$ also strong (overfitted or redundant)
- If C independent of F given $\neg A$, $P(F|C\neg A) = 0.3$
- $fr(CA) = 200$ and $fr(C\neg A) = 300$
- Now expectation for $fr(CF) = fr(CAF) + fr(C\neg AF)$ is $fr(CA)P(F|A) + fr(C\neg A)P(F|\neg A) = 200 \cdot 0.9 + 300 \cdot 0.3 = 180 + 90 = 270$
- Rule **cake → failure** was just a side-product (specious)!

How to identify overfitted rules?

- $\mathbf{XQ} \rightarrow C$ can improve $\mathbf{X} \rightarrow C$ only, if $P(C|\mathbf{XQ}) > P(C|\mathbf{X})$
- If $P(C|\mathbf{XQ}) = P(C|\mathbf{X})$, then conditional independence
- What if $P(C|\mathbf{XQ})$ just slightly higher than $P(C|\mathbf{X})$?
(may be due to chance)
- ⇒ test **statistical significance of improvement!**
 - Assume conditional independence and estimate the probability of the observed improvement (or more)
 - ⇒ measures M_C for evaluating conditional dependence

When improvement is significant?

- **Value-based** associations: evaluate $M_C(\mathbf{XQ} \rightarrow C \mid \mathbf{X} \rightarrow C)$
- **Variable-based** associations more tricky!
- **problem:** adding \mathbf{Q} to $\mathbf{X} \rightarrow C$ may improve $P(C|X)$, but worsen $P(\neg C|\neg X)$
 - $P(\text{Sweet}|\text{Red}) = 0.92 < 1.0 = P(\text{Sweet}|\text{Red}, \text{Big})$ but
 $P(\neg \text{Sweet}|\neg \text{Red}) = 1.0 > 0.75 = P(\neg \text{Sweet}|\neg (\text{Red}, \text{Big}))$
- evaluate $M_C(\mathbf{XQ} \rightarrow C \mid \mathbf{X} \rightarrow C)$ and $M_C(\neg \mathbf{X} \rightarrow \neg C \mid \neg (\mathbf{XQ}) \rightarrow \neg C)$

M = significance measure for unconditional dependence
 M_C = corresponding measure for conditional dependence

Lessons to learn

1. Importance of **statistical significance**
 - significant discoveries are likely to hold in future data
 - remember multiple hypothesis testing problem
2. Decide whether you need **variable-based or value-based** definition
 - choose right measures and tools
3. Overfitted (too specialized) rules can be misleading ([vodka, salmon → headache](#))
 - but sometimes other types of rules may also be specious ([cake → exam failure](#))

2. Pruning the search space

Problem: Statistical association and significance are **not monotone** properties!

- given goodness measure M reflecting strength or significance of association,
$$M(\mathbf{X} \rightarrow C=c) = f(n, fr(\mathbf{X}C=c), fr(\mathbf{X}), fr(C=c))$$
- M is **increasing by goodness** (ibg) if high values good ($\delta, \gamma, \chi^2, MI$) or **decreasing by goodness** (dbg) if small values good ($p_F, \ln(p_F)$)
- if M ibg, we may have $M(\mathbf{XQ} \rightarrow C=c) > M(\mathbf{X} \rightarrow C=c)$
- **How to prune exponential search space?**

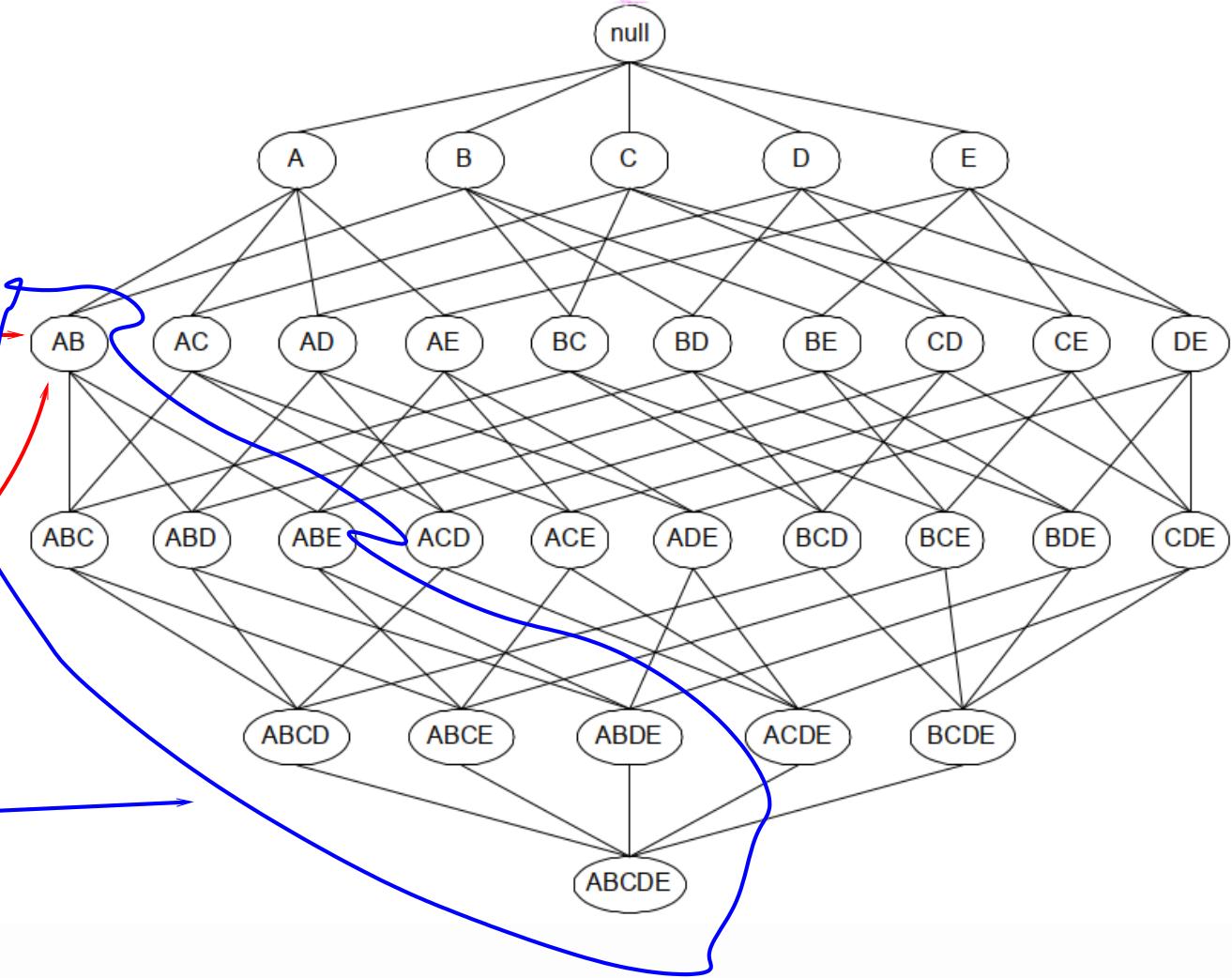
Basic idea: branch and bound search using monotonic upper or lower bounds for M

Idea: In set X evaluate monotonic upper or lower bounds for $M(XQ \rightarrow C)$

2 pruning rules:

1) find out here that consequent C cannot yield any good rules in AB or its extensions
→ disable consequent C here

2) find out here that no consequent can yield good rules in AB or its extensions
→ prune all sets here



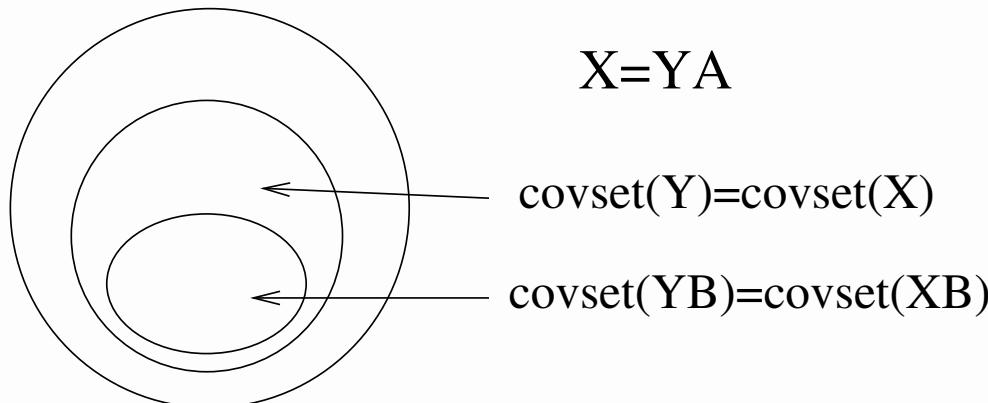
Pruning methods

- If M ibg, estimate upperbounds for $M(\mathbf{XQ} \rightarrow C=c)$ using information in \mathbf{X} (possibly $\mathbf{X} = \emptyset$) ($P(C)$ known)
 - e.g., upperbounds for δ :
 - i) $\delta(\mathbf{X} \rightarrow C) \leq P(C)P(\neg C)$ for any \mathbf{X}
 - ii) $\delta(\mathbf{XQ} \rightarrow C) \leq P(\mathbf{X})P(\neg C)$ for any \mathbf{Q}
 - iii) $\delta(\mathbf{XQ} \rightarrow C) \leq P(\mathbf{X}C)P(\neg C)$ for any \mathbf{Q}
 - If M is dbg, estimate lowerbounds
 - Search for only top- Q rules \Rightarrow update threshold \max_M or \min_M when new good rules found
 - Estimate in advance which rules would be overfitted and can be ignored

Pruning methods

- Utilize minimality condition: If $X = Y \cup \{A\}$ and $P(A=a|Y) = 1.0$ (i.e., $P(YA=a) = P(Y)$), then
 - i) A and $\neg A$ can be ignored in all XQ
rule $YQ \rightarrow A=a$ redundant vs. $Y \rightarrow A=a$
 - ii) all $B=b$, $B \notin X$, can be ignored in all XQ
e.g., $X \rightarrow B$ redundant vs. $Y \rightarrow B$

$\text{covset}(A)$ = transactions covered by A



In the figure, $\text{covset}(Y) \subseteq \text{covset}(A)$ since $P(A|Y) = 1$

3. Algorithms for statistical association rules

- **Magnum Opus** ^a: search for sufficiently strong rules with γ or δ and test significance of improvement with a hypergeometric test
 - value-based: compares only $\mathbf{XQ} \rightarrow C$ vs. $\mathbf{X} \rightarrow C$
- **Kingfisher** ^b: search for the most significant rules with a significance measure (Fisher's p_F , χ^2 , MI , etc.)
 - variable-based: compares both $\mathbf{XQ} \rightarrow C$ vs. $\mathbf{X} \rightarrow C$ and $\neg(\mathbf{XQ}) \rightarrow \neg C$ vs. $\neg\mathbf{X} \rightarrow \neg C$

^aWebb 2005, Webb 2007

^bHämäläinen 2012

Algorithms for statistical classification rules

Now the consequence is fixed! Methods for finding best rules $\mathbf{X} \rightarrow C$ with

- χ^2 (Morishita and Sese, 2000 ^a and Nijssen and Kok, 2006)
- with any convex measure (like χ^2 , MI), when \mathbf{X} is closed (Nijssen et al., 2009)
- with measures of strength like δ and γ (Li, 2006)
- etc.

^aalso show that the problem of finding the best classification rule with χ^2 is NP -hard

Example: Kingfisher algorithm

Problem: Given a set of binary attributes $\mathbf{R} = \{A_1, \dots, A_k\}$. Search for the most significant positive and negative dependency rules $\mathbf{X} \rightarrow A=a$, where $\mathbf{X} \subsetneq \mathbf{R}$, $A \in \mathbf{R} \setminus \mathbf{X}$, $a \in \{0, 1\}$!

- significance measures: p_F , $\ln(p_F)$, χ^2 , MI , z -score
- in the following, let's use p_F
- rules are **non-redundant** (\mathbf{X} contains no extra attributes which do not improve the dependency):
 $\nexists \mathbf{Y} \subsetneq \mathbf{X}$ such that $p_F(\mathbf{Y} \rightarrow A=a) \leq p_F(\mathbf{X} \rightarrow A=a)$
 - remember: smaller p -values better

Algorithm: the main idea

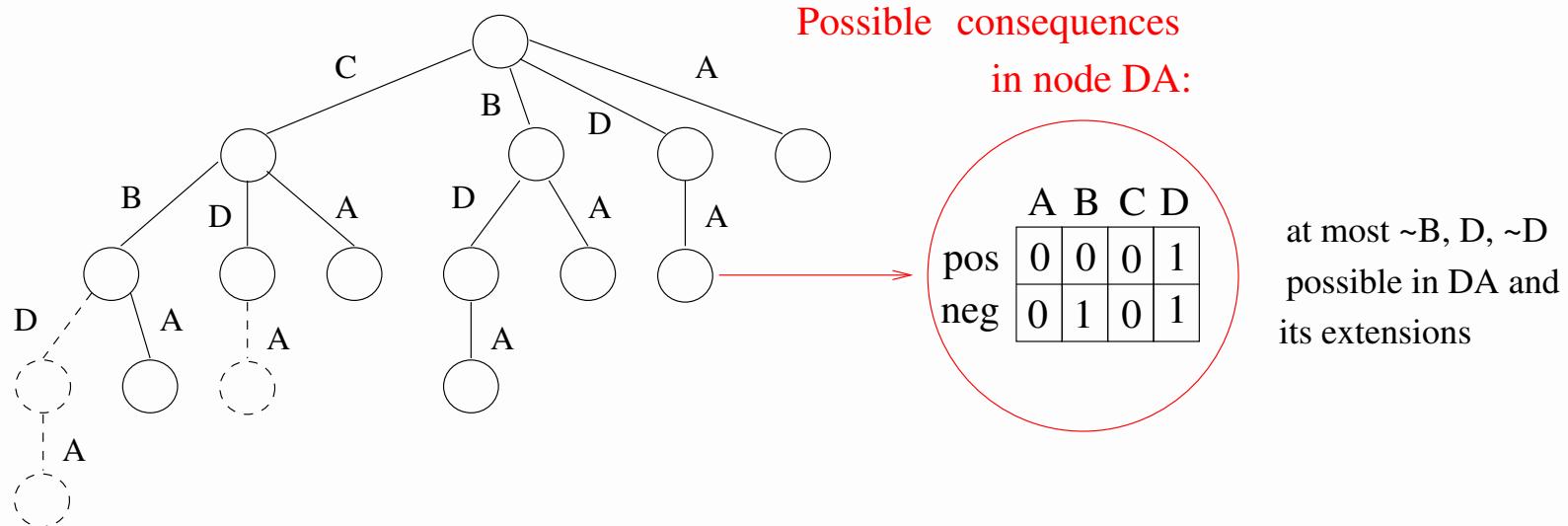
≈ Boosted branch-and-bound

- Generate an enumeration tree (breadth-first) and keep record on possible consequences at each node
- Consequence $A_i=a_i$ is set **impossible** in set **X**, if for all sets **Q**, rule $\mathbf{X} \setminus \{A_i\}\mathbf{Q} \rightarrow A_i=a_i$ is insignificant or redundant
- Consequence can be possible **only if it was possible in all parents** (monotonicity!)
- A node is pruned when no possible consequences left

Algorithm: the main idea

In each node

- bitvectors for possible positive and negative consequences
- numeric vector for best p_F -values of simpler rules
 - for checking redundancy



How to determine possible consequences?

At each node (set) \mathbf{X}

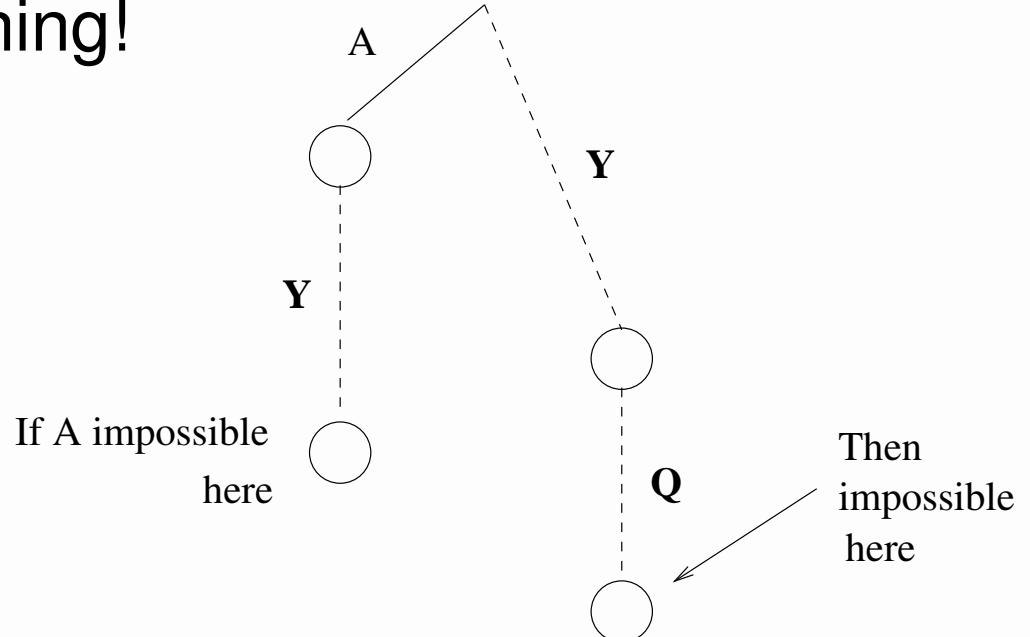
1. Initialization: combine parents' "possible" vectors (bit-and)
2. Updating: estimate lower bounds (LB) for $p_F(\mathbf{X} \setminus \{A_i\} \mathbf{Q} \rightarrow A_i = a_i)$ and decide if $A_i = a_i$ impossible in node \mathbf{X}
 - both for $A_i \in \mathbf{X}$ and $A_i \notin \mathbf{X}$
 - uses 3 different lower bounds $LB1$, $LB2$, $LB3$
 $LB1$: only $fr(A_i = a_i)$ known; $LB2$: $fr(A_i = a_i)$ and $fr(\mathbf{X})$ known, $A_i \notin \mathbf{X}$; $LB3$: $fr(A_i = a_i)$ and $fr(\mathbf{X})$ known, $A_i \in \mathbf{X}$
3. Utilize minimality condition

Algorithm: possible consequences

4. Lapis philosophorum principle (LP):

If $A=a$ impossible in X , then set it impossible in parent Y , $X = Y \cup \{A\}$, and all its descendants YQ

- Note: Y was already processed at previous level, but now updated again
- Very efficient pruning!



Extra: Simulation

Data

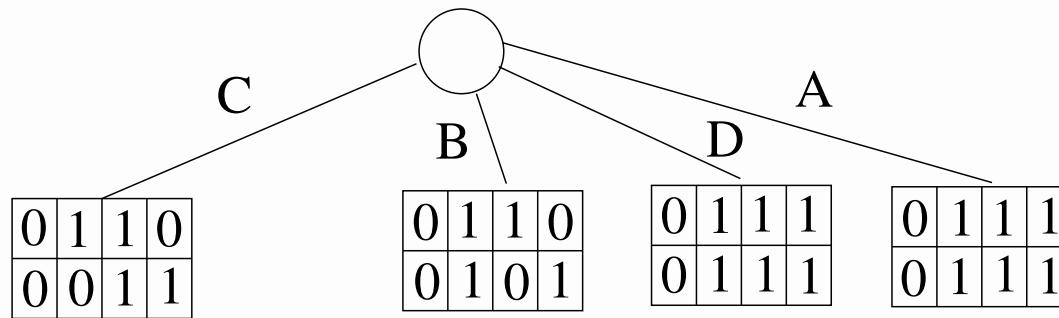
$$\mathbf{R} = \{A, B, C, D\}$$

$$n = 100$$

set	freq.
$ABC\neg D$	10
$A\neg B\neg CD$	85
$\neg AB\neg CD$	5

Search for the 10 best rules from the example data, when initial $p_{max} = 1.2 \cdot 10^{-8}$.

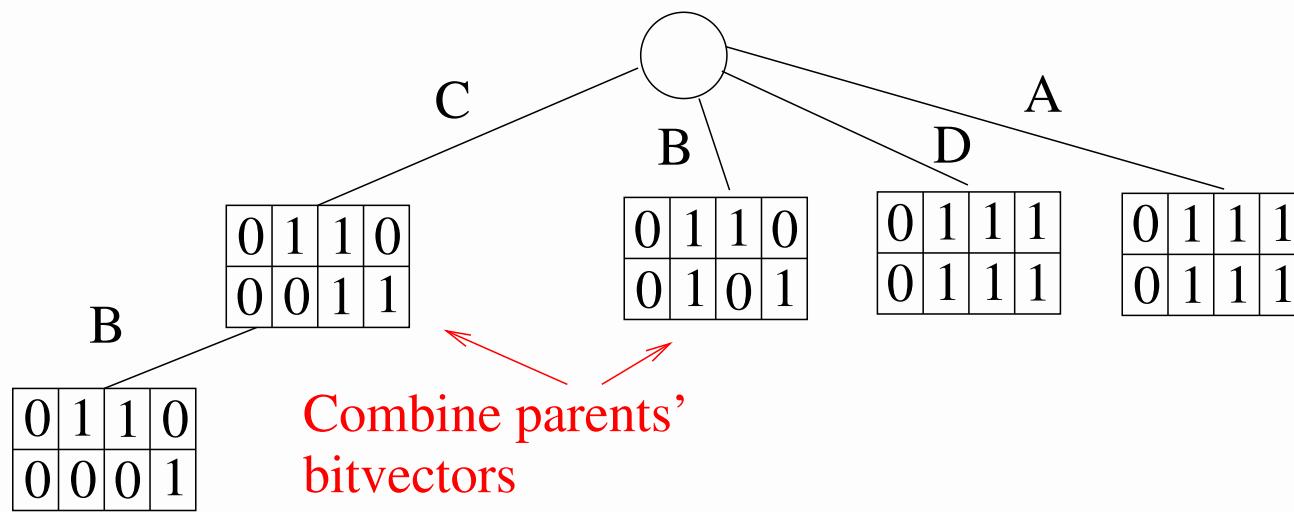
Simulation level 1: use LBs



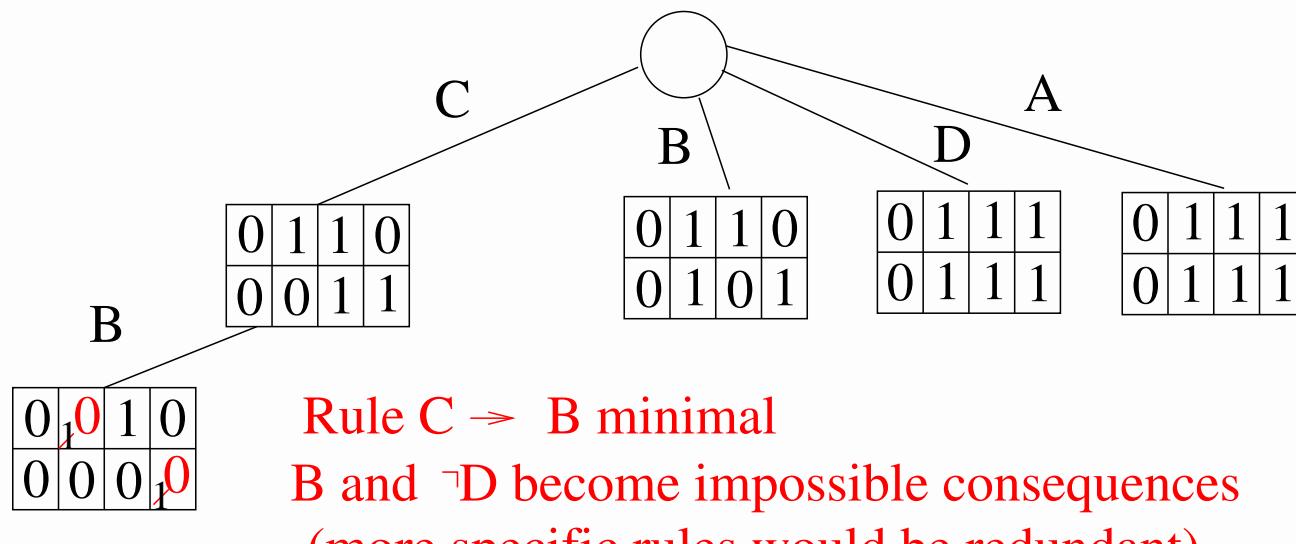
LB1: A and $\neg A$ are impossible consequences

Otherwise, possible consequences are determined by LB2

Simulation level 2: initialize CB

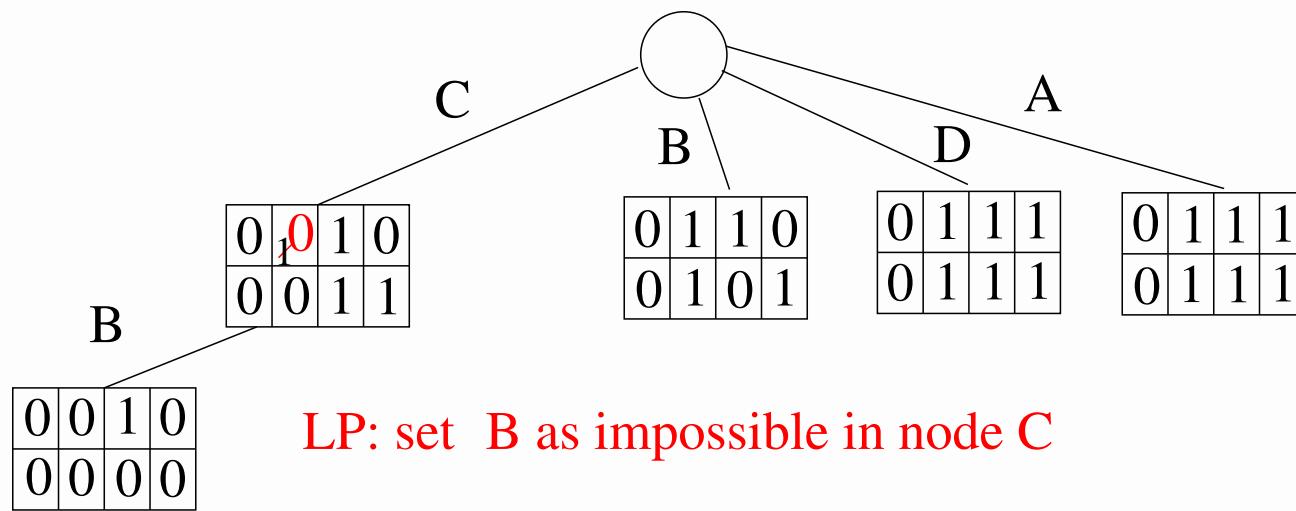


Simulation level 2: evaluate CB



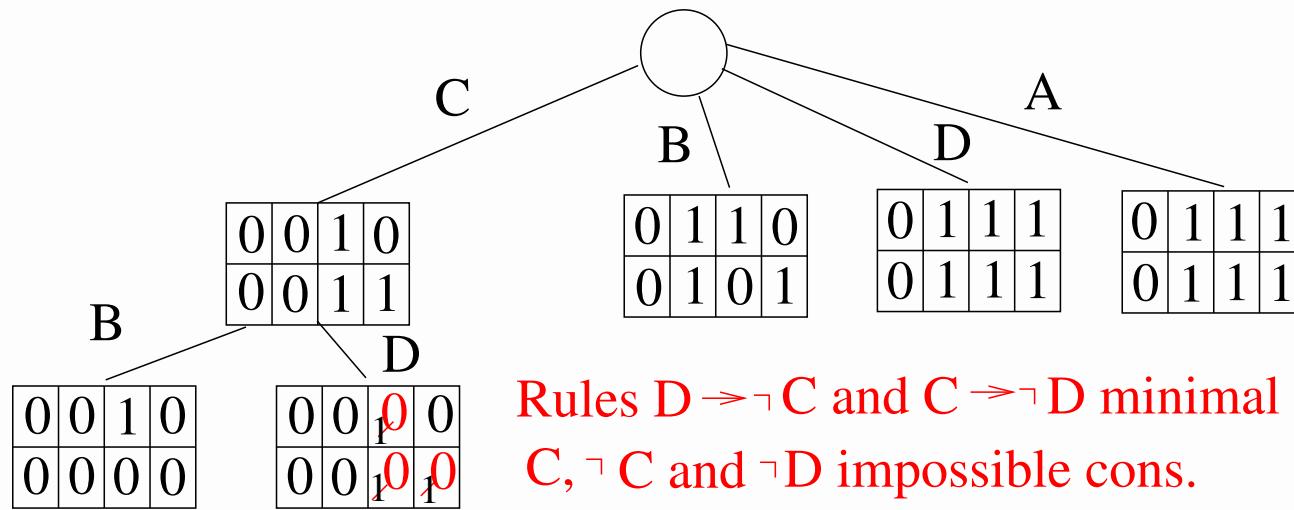
Rule	p
$C \rightarrow B$	$1.7 \cdot 10^{-10}$

Simulation level 2: utilize LP



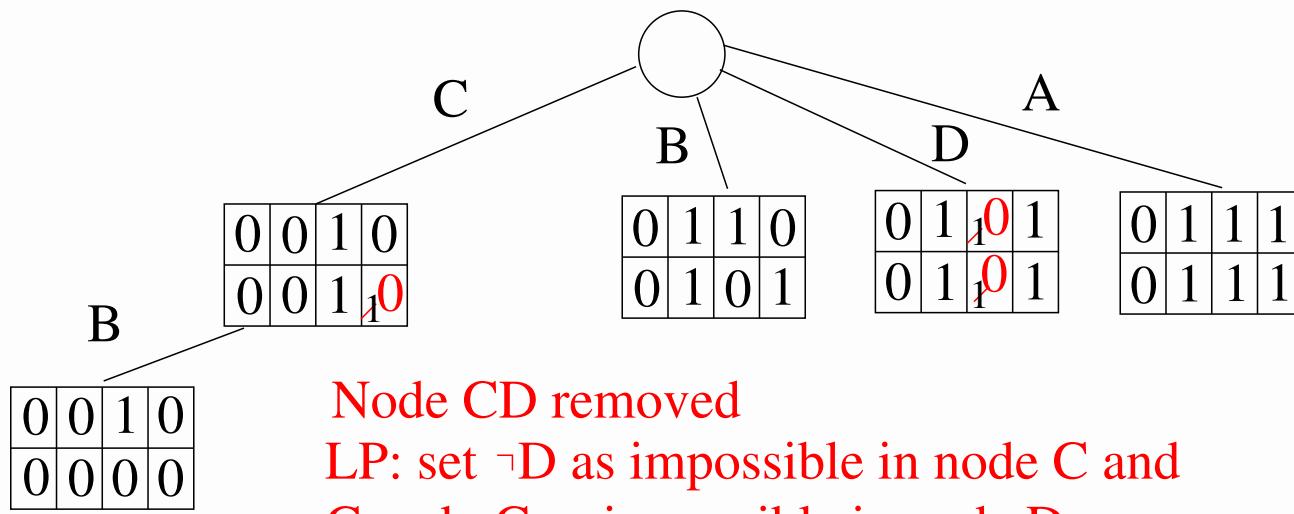
Rule	p
$C \rightarrow B$	$1.7 \cdot 10^{-10}$

Simulation level 2: create and evaluate CD



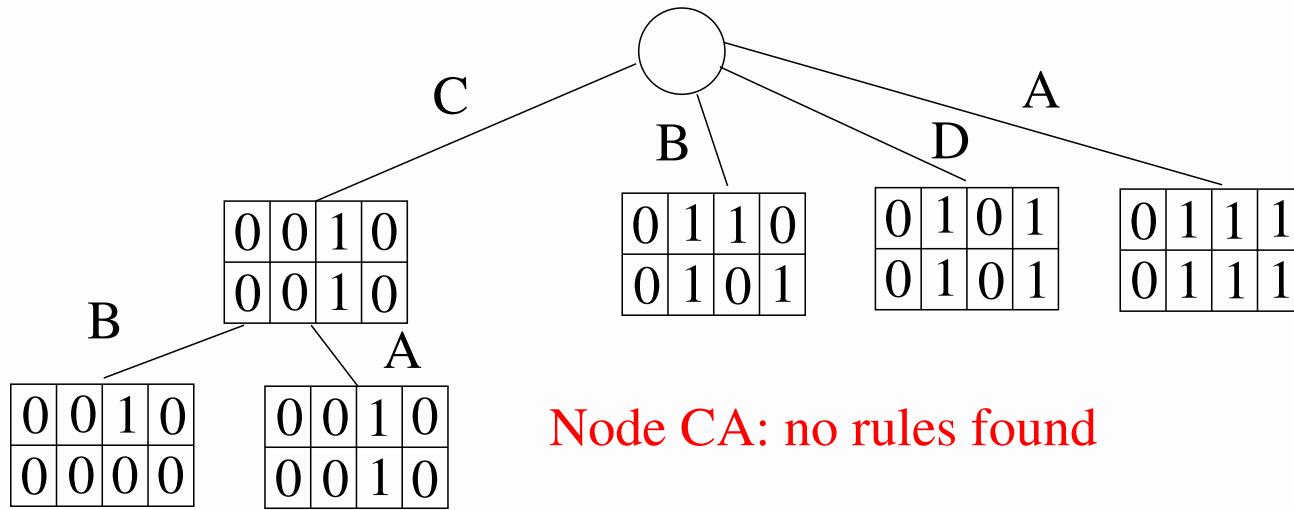
Rule	p
$D \rightarrow \neg C$	$5.8 \cdot 10^{-14}$
$C \rightarrow B$	$1.7 \cdot 10^{-10}$

Simulation level 2: utilize LP



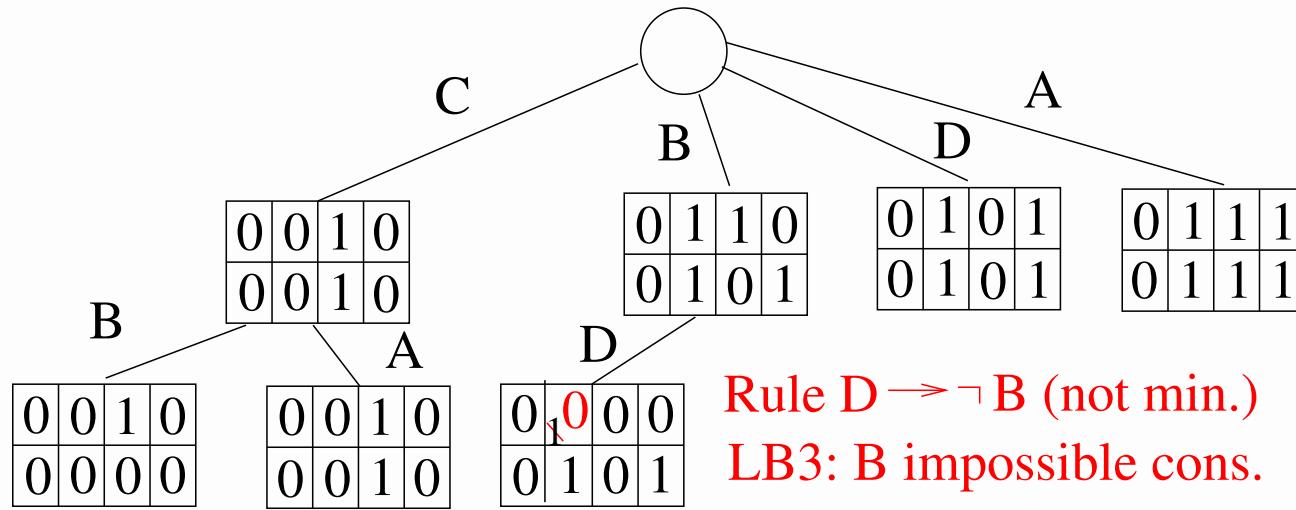
Rule	p
$D \rightarrow \neg C$	$5.8 \cdot 10^{-14}$
$C \rightarrow B$	$1.7 \cdot 10^{-10}$

Simulation level 2: evaluate CA



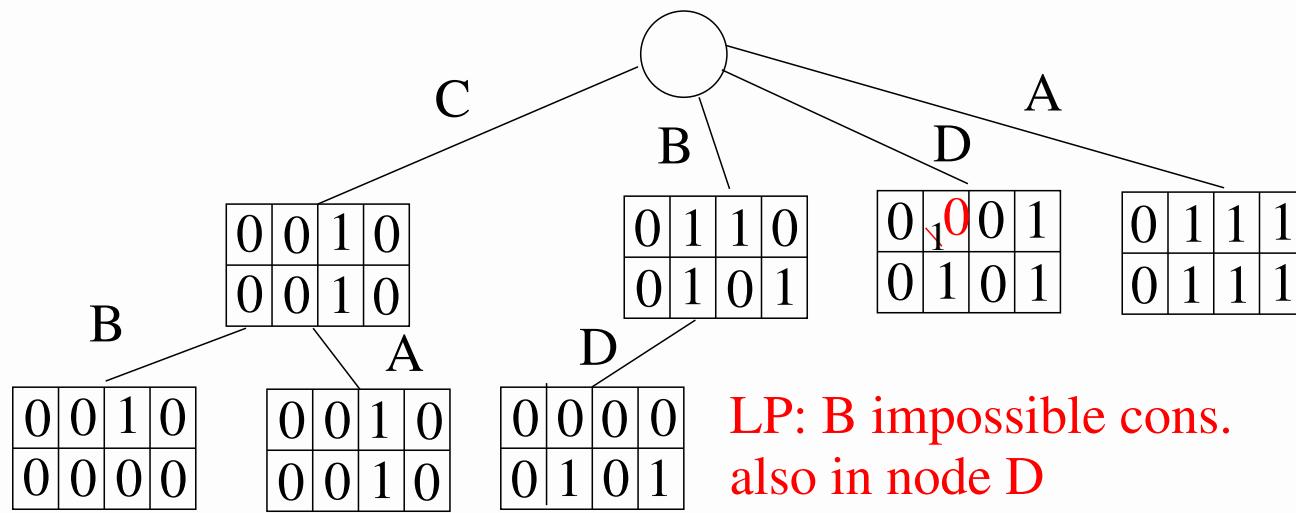
Rule	p
$D \rightarrow \neg C$	$5.8 \cdot 10^{-14}$
$C \rightarrow B$	$1.7 \cdot 10^{-10}$

Simulation level 2: evaluate BD



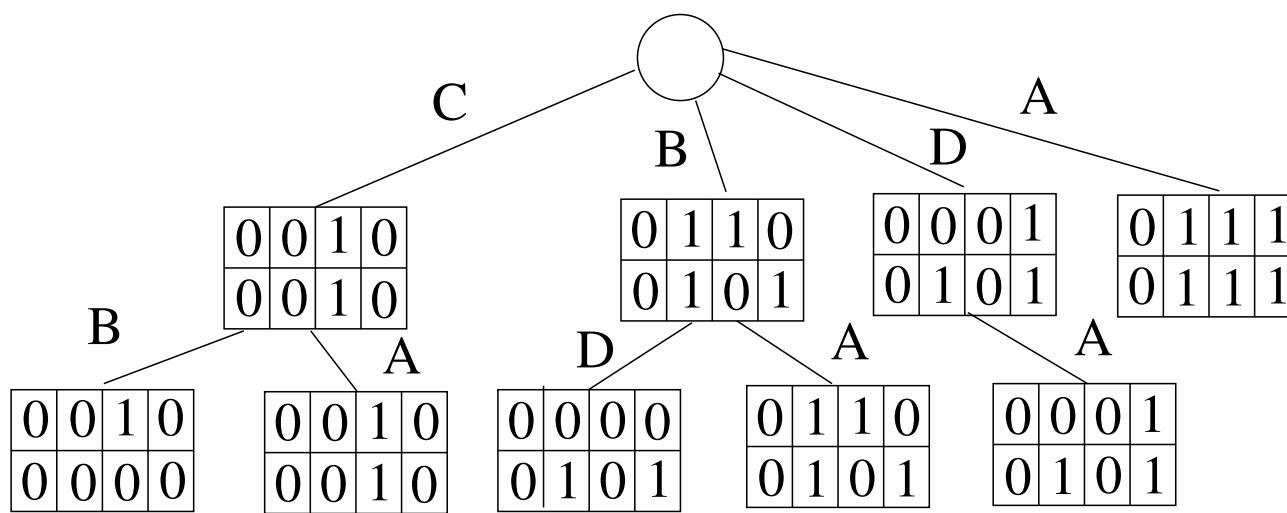
Rule	p
$D \rightarrow \neg C$	$5.8 \cdot 10^{-14}$
$C \rightarrow B$	$1.7 \cdot 10^{-10}$
$D \rightarrow \neg B$	$1.7 \cdot 10^{-10}$

Simulation level 2: utilize LP



Rule	p
$D \rightarrow \neg C$	$5.8 \cdot 10^{-14}$
$C \rightarrow B$	$1.7 \cdot 10^{-10}$
$D \rightarrow \neg B$	$1.7 \cdot 10^{-10}$

Simulation level 2: evaluate BA and DA

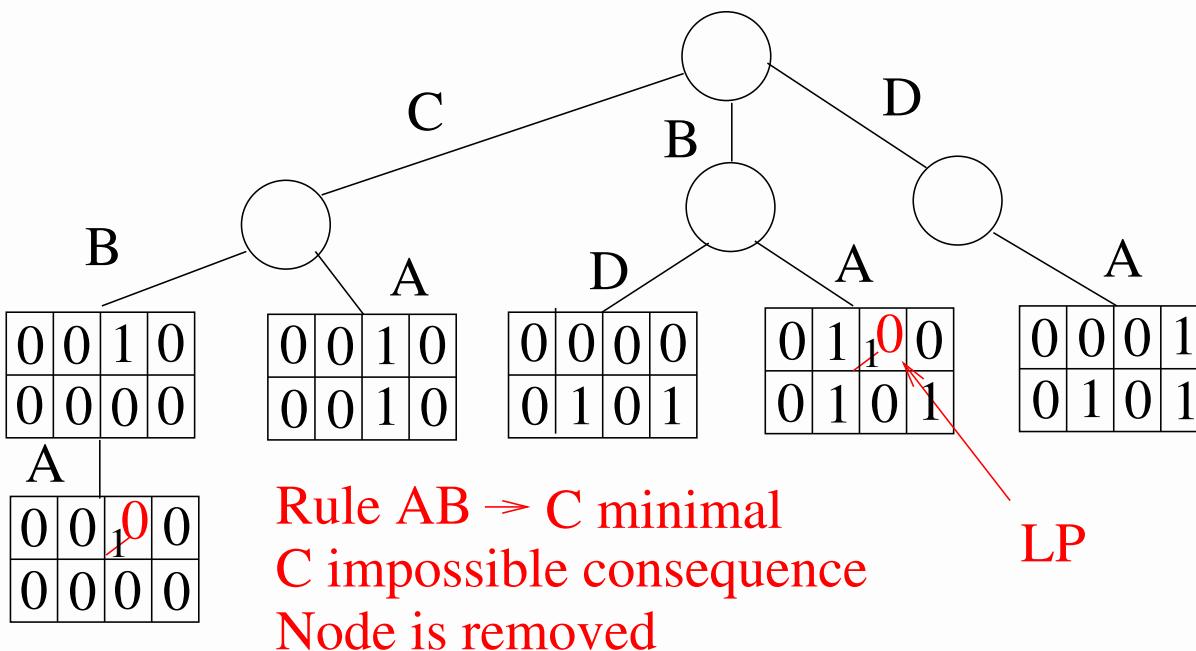


BA and DA: no rules found

Rule	p
$D \rightarrow \neg C$	$5.8 \cdot 10^{-14}$
$C \rightarrow B$	$1.7 \cdot 10^{-10}$
$D \rightarrow \neg B$	$1.7 \cdot 10^{-10}$

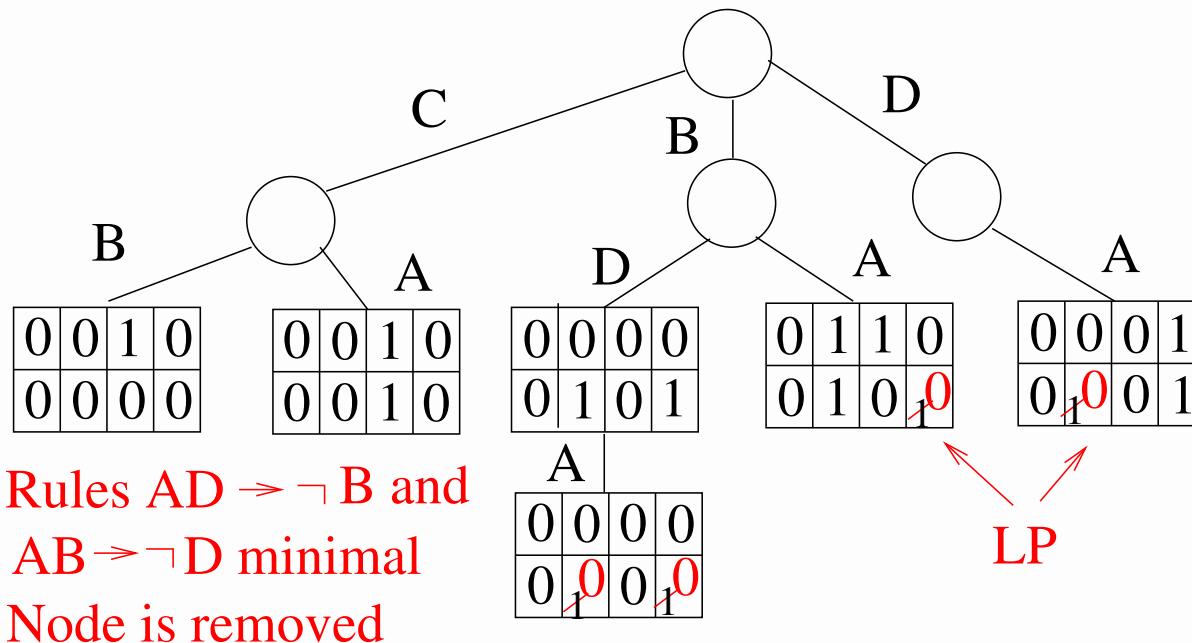
Note: Set A (1st level) no more needed.

Simulation level 3: evaluate CBA + use LP



Rule	p
$D \rightarrow \neg C$	$5.8 \cdot 10^{-14}$
$AB \rightarrow C$	$5.8 \cdot 10^{-14}$
$C \rightarrow B$	$1.7 \cdot 10^{-10}$
$D \rightarrow \neg B$	$1.7 \cdot 10^{-10}$

Simulation level 3: evaluate BDA + use LP



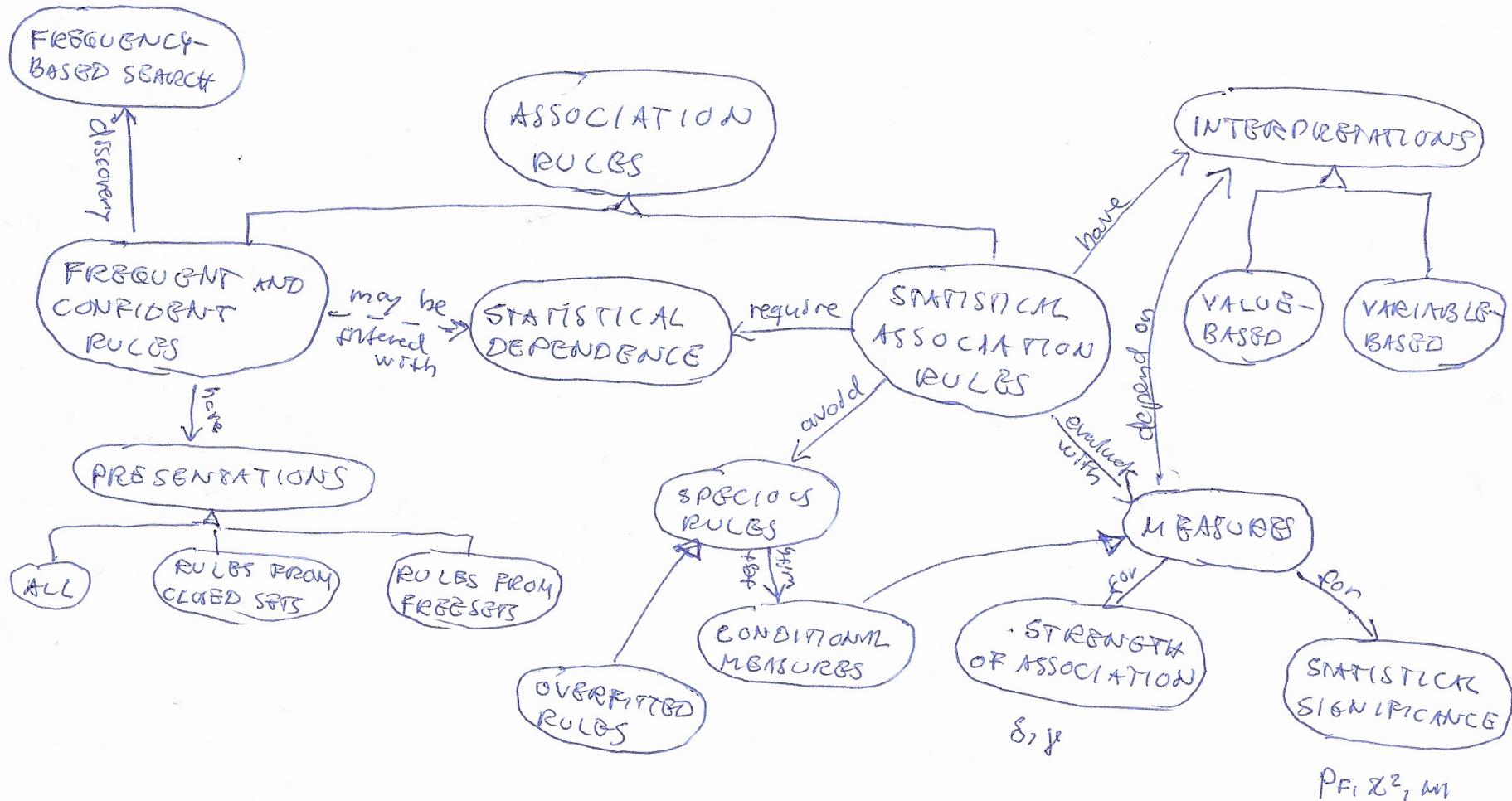
Rule	p
$AD \rightarrow \neg B$	$3.9 \cdot 10^{-18}$
$D \rightarrow \neg C$	$5.8 \cdot 10^{-14}$
$AB \rightarrow C$	$5.8 \cdot 10^{-14}$
$AB \rightarrow \neg D$	$5.8 \cdot 10^{-14}$
$C \rightarrow B$	$1.7 \cdot 10^{-10}$
$D \rightarrow \neg B$	$1.7 \cdot 10^{-10}$

DONE!

Summary

- If you want statistical associations, search them directly! (postprocessing frequent sets causes false positives & false negatives)
- statistical dependence **not monotonic** property (may be $AB \rightarrow C$ even if $A \perp\!\!\!\perp B$, $A \perp\!\!\!\perp C$, $B \perp\!\!\!\perp C$)
- **Secret:** $UB(M)$ or $LB(M)$ may behave monotonically!
 - $\forall Q$: if $UB(M(XQ \rightarrow C)|fr(c)) < min_M$,
 $UB(M(XQ \rightarrow C)|fr(c), fr(X)) < min_M$, or
 $UB(M(XQ \rightarrow C)|fr(c), fr(X), fr(XC)) < min_M$, $XQ \rightarrow C$ can be pruned! (here M ibg)
- Remember overfitted and other misleading rules!

Concept map of Association rules



Reading

- Hämäläinen and Webb: A tutorial on statistically sound pattern discovery. Data Mining and Knowledge Discovery 33(2):325-377, 2019. **Sections 3.1, 4.1, 4.4.**
- Properties of statistical association rules explained with the Mega Party example (in MyCourses)
- (Optional) Hämäläinen: Kingfisher: an efficient algorithm for searching for both positive and negative dependency rules with statistical significance measures, Knowledge and Information Systems 32: 383-414, 2012. **Sections 1-3, 4.1-4.2, the main idea from 5.1.**

Other references

- Li: On optimal rule discovery. IEEE Transactions on Knowledge and Data Engineering, 18(4):460-471, 2006.
- Lindley and Novick: The Role of Exchangeability in Inference, Annals of Statistics 9(1):45-58, 1981.
- Morishita and Sese: Traversing itemset lattices with statistical metric pruning. ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, 2000.
- Nijssen and Kok: Multi-class correlated pattern mining. 4th International Workshop on Knowledge Discovery in Inductive Databases, 2006.

Other references

- Nijssen, Guns, and De Raed. Correlated itemset mining in ROC space: a constraint programming approach. 15th ACM SIGKDD conf. Knowledge Discovery and Data Mining, 2009.
- Webb: Discovering significant patterns. Machine Learning, 68(1):1-33, 2007.
- Webb: Magnum Opus. Software, G. I. Webb & Associates, Melbourne, Australia.
- Webb and Zhang. K-optimal rule discovery. Data Mining and Knowledge Discovery, 10(1), 2005.

Mining association patterns (Part 3)

milk, cheese and bread
are often bought together

genes g1, g2, g3 and g4
are often over-expressed
in DLBC lymphomas

occurrence of certain insect species
makes it more likely to meet the
threatened white-backed woodpecker



Contents

1. Recap Apriori
2. Computational strategies and tricks
 - 2.1 enumeration tree and how to traverse it
 - 2.2 efficient frequency counting
3. Generic Apriori
4. Specious associations
(cake → exam failure)

1. Recap: *Apriori* algorithm (given \mathbf{R} , \mathcal{D} and min_{fr})

\mathcal{F}_i = frequent i -itemsets, C_i = candidate i -itemsets

$i=1$

$$\mathcal{F}_1 = \{A_i \in \mathbf{R} \mid P(A_i) \geq min_{fr}\}$$

while $\mathcal{F}_i \neq \emptyset$:

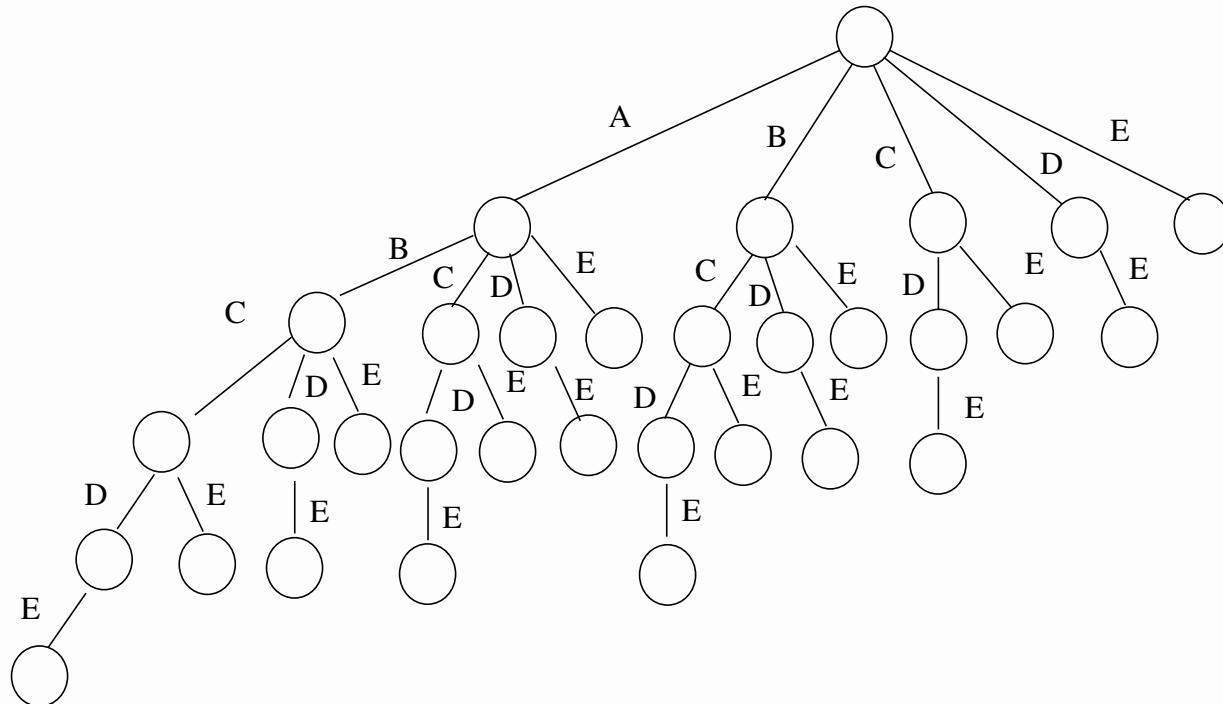
- Generate candidates C_{i+1} from \mathcal{F}_i
 - Prune $\mathbf{X} \in C_{i+1}$ if $\exists \mathbf{Y} \subsetneq \mathbf{X}, |\mathbf{Y}| = i, \mathbf{Y} \notin \mathcal{F}_i$
 - Count frequencies $fr(\mathbf{X}), \mathbf{X} \in C_{i+1}$
 - Set $\mathcal{F}_{i+1} = \{\mathbf{X} \in C_{i+1} \mid P(\mathbf{X}) \geq min_{fr}\}$
 - $i = i + 1$
- } (monotonicity)

Return $\cup_i \mathcal{F}_i$

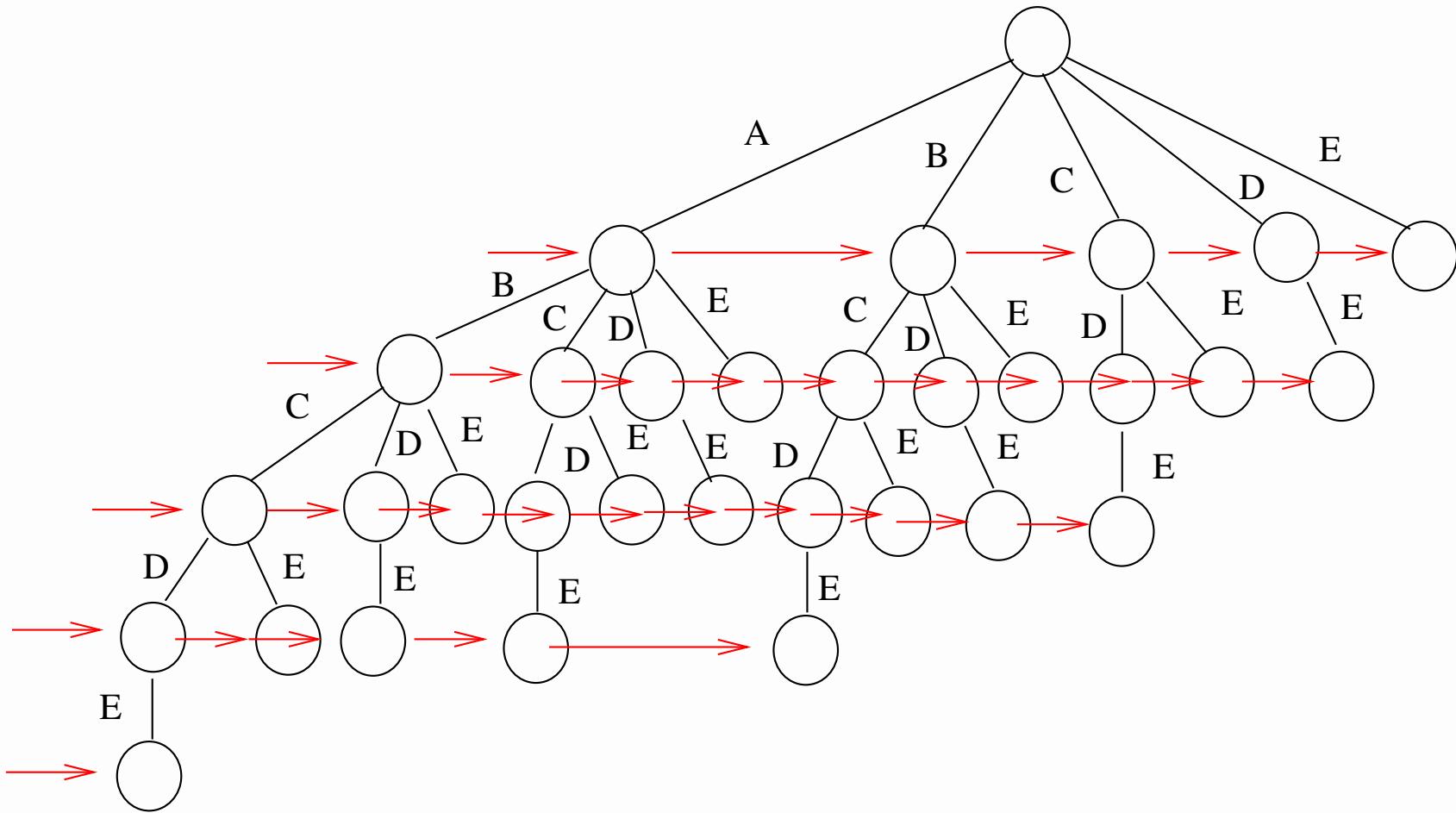
2.1 Enumeration tree can be large! ($\leq 2^k$ nodes, $k = |R|$)

Keep the tree as small as possible!

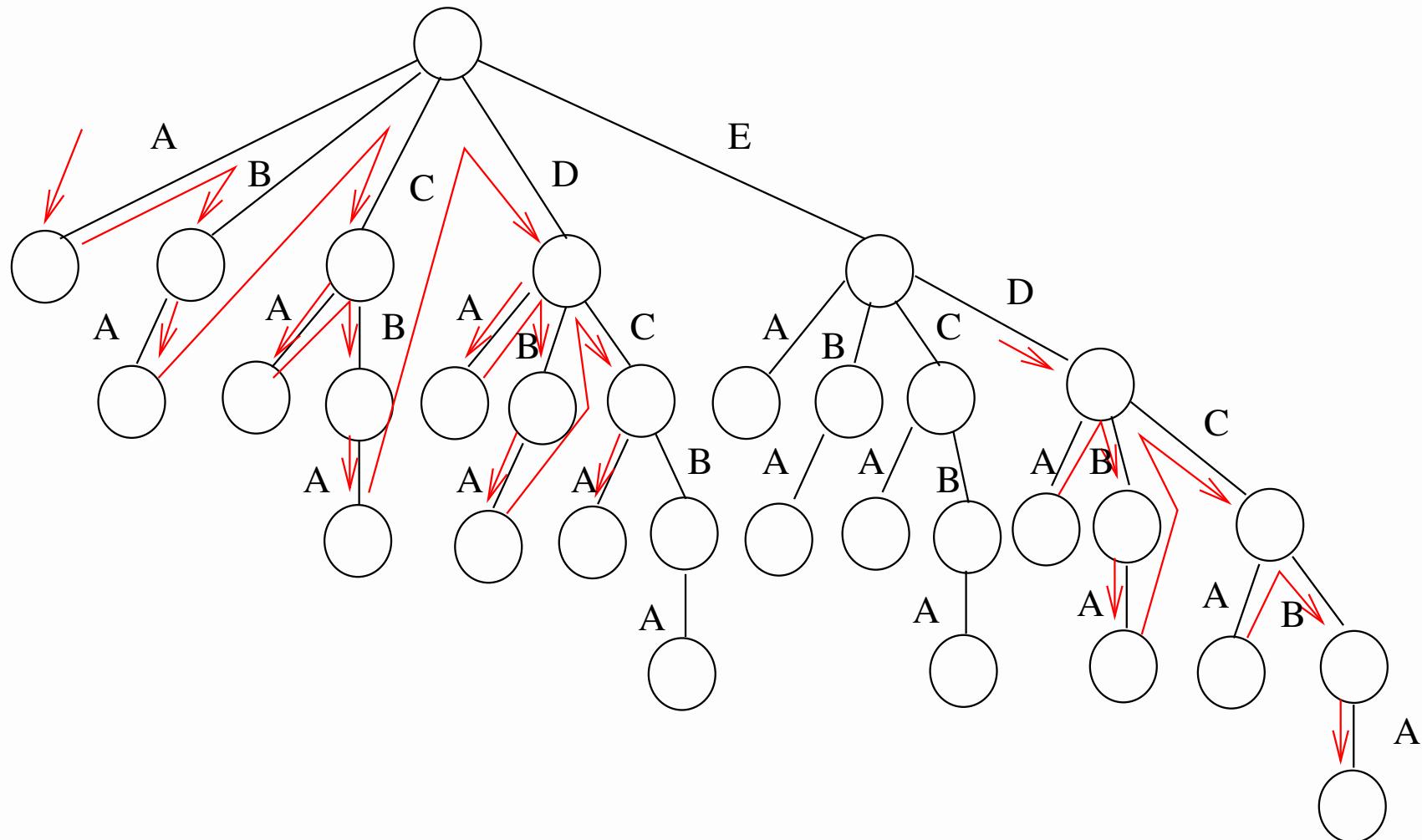
Trick: Order the main branches by ascending frequency
e.g., if $fr(A) \leq fr(B) \leq fr(C) \leq fr(D) \leq fr(E)$, put the largest branch under A :



Breadth-first traversal of the tree (like Apriori)

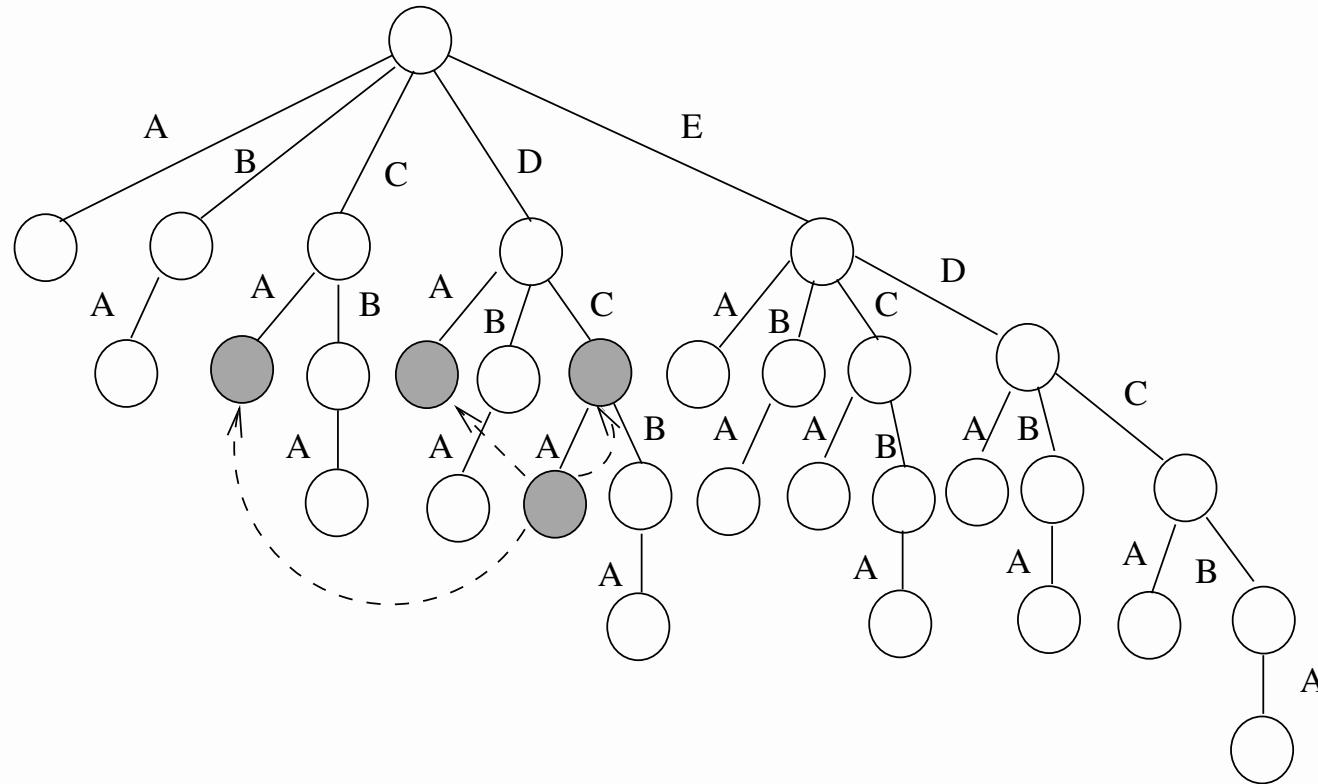


Depth-first traversal of the tree



Depth-first traversal of the tree

Construct the tree such that all parent sets are processed before child sets! e.g., parents of *DCA*:



2.2 Cost of frequency counting

- **breadth-first search:** frequencies of the entire level can be checked at once
- **depth-first search:** check frequency of each pattern before continuing ⇒ more database scans
- scanning database always costly!
- if enough memory, you can speed up frequency counting by **auxiliary structures** in the tree nodes, like
 - database projections
 - transaction id (tid) lists
- or construct a FP-tree

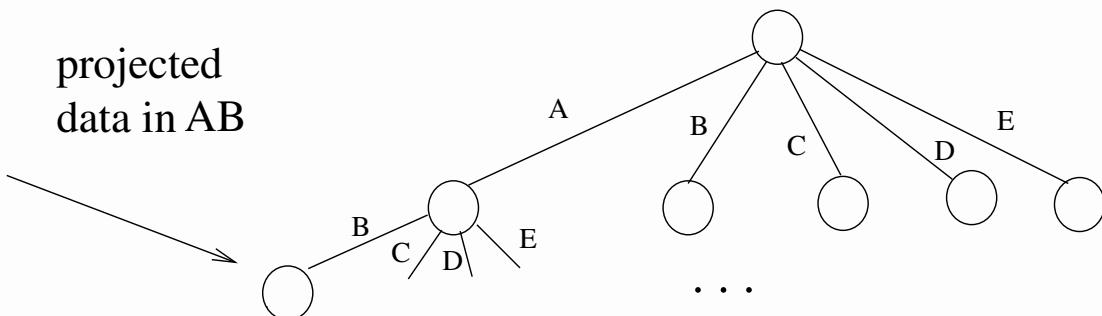
Database projections

Idea: Each node contains a projection of data onto the needed transactions and items.

- transactions covering **X** and items that can extend **X**
- child nodes inherit the projection and update it

	possible extensions		C	D	E
	A	B			
1	1	0	1	1	0
2	1	1	0	0	1
3	0	1	1	1	0
4	1	0	1	1	1
5	0	1	1	0	0
6	1	1	0	1	0

projected
data in AB



Tid lists (vertical counting methods)

Idea: Store into node **X** ids of transactions that cover **X**. When creating a new child, take an intersection of parents' tid lists.

- $tids(A) = \{1, 2, 4, 6\}$ and $tids(B) = \{2, 3, 5, 6\} \Rightarrow tids(AB) = \{2, 6\}$
- $tids(AB) = \{2, 6\}$ and $tids(AD) = \{1, 4, 6\} \Rightarrow tids(ABD) = \{6\}$
 - no need to intersect with $tids(BD)$
- can be implemented with bit-vectors \Rightarrow logical bit-and operation + count number of 1-bits

Extra: FP-tree

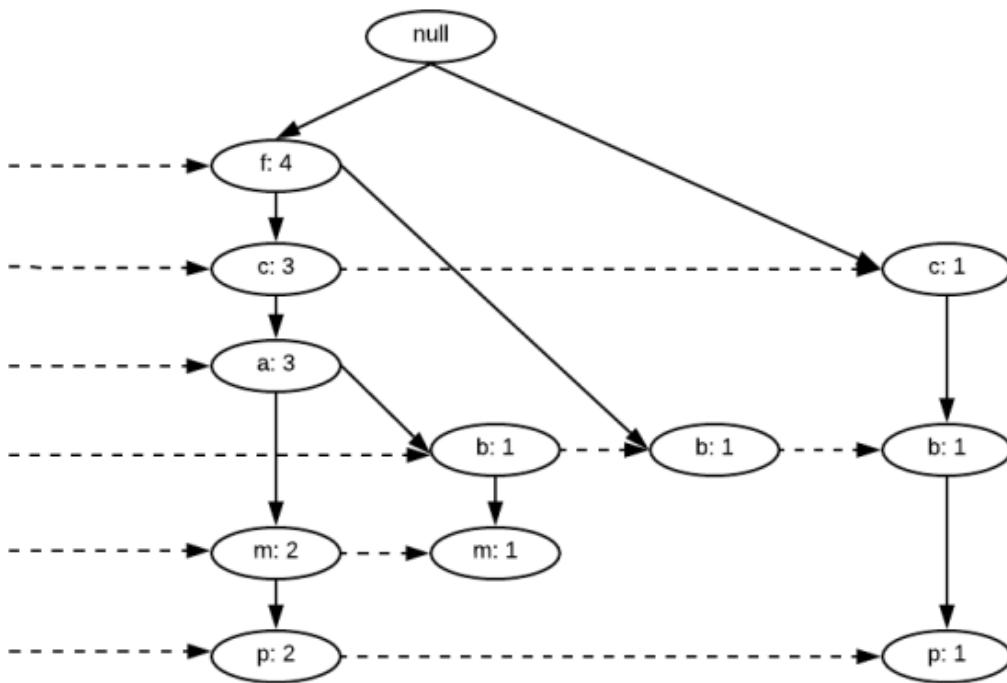
Idea: store data into a tree!

- present transaction database as a trie, FP-tree \mathcal{T}
- root–node path = prefix of a transaction
- each node contains a frequency counter = number of transactions having the prefix
- can be utilized in the FP-growth algorithm

See e.g., Aggarwal Sec 4.4.4

Extra: FP-tree example

Item	Frequency	Node Link
f	4	
c	4	
a	3	
b	3	
m	3	
p	3	



Transactions
f,c,a,m,p
f,c,a,b,m
f,b
c,b,p
f,c,a,m,p

(image from Jain 2018)

3. *Apriori* for other pattern types and properties?

Φ = monotonic property (like frequency)

α = pattern (like set, graph, sequence)

$\beta \subseteq \alpha$ = subpattern (subset, subgraph, subsequence)

$|\alpha|$ = complexity of α

i -pattern = pattern of complexity i

Idea: Begin from 1-patterns and search patterns incrementally utilizing monotonicity of Φ

Generic Apriori given monotonic property Φ

\mathcal{F}_i = i -patterns having property Φ

C_i = candidate i -patterns

i=1; \mathcal{F}_1 = {1-patterns with property Φ }

while $\mathcal{F}_i \neq \emptyset$

- Generate candidates C_{i+1} from \mathcal{F}_i
- Prune $\alpha \in C_{i+1}$ if $\exists \beta \subseteq \alpha, |\beta| = i, \beta \notin \mathcal{F}_i$
- Evaluate Φ for all $\alpha \in C_{i+1}$
- Set $\mathcal{F}_{i+1} = \{\alpha \in C_{i+1} \mid \alpha \text{ has property } \Phi\}$
- $i = i + 1$

Return $\cup_i \mathcal{F}_i$

4. Yule-Simpson's paradox and other specious associations

Statistical dependence is a necessary but not a sufficient condition of causal relation!

- Often a **majority** of dependencies are **specious** (illusory, spurious, apparent) associations
- e.g., **cake → exam failure** was a sideproduct of **alcohol → exam failure** and **alcohol → cake**
- Don't make too fast conclusions!

Introduction: Yule-Simpson's paradox

Example: Does a new treatment kill or cure? a

T =treatment, R =recovery

	R	$\neg R$	Σ
T	20	20	40
$\neg T$	16	24	40
Σ	36	44	80

$$P(R|T) = 0.50 > 0.475 = P(R)$$

positive association Treatment \rightarrow Recovery

^aLindley and Novick 1981

Let's analyze female and male separately

Among female patients (F)

	R	$\neg R$	Σ
T	2	8	10
$\neg T$	9	21	30
Σ	11	29	40

$P(R|T, F) = 0.20 < 0.275 = P(R|F)$ **negative association**

i.e., **Treatment $\rightarrow \neg$ Recovery**
in the female subgroup!

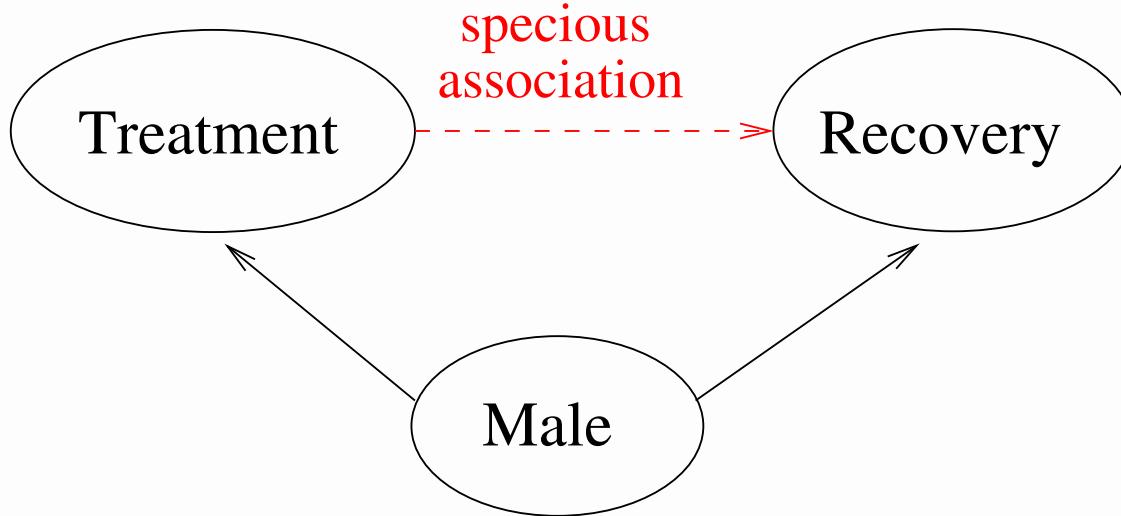
Among male patients (M)

	R	$\neg R$	Σ
T	18	12	30
$\neg T$	7	3	10
Σ	25	15	40

$P(R|T, M) = 0.60 < 0.625 = P(R|M)$ **negative association**

i.e., **Treatment $\rightarrow \neg$ Recovery**
in the male subgroup!

Explanation: a specious sideproduct of other rules



male → treatment $\phi = 0.75, \gamma = 1.50, \delta = 0.125, p_F = 7.44e-6$
male → recovery $\phi = 0.63, \gamma = 1.32, \delta = 0.088, p_F = 1.61e-3$

Expected freq. given $H_{01} : T \perp\!\!\!\perp R | M$ and $H_{02} : T \perp\!\!\!\perp R | F$ is
 $E(fr(TR) | H_{01}, H_{02}) = fr(TM)P(R|M) + fr(TF)P(R|F) =$
 $30 \cdot \frac{25}{40} + 10 \cdot \frac{11}{40} = 21.5 > 20 = fr(TR)$. Conditionally negative dependence!

Types of specious associations $Q \rightarrow C=c$

Here association **disappears or reverses its sign** when conditioned on some confounding factor X :

- **Yule-Simpson's paradox:** $Q \rightarrow C=c$ positive association, but Q and $C=c$ either conditionally independent or negatively dependent given X and given $\neg X$
- **Specious generalization:** some $QZ \rightarrow C=c$ completely explains $Q \rightarrow C=c$ ($X = QZ$)
- **Specious specialization:** some $X \rightarrow C=c$ completely explains $XZ \rightarrow C=c$ ($Q = XZ$)
- **Equivalence between Q and X or $\neg X$** (not specious per se)

Definition: Conditional leverage δ_c

Conditional leverage of $\mathbf{Q} \rightarrow C=c$ given \mathbf{X} or $\neg\mathbf{X}$

$$\delta_1 = \delta_c(\mathbf{Q}, C=c | \mathbf{X}) = P(\mathbf{X}, \mathbf{Q}, C=c) - P(\mathbf{X}, \mathbf{Q})P(C=c | \mathbf{X})$$

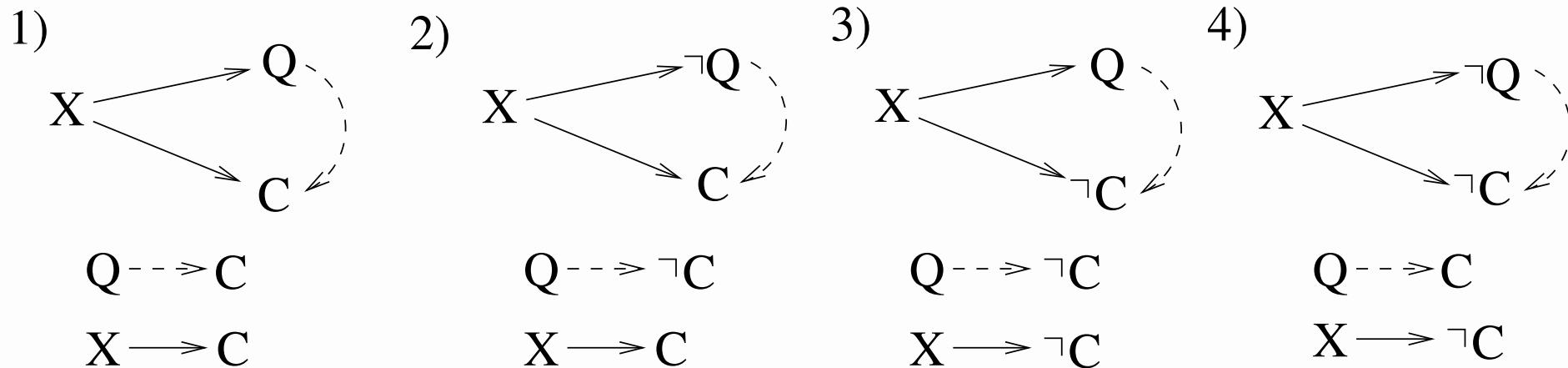
$$\delta_2 = \delta_c(\mathbf{Q}, C=c | \neg\mathbf{X}) = P(\neg\mathbf{X}, \mathbf{Q}, C=c) - P(\neg\mathbf{X}, \mathbf{Q})P(C=c | \neg\mathbf{X})$$

Recall: \mathbf{Q} and C are conditionally independent given \mathbf{X} if

$$P(\mathbf{X}\mathbf{Q}C) = P(\mathbf{X}\mathbf{Q})P(C|\mathbf{X})$$

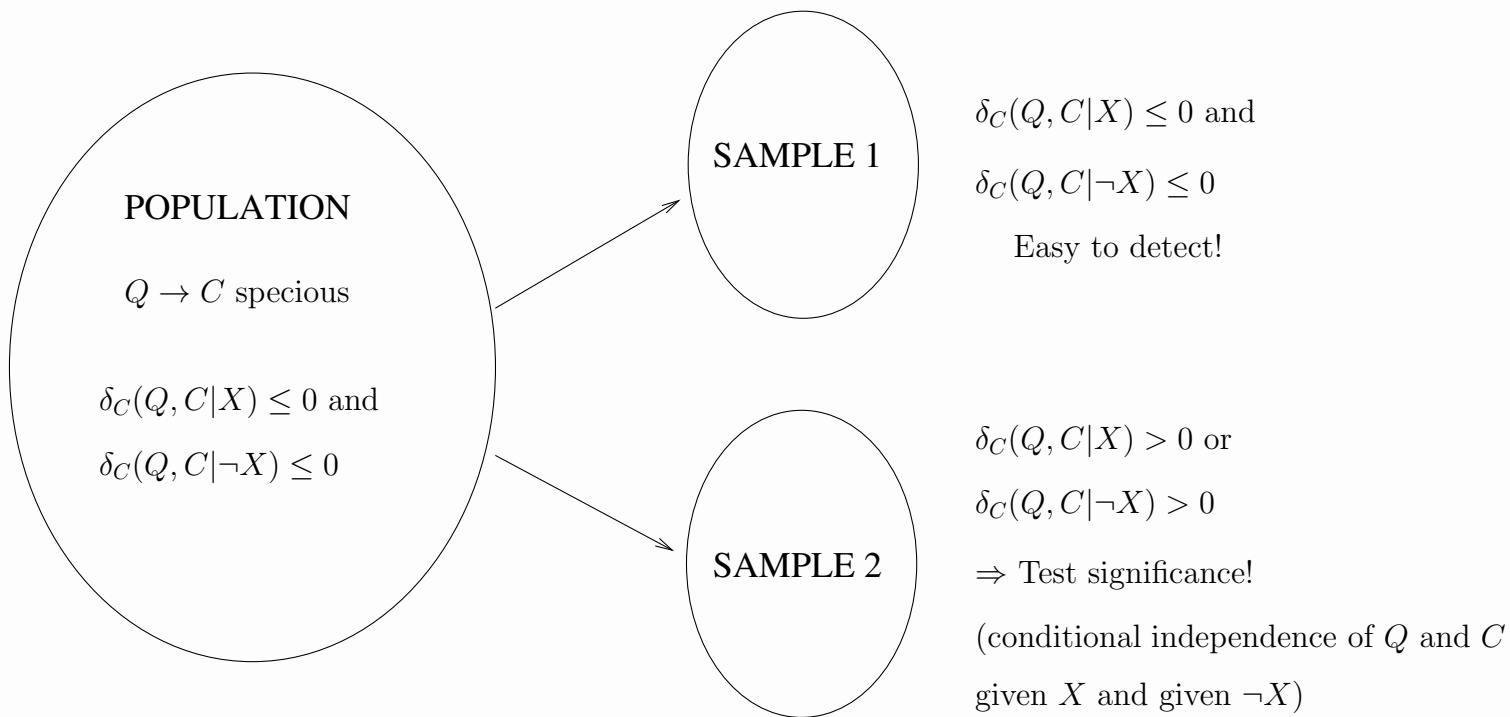
Definition: Specious association rule

Association rule $Q \rightarrow C=c$ ($c \in \{0, 1\}$) is specious if there is another rule $X \rightarrow C=c_x$ ($c_x \in \{0, 1\}$) such that $\delta_c(Q, C=c|X) \leq 0$ and $\delta_c(Q, C=c|\neg X) \leq 0$ in the **population**.



Detecting speciousness in the sample

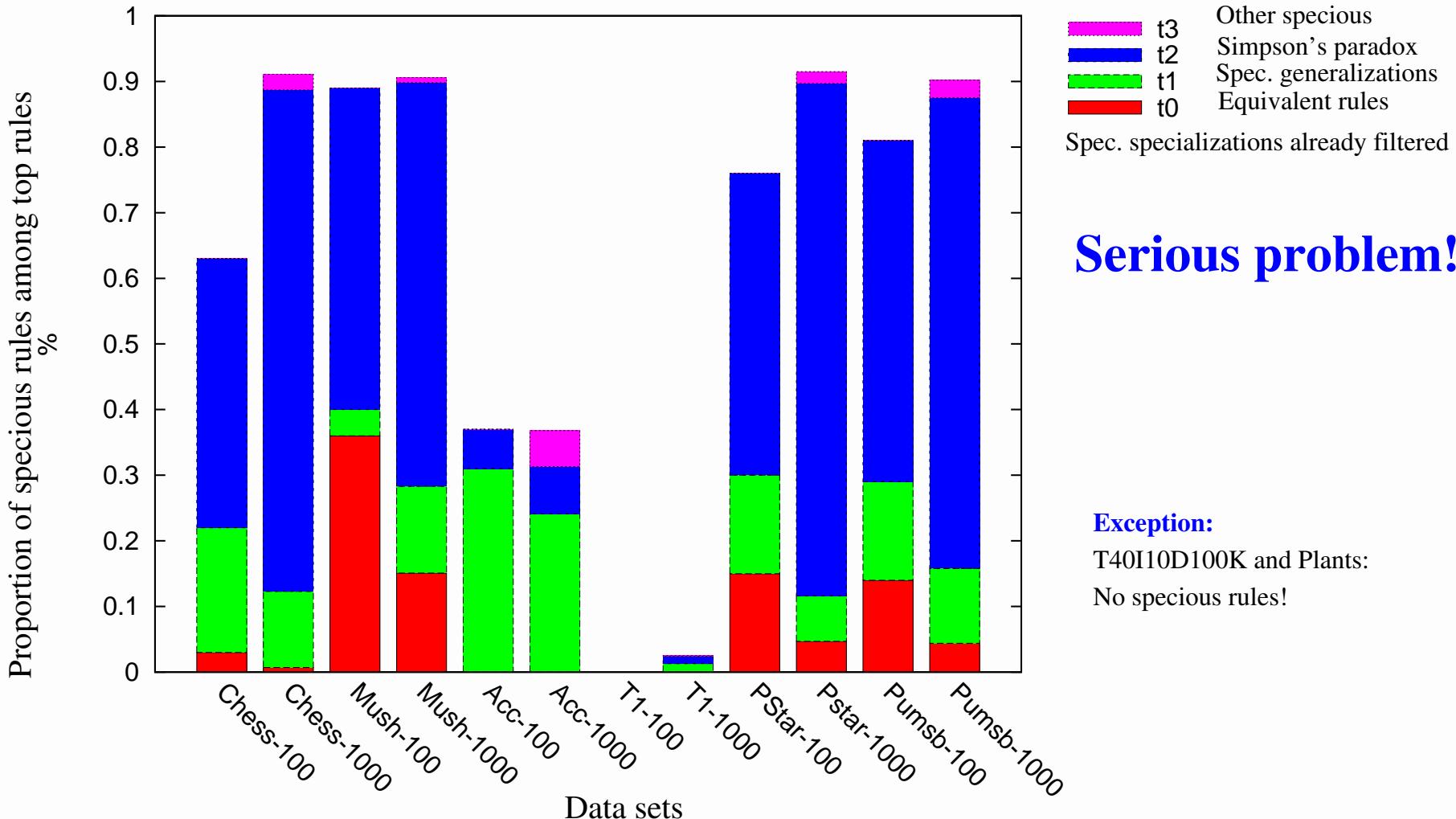
E.g., Is $Q \rightarrow C$ specious by $X \rightarrow C$ or $X \rightarrow \neg C$?



Problem: How to find confounding X among exponentially many possibilities?? For an efficient solution, see Hämäläinen and Webb 2017!

Experiments: Specious rules and Simpson's paradox are very common!

Proportion and types of specious rules among top-100 or top-1000 best rules



Summary

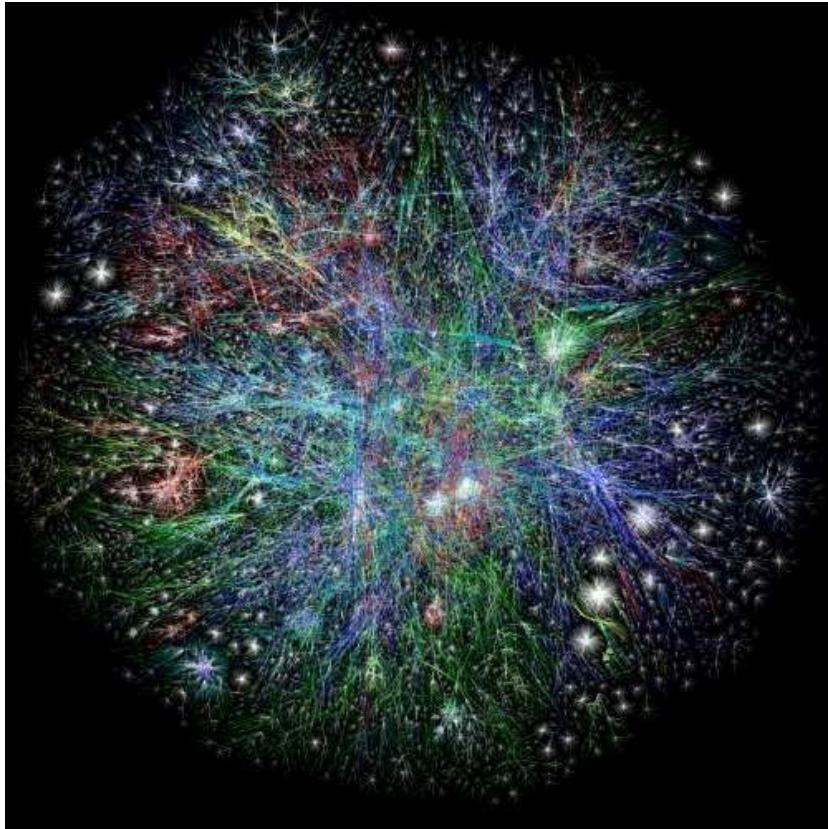
- **Computational problems:**
 - complete enumeration tree has 2^k nodes – not feasible!
 - frequency counting done for each generated node – costly!

⇒ strategies
- **Generic Apriori** given monotonic property Φ
- **Specious association** $Q \rightarrow C=c$ disappears or reverses its sign when conditioned on confounding factor X
 - sideproduct of $X \rightarrow C=c$ and $X \rightarrow Q$
 - or $X \rightarrow C \neq c$ and $X \rightarrow \neg Q$

References

- Hämäläinen and Webb: Specious rules: an efficient and effective unifying method for removing misleading and uninformative patterns in association rule mining. SIAM Int. Conf. Data Mining, 2017.
<https://arxiv.org/pdf/1709.03915.pdf>
- Lindley and Novick: The Role of Exchangeability in Inference, Annals of Statistics 9(1):45-58, 1981.

Mining graph data: Web and recommender systems



1. Introduction to Graph mining
2. Web mining and search
 - Emphasis: **Ranking algorithms**
3. Recommender systems
 - Emphasis: **Collaborative filtering**

image source: Opte project, <https://www.opte.org/the-internet>

1. Introduction to Graph mining

Data may consist of

1. one large graph
 - e.g., internet, social network
2. multiple small graphs
 - e.g., chemical compounds, biological pathways, program control flows, consumer behaviour, ...

Information to mine: interesting substructures, influential nodes, similarities, communities, clusters

Data: one large graph

Internet (Wikipedia hyperlink structure)

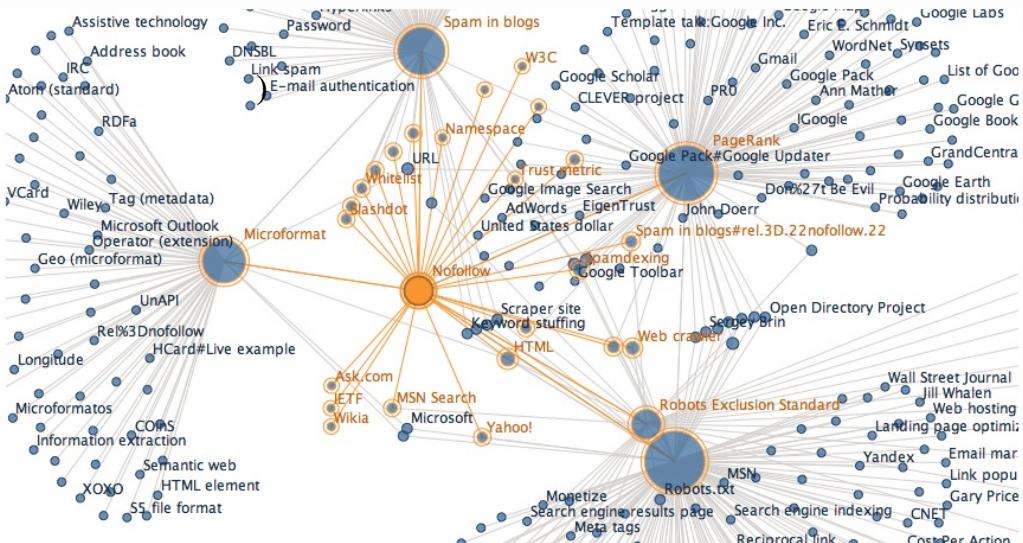


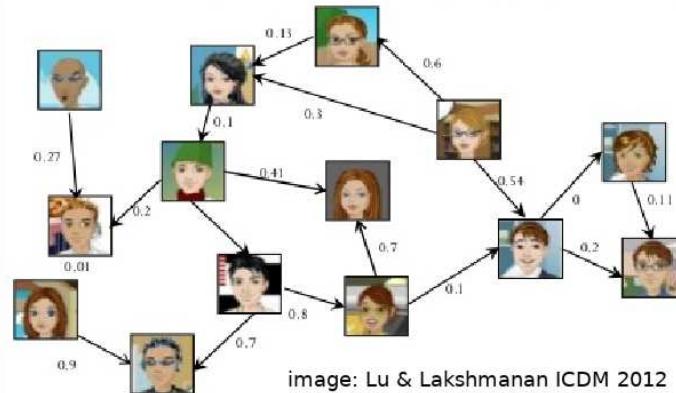
Image source:<https://wiki.digitalmethods.net/Dmi/WikipediaAnalysis>

Most influential nodes?

Communities or dense subgraphs?

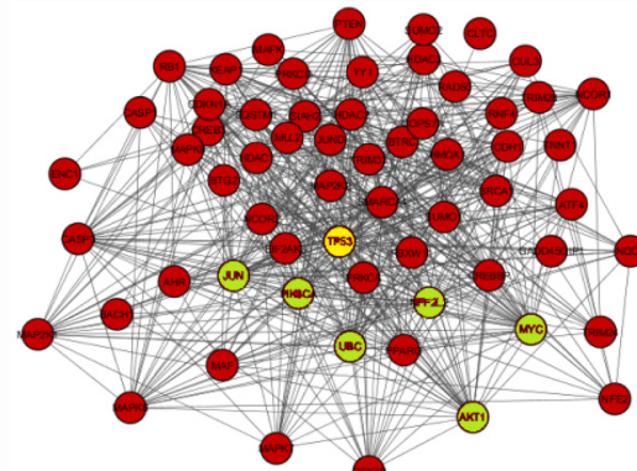
Node similarity/future links?

Social network



<https://www.slideshare.net/WeiLu12/profit-maximization-over-social-networks>

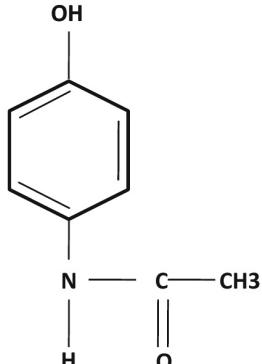
Protein–protein interaction network



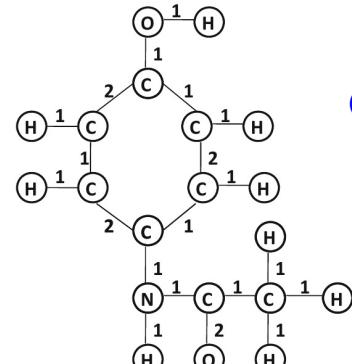
Srivastava et al. (2018)
doi 10.2174/1875036201811010240

Data: multiple (smaller) graphs

Molecular structure graphs



(a) Acetaminophen



(b) Graph representation

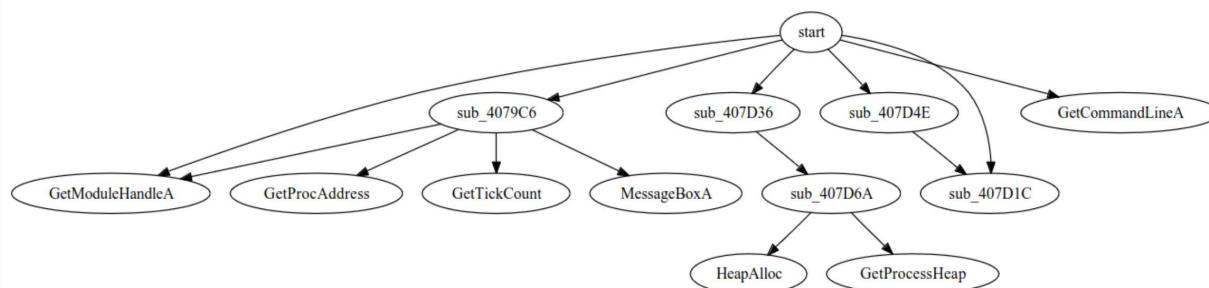
Aggarwal Fig. 17.1

Common and unique properties?

Frequent subgraphs?

Clusters?

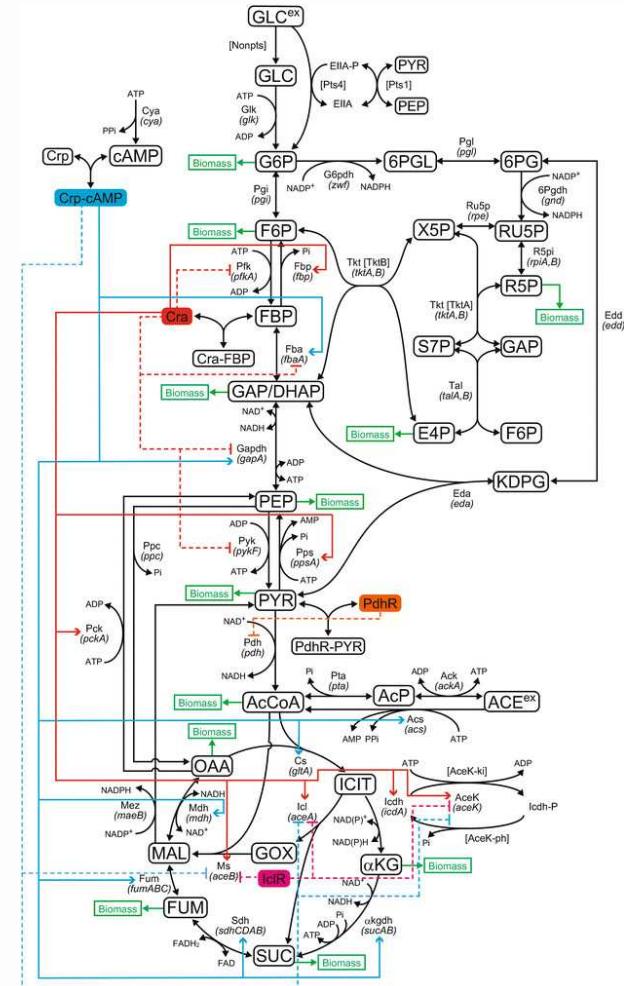
Software call graphs (malware)



Kinable & Kostakis (2011)

doi <https://doi.org/10.1007/s11416-011-0151-y>

Metabolic network (E. coli)

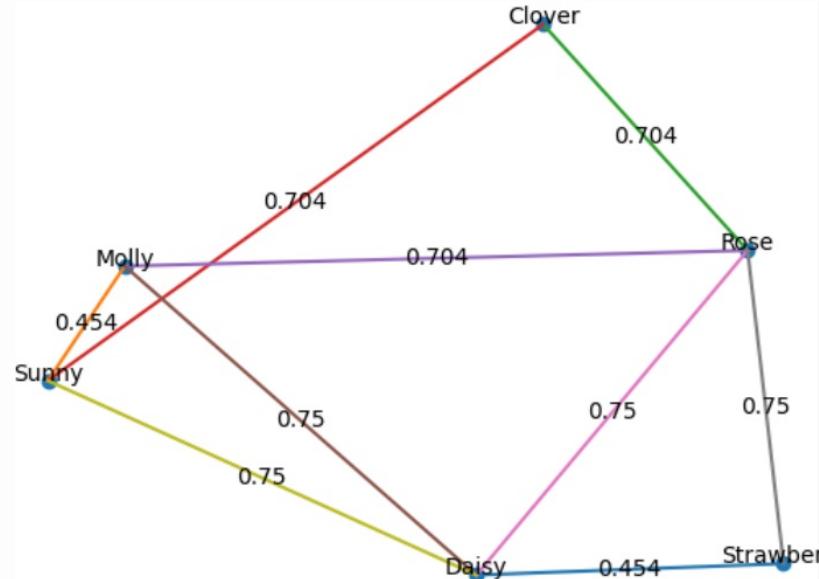


Jahan et al. (2016)

DOI:10.1186/s12934-016-0511-x

Data: graph-form presentation of other data

Similarity graph



User-item utility graph

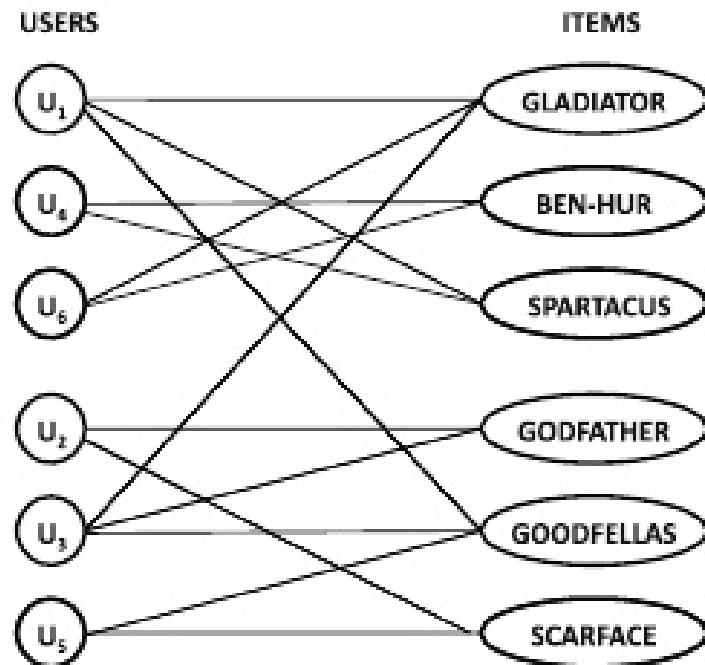


Image sources MDM2022/Laaksonen and Aggarwal Fig 18.5

2a Web mining

Data:

1. Web content = documents and links
2. Web usage data, like buying and browsing behaviour, ratings, logs

Applications:

1. **Content-centric**: document clustering, resource discovery, web search, linkage mining
2. **Usage-centric**: recommender systems, web log analysis

2b Web search (information retrieval)

1. **Web crawling**: collect all (relevant) documents into a central location (+ keep it up to date) → Aggarwal 18.2
2. **Index construction** for the collection (offline)
3. **Query processing** (online)
 - query = set of keywords $\mathbf{q} = (w_1, \dots, w_k)$
 - identify documents containing all/most $w_i \in \mathbf{q}$
 - **rank** documents (relevance to the query and quality)
 - return best ranked documents

Search engine: steps 2 + 3

Index construction

Preprocess documents, extract relevant tokens (words or phrases) and construct **inverted indices**

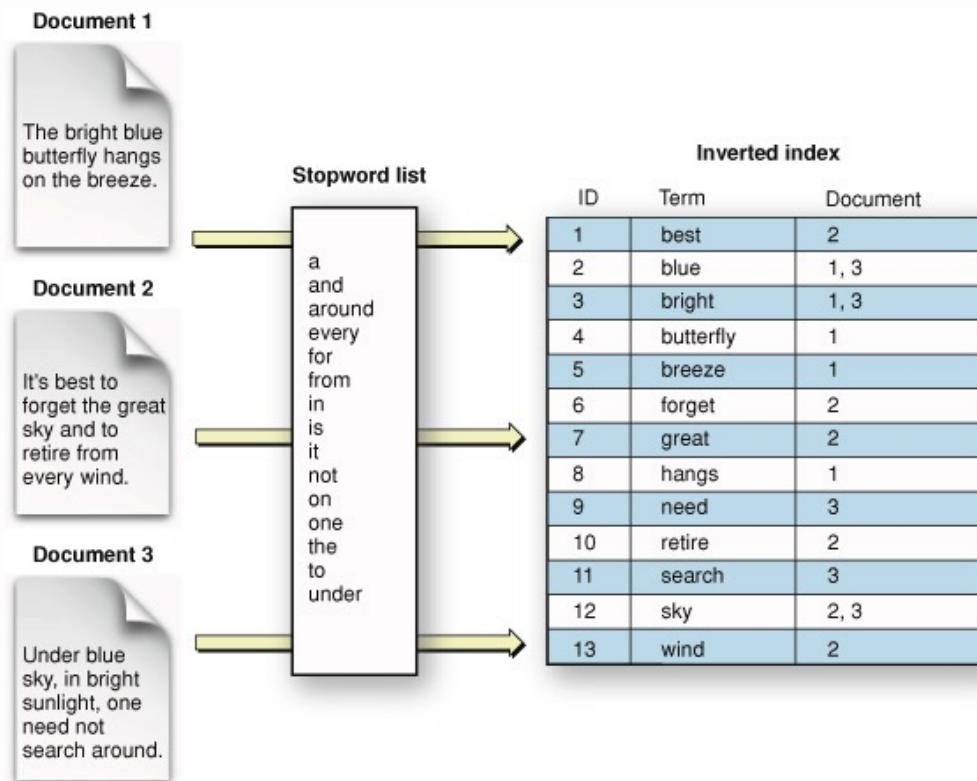


Image source <https://community.hitachivantara.com/s/article/search-the-inverted-index>

Ranking documents

Combine two types of scores:

1. Content-based scores based on occurrence of keywords

- frequency of word
- where the word occurs (more weight if in the title or anchor text)
- prominence of the word (font size, colour)
- relative position of keywords (more relevant if close to each other)

How web spammer can cheat the search engine to get high content-based scores?

Ranking documents

2. Reputation-based scores utilize natural voting mechanisms

a) assumptions based on page citations

- High quality pages are pointed by many pages or
- High quality pages are pointed by many high quality pages

b) user feedback

- users choose relevant pages
- return other similar pages or pages accessed by similar users (recommender systems)

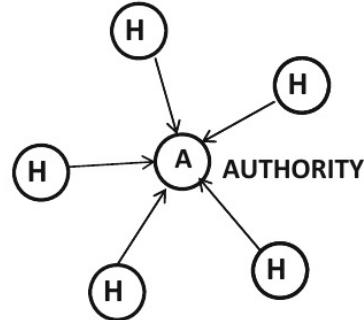
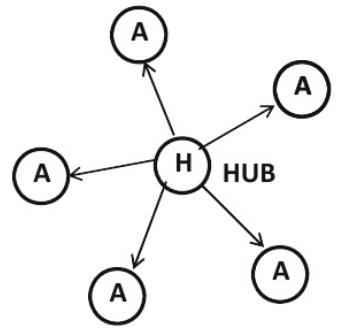
Two ranking algorithms

1. HITS = Hyperlink-Induced Topic Search =
"Hubs and Authorities algorithm"
 - query-dependent
 - There are two types of good pages: good authority pages are reputable sources on topics and good hub pages offer links to good sources.
2. PageRank
 - query-independent
 - Highly reputabe pages are likely cited by other higly reputable pages.

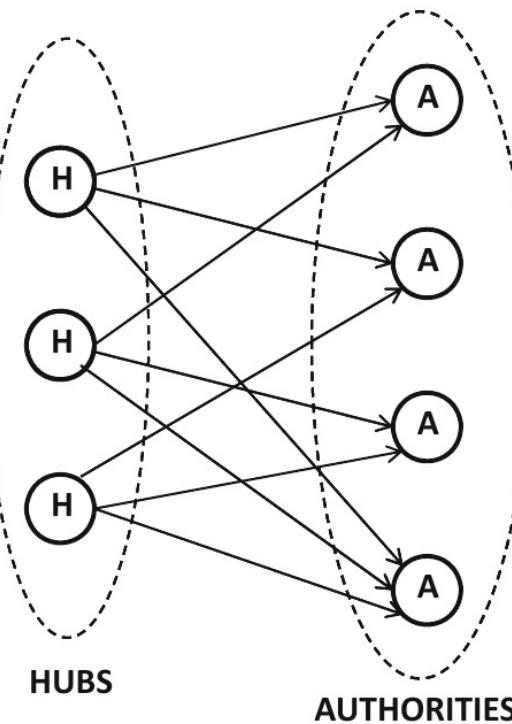
Hyperlink-Induced Topic Search (HITS)

hub = page that refers to many authoritative pages

authority = authoritative page referred by many hubs



(a) Hub and authority
examples



(b) Network organization between
hubs and authorities

image source Aggarwal Fig. 18.3

HITS basic idea

1. Construct an input graph:

- search top- K pages most relevant to the query (e.g., top-200) → root set \mathbf{R}
- add pages pointed by $v \in \mathbf{R}$ and some of pages pointing to $v \in \mathbf{R}$ (e.g., max 50/per node) → base set \mathbf{V}
- construct a graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ where \mathbf{E} contains all links between nodes in \mathbf{V}

2. Assign all nodes hub and authority weights, $h(v)$ and $a(v)$

- update weights until the system converges
- return nodes v with highest $a(v)$

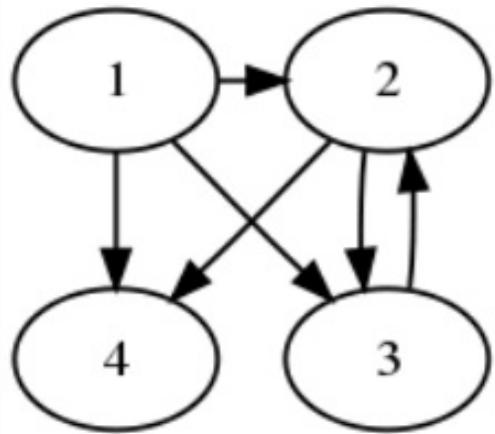
HITS algorithm on $G = (V, E)$

$In(v) = \{u \mid (u, v) \in E\}$ nodes pointing to v

$Out(v) = \{u \mid (v, u) \in E\}$ nodes to which v points

1. initialize h and a such that $\sum_i h(v_i)^2 = \sum_i a(v_i)^2 = 1$
(e.g., $h(v_i) = a(v_i) = \frac{1}{\sqrt{n}}$)
2. until h and a values converge, update for all v_i :
 - $h(v_i) = \sum_{w \in Out(v_i)} a(w)$ (authorities pointed by v_i)
 - $a(v_i) = \sum_{w \in In(v_i)} h(w)$ (hubs pointing to v_i)
 - normalize weights: $h(v_i) = \frac{h(v_i)}{H}$ and $a(v_i) = \frac{a(v_i)}{A}$,
where $H = \sqrt{\sum_i h(v_i)^2}$ ja $A = \sqrt{\sum_i a(v_i)^2}$
3. return v_i s with largest $a(v_i)$

Example: calculation of h and a



initialize $h(i) = a(i) = \frac{1}{\sqrt{4}} = \frac{1}{2}$

round 1:

$$h(1) = \frac{3}{2}, a(1) = 0$$

$$h(2) = \frac{2}{2}, a(2) = \frac{2}{2}$$

$$h(3) = \frac{1}{2}, a(3) = \frac{2}{2}$$

$$h(4) = 0, a(4) = \frac{2}{2}$$

normalize by $H = \sqrt{\frac{7}{2}}$ and $A = \sqrt{3}$

$$h(1) = \frac{3}{\sqrt{14}}, a(1) = 0$$

$$h(2) = \frac{2}{\sqrt{14}}, a(2) = \frac{1}{\sqrt{3}}$$

$$h(3) = \frac{1}{\sqrt{14}}, a(3) = \frac{1}{\sqrt{3}}$$

$$h(4) = 0, a(4) = \frac{1}{\sqrt{3}}$$

Example: final result

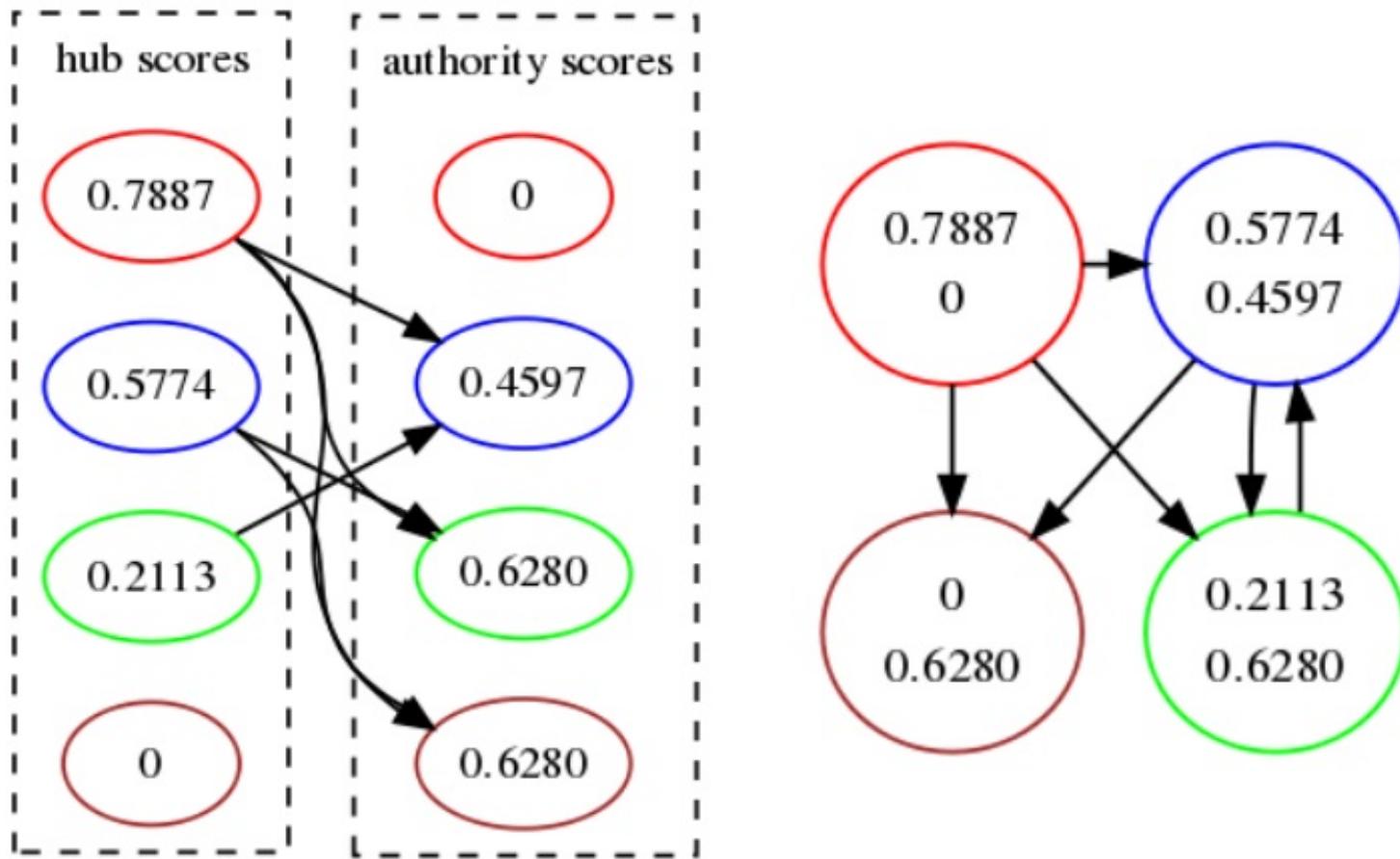


image source Pajarinan (2008): The PageRank/HITS algorithms (slides)

PageRank

Uses a **random surfer model** = **random walk model**

- visit random pages by selecting random links
- long term visiting frequency of a page depends on the number of in-linking pages and their visiting frequency
- ⇒ frequency is high, if linked from other frequently visited pages
- reputation \approx long term frequency of visits by a random surfer = **steady-state probability** π

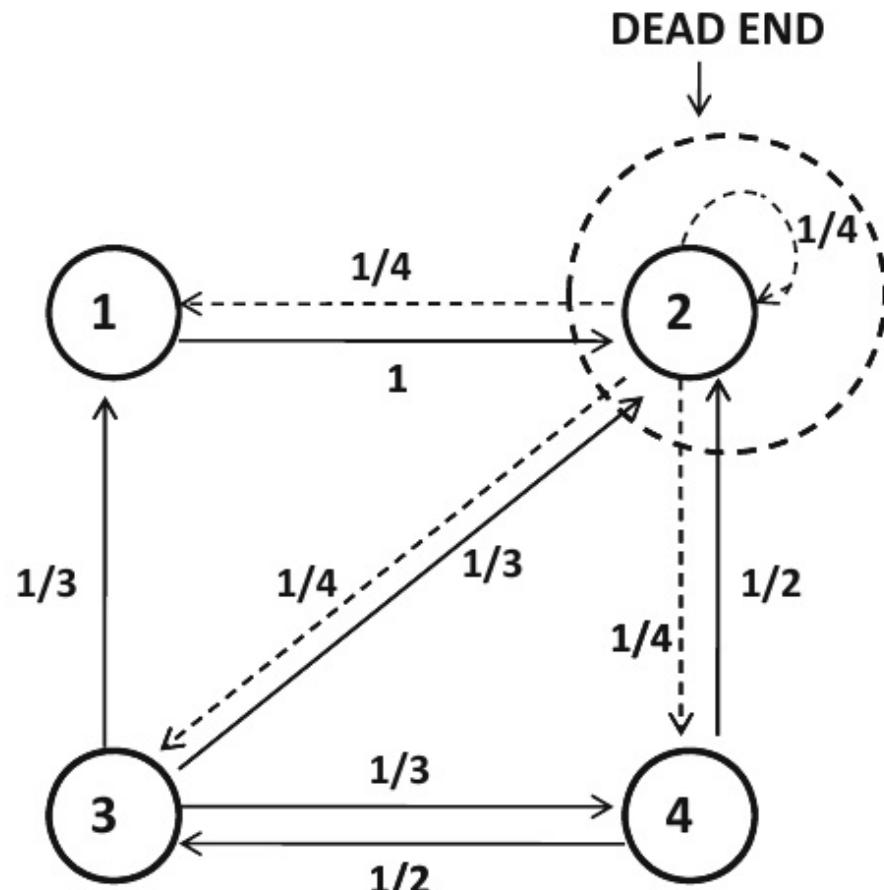
⇒ calculate $\pi = (\pi_1, \dots, \pi_n)^T$, where π_i = PageRank of the i th node v_i (n =number of pages)

Problem: the surfer can get trapped!

Originally, all out-going links from a node have equal probability.

Dead-end nodes don't have out-going links

⇒ add links from a dead-end node to all n pages with transition probability $\frac{1}{n}$



DASHED TRANSITIONS ADDED
TO REMOVE DEAD END

Problem: the surfer can get trapped!

Dead-end components don't contain out-going links from the group

⇒ all steady-state probability becomes concentrated into such groups!

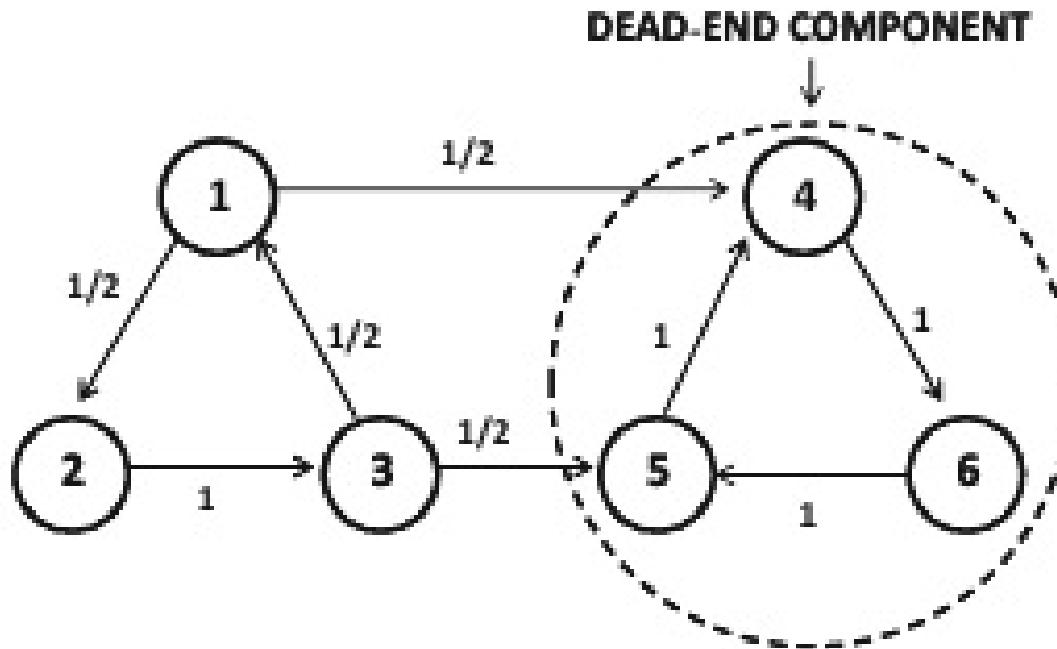


image source Aggarwal Fig. 18.2

Teleportation offers a solution to dead-end components

At each transition, a random surfer may

- jump to an arbitrary page with probability α or
- follow one of links with probability $1 - \alpha$

α = smoothing or damping probability

- typically $\alpha = 0.1$
- large α makes steady-state probability distribution more uniform
- What happens if $\alpha = 1$?

Presentations as a Markov chain

- graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, \mathbf{V} = pages, $|\mathbf{V}| = n$, \mathbf{E} = links
- $In(v) = \{u \mid (u, v) \in \mathbf{E}\}$ nodes pointing to v
- $Out(v) = \{u \mid (v, u) \in \mathbf{E}\}$ nodes pointed by v
- transition matrix \mathbf{P} , where $p_{ij} = \mathbf{P}[i, j]$ probability of transitioning $v_i \rightarrow v_j$
- set $p_{ij} = \frac{1}{|Out(v_i)|}$
- $\pi = (\pi_1, \dots, \pi_n)^T$ steady-state probabilities

$$\pi_i = \frac{\alpha}{n} + (1 - \alpha) \sum_{v_j \in In(v_i)} p_{ji} \cdot \pi_j$$

Computation with the power iteration method

Idea: update π iteratively, notate by $\pi^{(t)}$ the current π at time step t .

Let $\alpha = (\alpha, \alpha, \dots, \alpha)$ (vector of n α s)

- initialize $\pi_i^{(0)} = 1/n$, $t = 0$
- until $\pi^{(t)}$ converges do
 - i) calculate $\pi^{(t+1)}$:

$$\pi^{(t+1)} = \frac{\alpha}{n} + (1 - \alpha)\mathbf{P}^T \pi^{(t)}$$

- ii) normalize such that $\sum \pi_i = 1$

Computation with the power iteration method

Given convergence threshold θ :

- initialize $\pi_i^{(0)} = 1/n$, $t = 0$
- until $(\|\boldsymbol{\pi}^{(t)} - \boldsymbol{\pi}^{(t+1)}\| \leq \theta)$ do
 - for all $i = 1, \dots, n$

$$\pi_i^{(t+1)} = \frac{\alpha}{n} + (1 - \alpha) \sum_{v_j \in In(v_i)} p_{ji} \pi_j^{(t)}$$

- for all $i = 1, \dots, n$ normalize $\pi_i^{(t+1)}$ s

Note: heavy computation! \rightarrow calculate beforehand for all pages

Alternatively integrate teleportation probabilities into the transition matrix (here $\alpha = 0.1$)

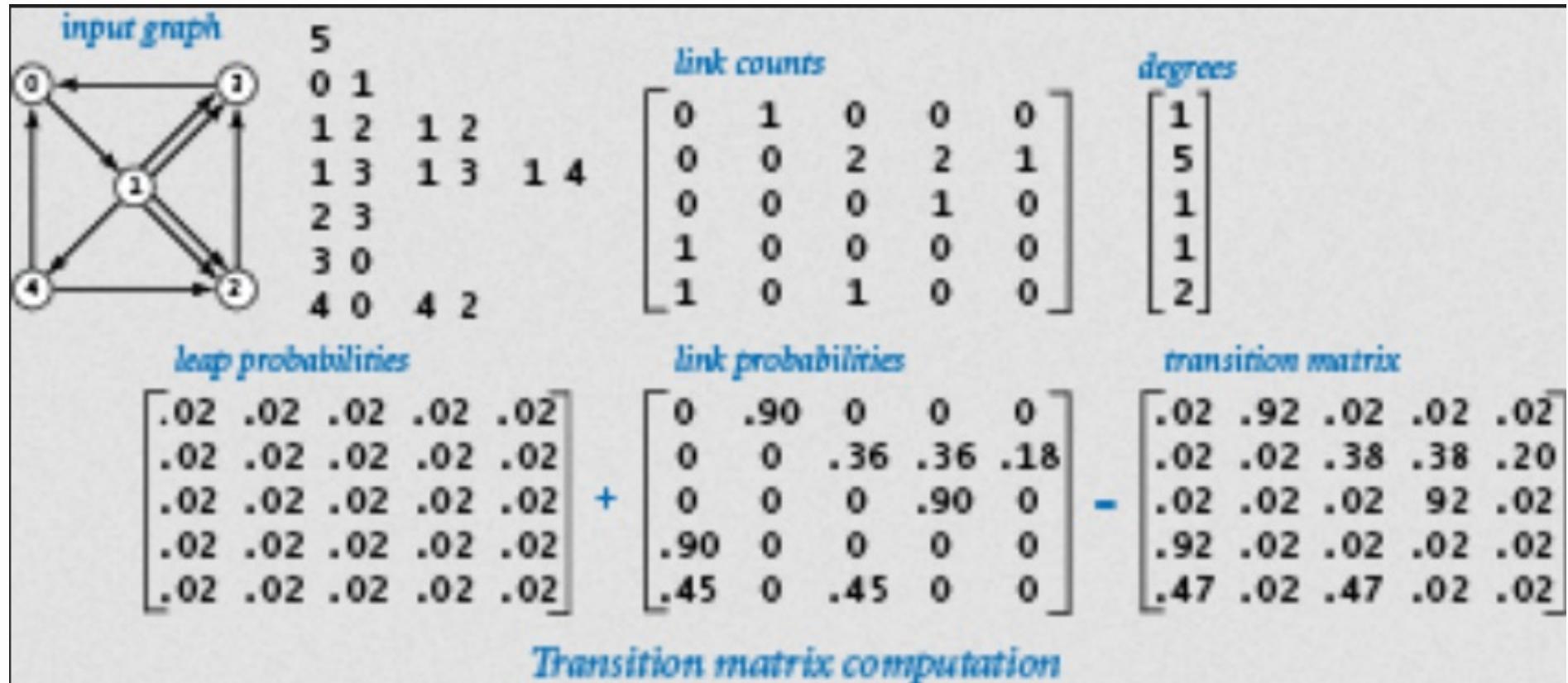


image source

<https://introcs.cs.princeton.edu/python/16pagerank/>

Other PageRank style methods

1. Topic-sensitive PageRank favours certain topics (like user's interests)

- fix a list of base topics
- search high quality sample of pages from each topic
- restrict teleportation only to these pages

2. SimRank measures similarity between nodes

Intuition: Two random surfers walking backwards from nodes i and j take $L(i, j)$ steps, until they meet. Then $\text{SimRank}(i, j)$ is expected value of $c^{L(i,j)}$, where c = decay constant.

SimRank

$SimRank(v_i, v_j) =$

$$\begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } |In(v_i)| = 0 \text{ or } |In(v_j)| = 0 \\ \frac{c}{|In(v_i)| \cdot |In(v_j)|} \sum_{p \in In(v_i)} \sum_{q \in In(v_j)} SimRank(p, q) & \text{otherwise} \end{cases}$$

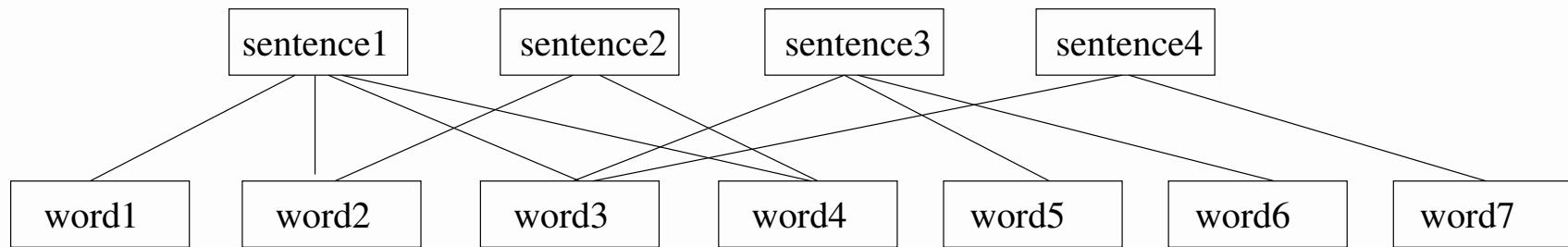
Further reading Aggarwal Ch 18.4.1.2

Summary

- Important problem how to rank documents!
- Content-based scores based on page contents
- Reputation-based scores utilize e.g., link structure
- Ranking algorithms:
 - HITS: query-dependent → smaller input graph
 - PageRank: query-independent → calculate for all pages (heavy!)

Extra: Applying HITS to sentences and words

Given a keyword or words, create a bipartite graph:



- e.g., all sentences where the given word occurs, all their words, all their sentences and randomly some of their words
- sentences (s) are given S weights and words (w) W weights
- initialize uniformly such that the square sum is 1

Extra: Applying HITS to sentences and words

- update rule: $S(s) = \sum_{w_i \in s} W(w_i)$ and $W(w) = \sum_{w \in s_i} S(s_i)$ + normalization
- in the end the words with largest S -weight are most similar
- analogously search the most similar sentences with a given one
- quality of results varies a lot depending on the documents, language, preprocessing, and how the graph was constructed!

3. Recommender systems

Problem: What items to recommend to which user? Products, music, movies, dishes, learning material,...



Image source <https://sudonull.com/post/12374-Anatomy-of-recommendation-systems-Part-one>

Data and utility matrix

Data: User profiles, product descriptions, browsing and buying behaviour, explicit ratings.

Often possible to derive a **utility matrix A**, where
 $A[i, j]$ = utility of item j for user i

- $n \times d$ matrix, n =number users, d =number of items

Two types:

1. Only positive preferences (“likes”, browsing, buying)
 2. Positive and negative preferences (“likes” and “dislikes”, ratings)
- extremely large and sparse matrices!

Example utility matrices (movie preferences)

	GLADIATOR	GODFATHER	BEN-HUR	GOODFELLAS	SCARFACE	SPARTACUS
U_1	1			5		2
U_2		5			4	
U_3	5	3		1		
U_4			3			4
U_5				3	5	
U_6	5		4			

(a) Ratings-based utility

	GLADIATOR	GODFATHER	BEN-HUR	GOODFELLAS	SCARFACE	SPARTACUS
U_1	1			1		1
U_2		1			1	
U_3	1	1		1		
U_4			1			1
U_5				1	1	
U_6	1		1			

(b) Positive-preference utility

Empty cell=unspecified; in data, e.g., -, na, 0 (if non-positive ratings).

Image source Aggarwal Ch 18

Main approaches: Content-based and preference-based (collaborative filtering)

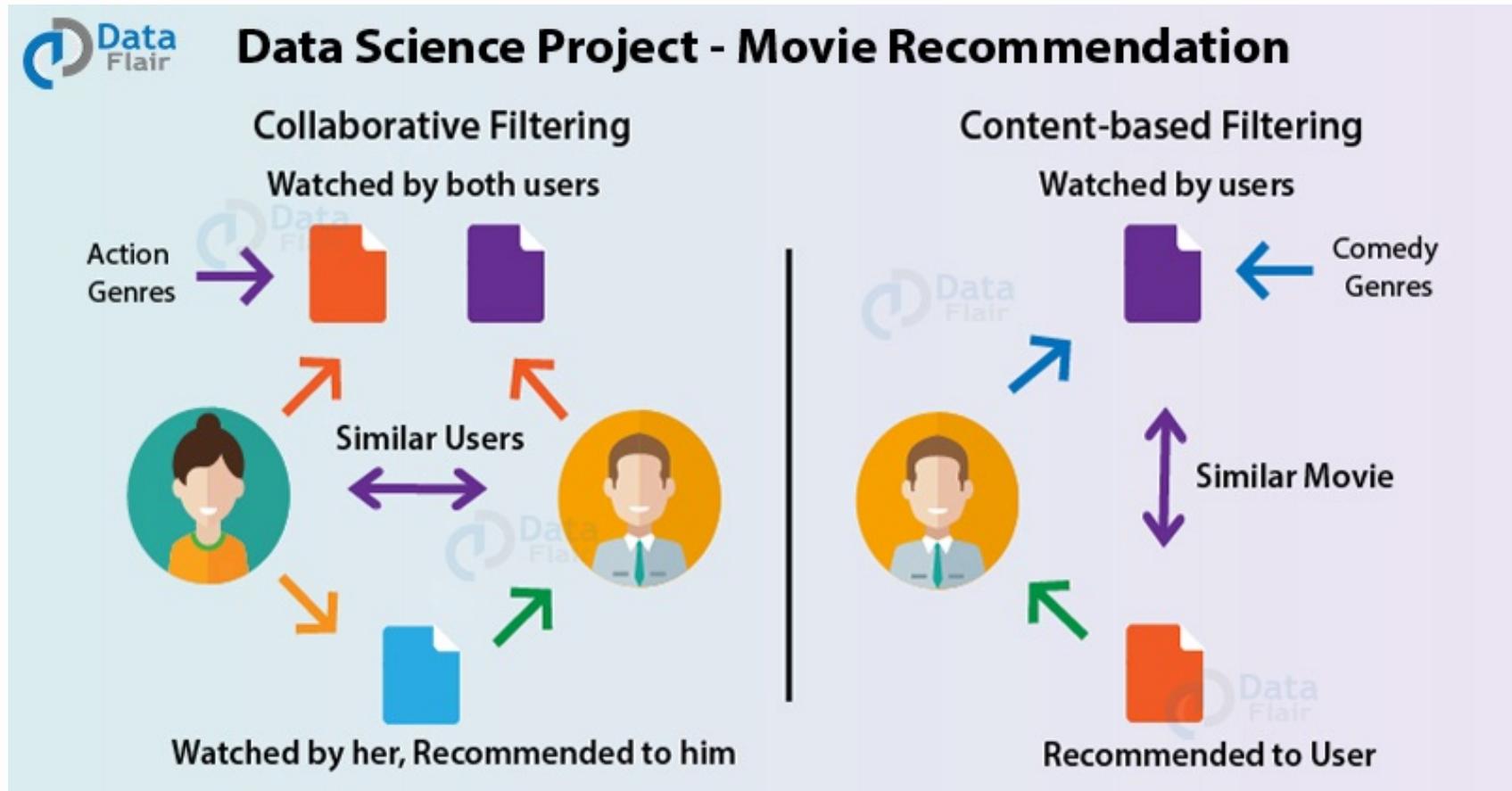


Image source

<https://data-flair.training/blogs/data-science-r-movie-recommendation/>

Content-based recommendations

Given

1. **item profile** = text descriptions, keywords
2. **user profile** = documents describing user's interests
(e.g., descriptions of previously bought/liked items,
explicitely specified or derived interests)

Search items whose profiles match (are similar) to the user's profile

a) **If no utility matrix**

- search K most similar items to the user profile
- e.g., tf-idf presentation + cos similarity

Content-based recommendations

- b) If **utility matrix** exists, utilize the user's previous preferences!
- = prediction task where vector $\mathbf{A}[i]$ = target values for user i
- If positive preference matrix, learn a classifier
 - If numerical ratings, learn a regression model
 - training sets extremely small
 - over-specialization: recommendations tend to favour items described by the same keywords
 - e.g., recommend movies with the same actors as before

Collaborative filtering

Assumption: The user probably likes what other similar users have liked.

Approaches for recommendation

- i) Neighbourhood-based
- ii) Graph-based
- iii) Clustering-based
- iv) Latent factor -based

Neighbourhood-based methods: 1. user-based

Utilize user–user similarity

e.g., **Pearson correlation coefficient r** for similarity between two users' rating vectors $\mathbf{x} = (x_1, \dots, x_d)$ and $\mathbf{y} = (y_1, \dots, y_d)$:

$$r(\mathbf{x}, \mathbf{y}) = \frac{\sum_{j \in J} (x_j - \mu_x)(y_j - \mu_y)}{\sqrt{\sum_{j \in J} (x_j - \mu_x)^2 \sum_{j \in J} (y_j - \mu_y)^2}}$$

$J = \{j \mid x_j \neq na, y_j \neq na\}$ (items rated by both)
 μ_x is average rating, two alternatives:

- i) $\mu_x = \frac{1}{|J|} \sum_{j \in J} x_j$ (only common items) or
- ii) $\mu_x = \frac{1}{|J_x|} \sum_{j \in J_x} x_j$, where $J_x = \{j \mid x_j \neq na\}$ (all rated items; more common approach)

Predict missing ratings in rating vector \mathbf{x}

1. search K nearest neighbours $NN_{\mathbf{x}}$ using similarity r
2. remove neighbours from $NN_{\mathbf{x}}$ if $r \leq \theta$ (negative or weak correlations)
3. normalize ratings: $y'_j = y_j - \mu_y$ (since in different scales)
4. calculate predicted rating for all items j with missing entries in \mathbf{x} :

$$\tilde{x}_j = \frac{\sum_{\mathbf{y} \in NN_{\mathbf{x}}} w_{\mathbf{y}} \cdot y'_j}{\sum_{\mathbf{y} \in NN_{\mathbf{x}}} w_{\mathbf{y}}} + \mu_x$$

- $w_{\mathbf{y}} = 1$ or weigh by similarity $w_{\mathbf{y}} = r(\mathbf{x}, \mathbf{y})$
- i.e., weighted average rating by similar users + return to \mathbf{x} 's original scale

Example: Predict missing ratings ($K = 2$, $r \geq 0.5$)

	m_1	m_2	m_3	m_4	m_5	m_6
u_1	—	1	2	2	3	—
u_2	3	1	1	2	4	3
u_3	4	2	3	3	—	5
u_4	2	5	4	—	1	2

User means:

$$\mu_1 = 2.000$$

$$\mu_2 = 2.333$$

$$\mu_3 = 3.400$$

$$\mu_4 = 2.800$$

	u_1	u_2	u_3	u_4
u_1	1.000	0.836	0.927	-0.917
u_2	0.836	1.000	0.822	-0.974
u_3	0.927	0.822	1.000	-0.862
u_4	-0.917	-0.974	-0.862	1.000

m_1 =Gladiator, m_2 =Godfather, m_3 =Ben-Hur, m_4 =Goodfellas, m_5 =Scarface, m_6 =Spartacus

Example: Predict missing ratings ($K = 2$, $r \geq 0.5$)

	u_1	u_2	u_3	u_4
u_1	1.000	0.836	0.927	-0.917
u_2	0.836	1.000	0.822	-0.974
u_3	0.927	0.822	1.000	-0.862
u_4	-0.917	-0.974	-0.862	1.000

$$\begin{aligned}\mu_1 &= 2.000 \\ \mu_2 &= 2.333 \\ \mu_3 &= 3.400 \\ \mu_4 &= 2.800\end{aligned}$$

for u_1 nearest u_3 and u_2 , predicted for u_1 , m_1 :

$$\frac{0.836 \cdot (3 - 2.333) + 0.927 \cdot (4 - 3.400)}{0.836 + 0.927} + 2.000 = 2.63 > \mu_1 \rightarrow \text{recommend}$$

for u_1 , m_6 predicted 3.16

for u_3 nearest u_1 and u_2 , for m_5 predicted 4.71

for u_4 not enough neighbours! (all $r < 0$)

Neighbourhood-based methods: 2. item-based

Utilize **item–item similarity**

$\mathbf{v} = j$ th item's rating vector, $\mathbf{x} = i$ th user's rating vector

1. search K nearest neighbours $NN_{\mathbf{v}}$ of \mathbf{v}
2. select a subset $NN_{\mathbf{v}, \mathbf{x}} \subseteq NN_{\mathbf{v}}$ of those items's ratings that user i has rated: $NN_{\mathbf{v}, \mathbf{x}} = \{\mathbf{u}_r \mid \mathbf{u}_r \in NN_{\mathbf{v}}, x_r \neq na\}$
3. Predicted rating is

$$\tilde{x}_j = \frac{\sum_{\mathbf{u}_r \in NN_{\mathbf{v}, \mathbf{x}}} w_{\mathbf{v}, \mathbf{u}_r} \cdot x_r}{\sum_{\mathbf{u}_r \in NN_{\mathbf{v}, \mathbf{x}}} w_{\mathbf{v}, \mathbf{u}_r}}$$

- i.e., weighted average rating on similar items by user i

Neighbourhood-based methods: 2. item-based

What similarity measure to use? Should we normalize ratings?

- Pearson correlation (+ mean centering can be used also here)
- **adjusted cosine similarity** = cosine similarity after mean centering each user's ratings
- **Problem:** cosine similarity with 0 vectors (movies with average ratings from everybody) is not defined
↔ cos works better when all values positive

See Aggarwal 18.5.2.2

Graph-based methods

Idea: Create a bipartite **user-item graph** and utilize random walk approaches.

- graph $\mathbf{G} = (\mathbf{U} \cup \mathbf{V}, \mathbf{E})$
- \mathbf{U} = nodes for users
- \mathbf{V} = nodes for items
- \mathbf{E} = edges such that $(u_i, v_j) \in \mathbf{E}$, $u_i \in \mathbf{U}$, $v_j \in \mathbf{V}$, if the i th user has rated the j th item
- if rating matrix (positive and negative preferences), the edges may have weights:
 - normalize rating $A[i, j]$ by subtracting mean of ratings on row $A_i \rightarrow$ signed network

Bipartite user-item graph

	GLADIATOR	GODFATHER	BEN-HUR	GOODFELLAS	SCARFACE	SPARTACUS
U_1	1				1	
U_2		1				1
U_3	1	1		1		
U_4			1			1
U_5				1	1	
U_6	1		1			

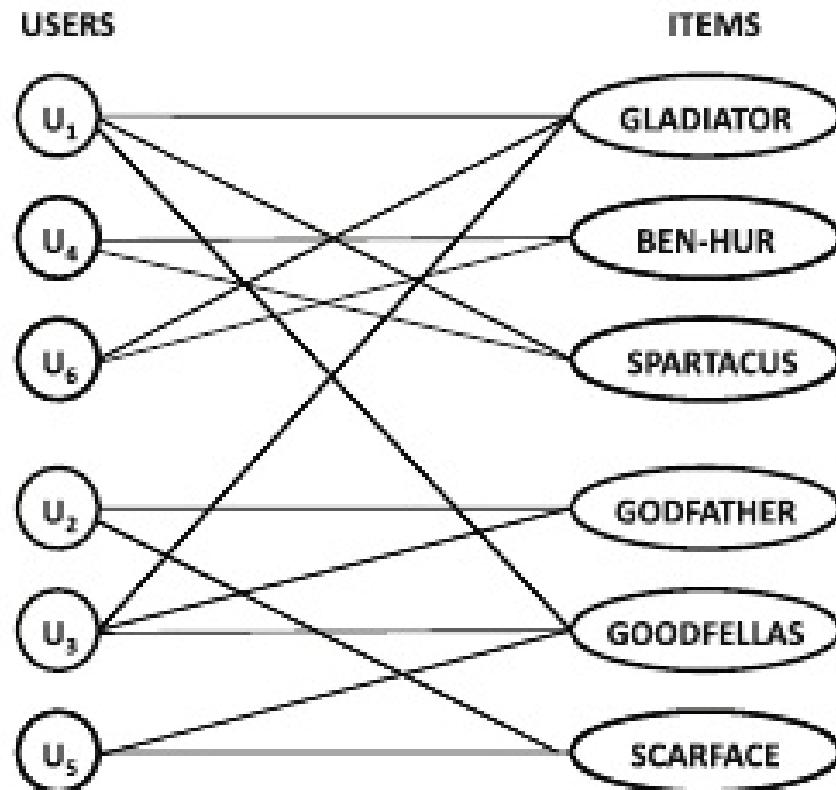


Image source Aggarwal Ch 18

Making recommendations

Let G be unweighted (presents only positive preferences).

Two approaches:

1. Use G only to determine nearest neighbours:

- determine K most similar users to the i th user using personalized PageRank or SimRank
- or K most similar items to the j th item
- make recommendations as before (user-based or item-based)

Making recommendations

2. Use PageRank values to decide recommendations:

- i) given user i , search **item nodes** with largest PageRank values, when teleportation to user node u_i
→ recommend these items to the i th user
- ii) given item j , search **user nodes** with largest PageRank values, when teleportation to item node v_j
→ recommend the j th item to these users
- teleportation probability α affects results
 - small α favours popular items
 - larger α makes recommendations more specific to the given user

Clustering-based methods

Idea: Determine peer groups (similar users or similar items) beforehand by clustering.

↔ neighbourhood-based methods determine them separately for all users

What clustering methods to use?

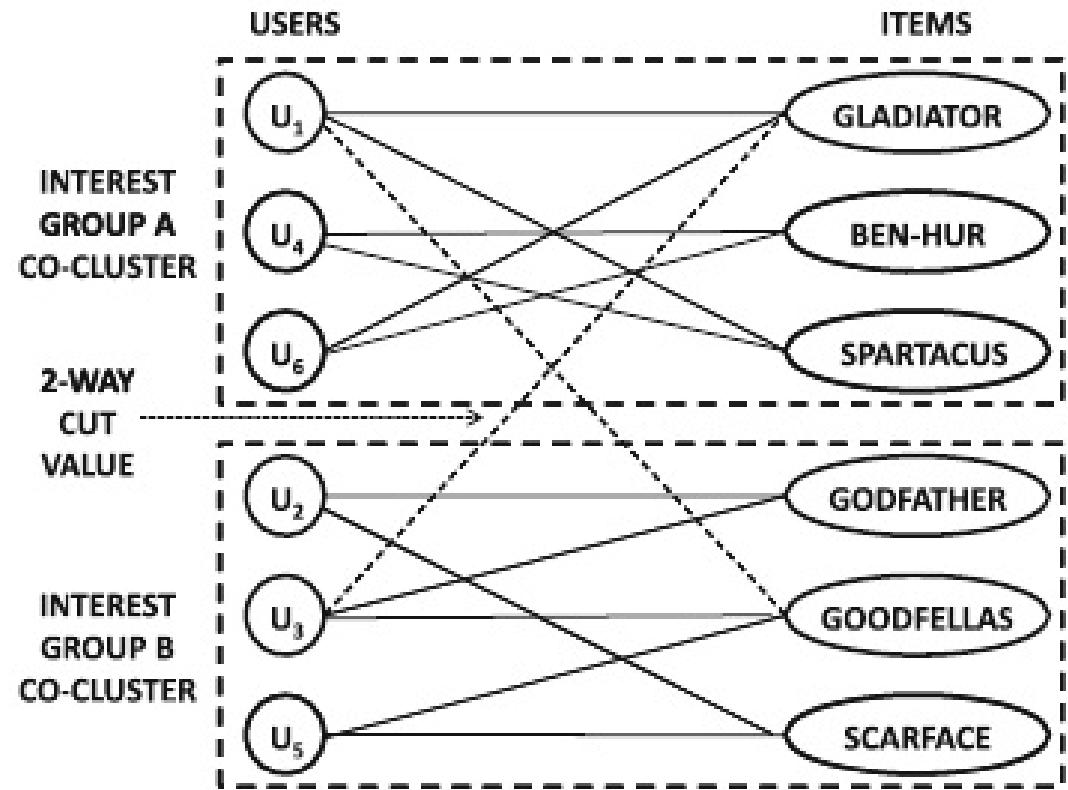
- problem: data sets very sparse (many missing values)
- adapt K -means:
 - calculate distances $d(\mathbf{x}, \mathbf{c}_i)$ and centroids \mathbf{c}_i only over those dimensions where ratings available
- co-clustering approaches

Co-clustering of movie preference data

	GLADIATOR	BEN-HUR	SPARTACUS	GODFATHER	GOODFELLAS	SCARFACE
INTEREST GROUP A CO-CLUSTER	1		1		1	
U ₁	1		1		1	
U ₄		1	1			
U ₆	1	1				
U ₂				1		1
U ₃	1			1	1	
U ₅					1	1

INTEREST GROUP B CO-CLUSTER

(a) Co-cluster



(b) User-item graph

Image source Aggarwal Ch 18

Latent factor -based methods

Idea: summarize correlations by latent factors → smaller dimensional representation of utility matrix $\mathbf{A} \approx \mathbf{F}_U \mathbf{F}_I^T$

- present n users by n k -dimensional latent factors,
 $\mathbf{F}_{U1}, \dots, \mathbf{F}_{Un}$
- present d items by d k -dimensional latent factors,
 $\mathbf{F}_{I1}, \dots, \mathbf{F}_{In}$
- k = new reduced dimensionality of latent representation
- estimate rating $\mathbf{A}[i, j] \approx \mathbf{F}_{Ui} \cdot \mathbf{F}_{Ij}$
- use (modified) SVD or other matrix factorization to get latent factors

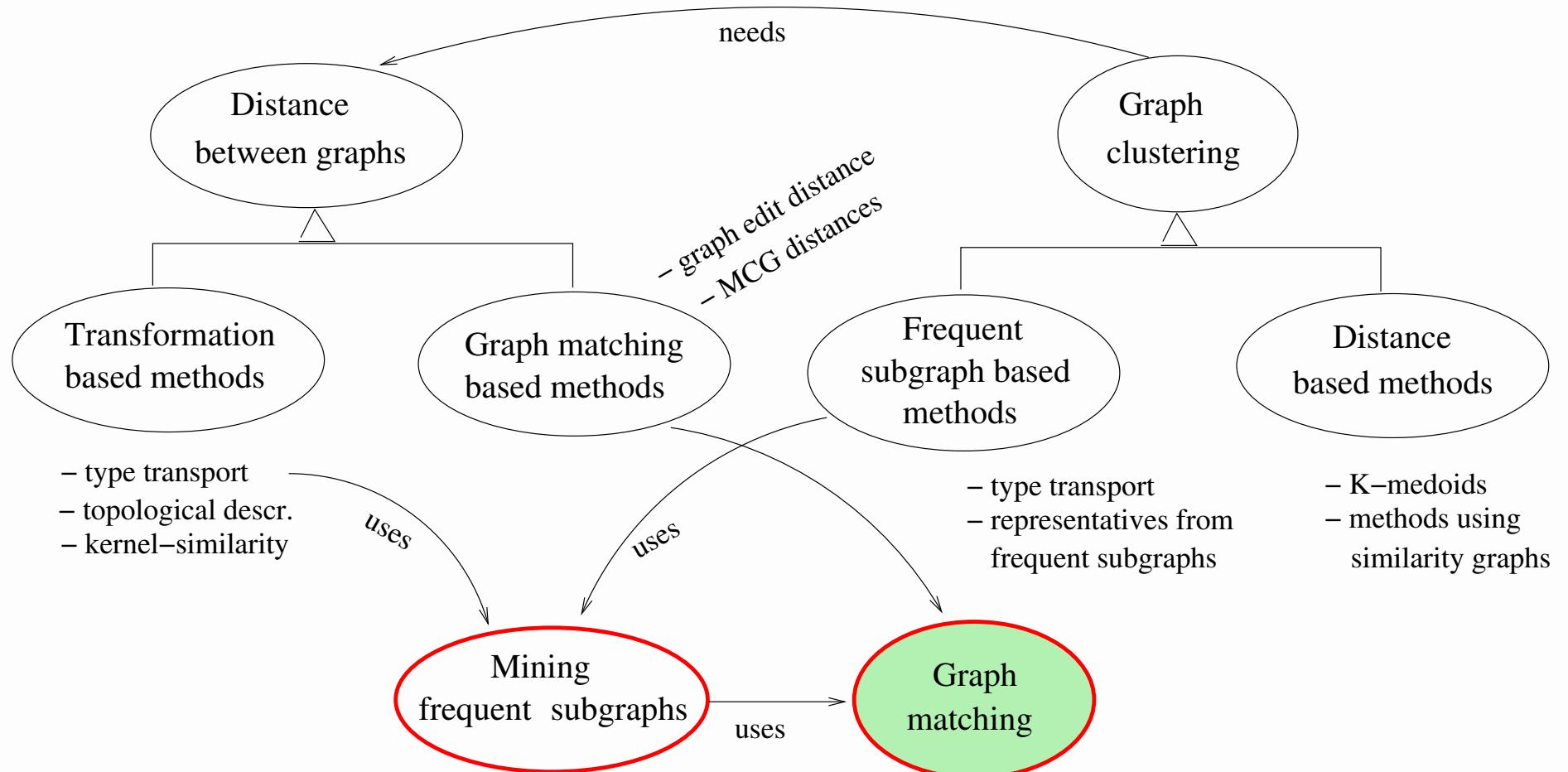
Further reading Aggarwal 18.5.5 and 6.8

Summary

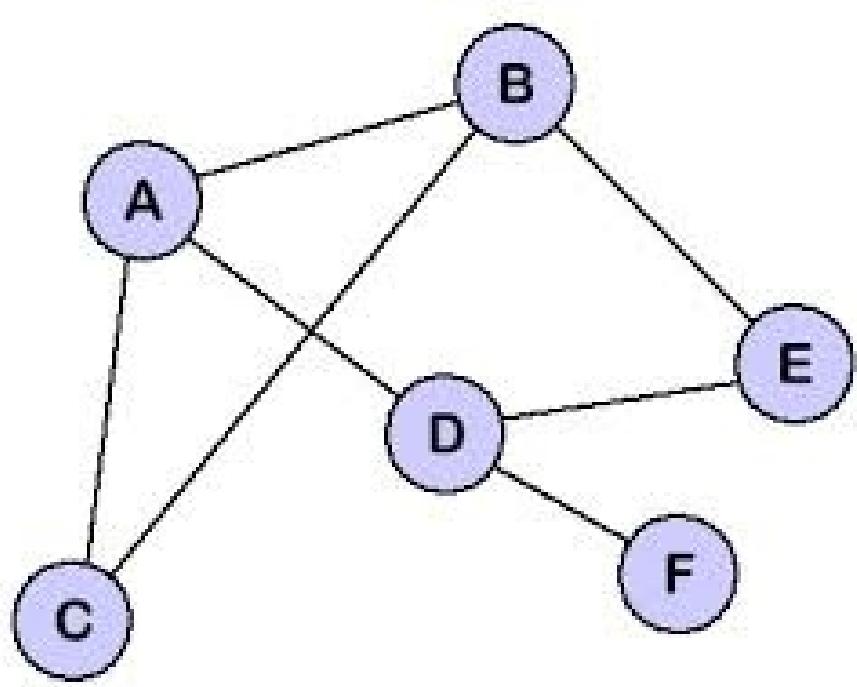
- Content-based recommendations: evaluate similarity between text descriptions and utilize only the user's own ratings
- Collaborative filtering: utilize all users' ratings
 - many approaches: neighbourhood-based, graph-based, clustering-based, latent factor-based
 - often 2 steps: determine a peer group and then calculate predicted ratings
 - sometimes 1 step: choose recommended items directly by PageRank or use matrix factorization

Further reading: Desrosiers and Karypis: A comprehensive survey of neighborhood-based recommendation methods. In Recommender Systems Handbook, 2011.

Mining database of multiple graphs



Graph notations



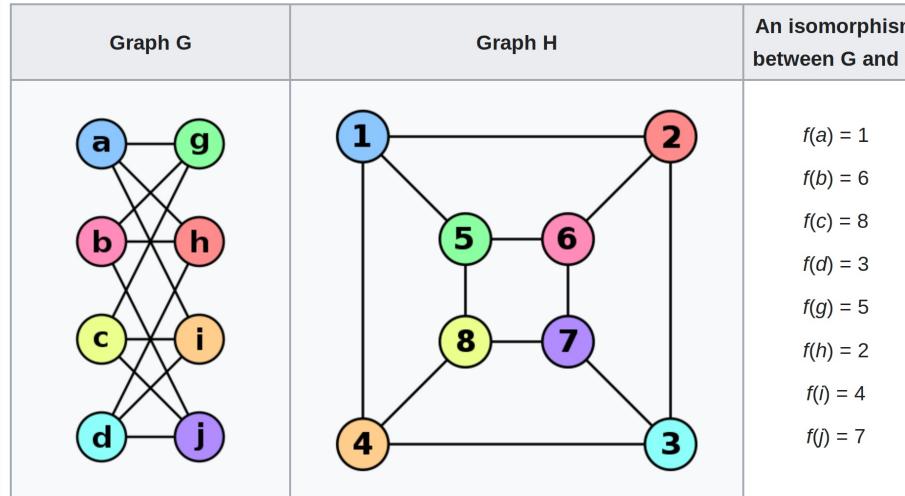
- $G = (V, E)$ graph
- $V = \{v_1, \dots, v_n\}$ = set of vertices or nodes
- $|V|$ = number of nodes
- node label $l(v_i)$
- $E = \{e_1, \dots, e_m\}$ = set of edges, $e_i = (v, u)$, $v, u \in V$
- $|E|$ = number of edges

Now we assume that edges undirected and don't have labels

Graph isomorphism of unlabelled graphs

Two unlabelled graphs $G_1 = (V, E)$ and $G_2 = (U, F)$ are **isomorphic** or **matching** if there is an edge-preserving bijection $f : V \rightarrow U$ such that for any $v_1, v_2 \in V$:

$(v_1, v_2) \in E \Leftrightarrow (f(v_1), f(v_2)) \in F.$



Matching can be presented as $\mathcal{M} = \{(v, u) \mid v \in V, u \in U, u = f(v)\}$

Image source https://en.wikipedia.org/wiki/Graph_isomorphism

Graph isomorphism of labelled graphs

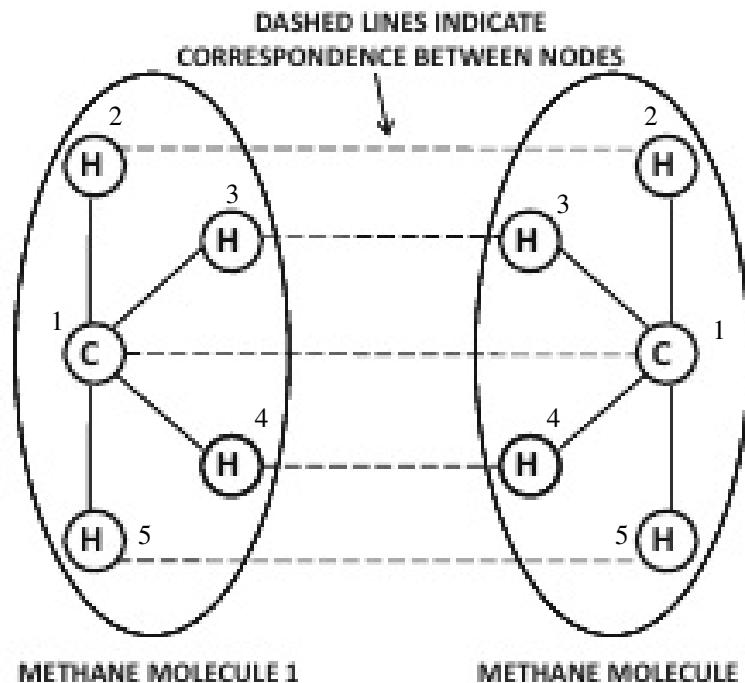
Two labelled graphs $G_1 = (V, E)$ and $G_2 = (U, F)$ are **isomorphic**, if there is an edge- and label-preserving **bijection** $f : V \rightarrow U$ such that

- (i) Corresponding nodes have same labels: $\forall v \in V$ and $f(v) \in U$ $l(v) = l(f(v))$.
- (ii) An edge between matched nodes exists in G_1 iff the corresponding edge exists in G_2 : $\forall v_1, v_2 \in V$:
 $(v_1, v_2) \in E \Leftrightarrow (f(v_1), f(v_2)) \in F$.

Note: no polynomial time algorithms are known (except special cases)

There can be many matchings!

Two matchings for molecules 1 and 2. Totally $4! = 24$ matchings!



$$\mathcal{M} = \{(1, 1), (2, 2), (3, 3), (4, 4), (5, 5)\}$$

$$\mathcal{M} = \{(1, 1), (2, 2), (3, 4), (4, 3), (5, 5)\}$$

Image source: Aggarwal Fig. 17.2

Subgraph isomorphism

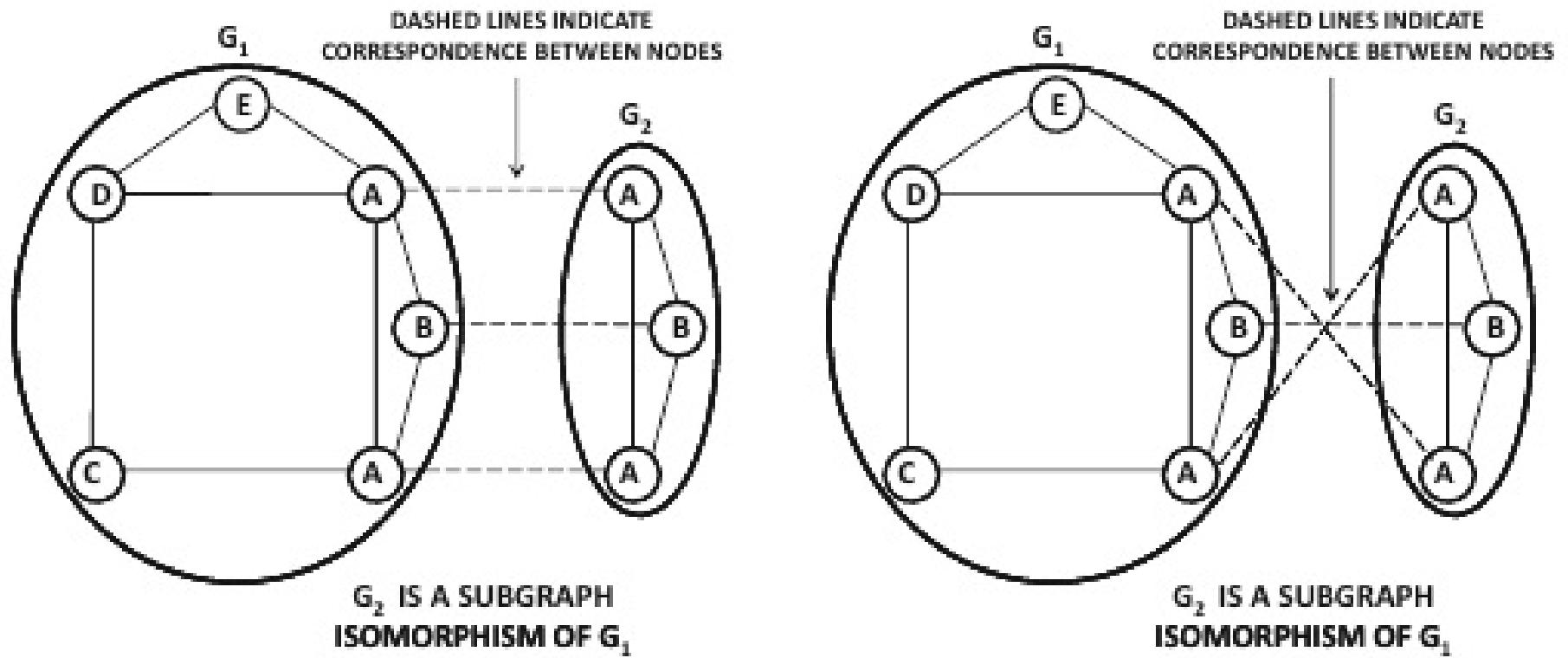
Does a certain **query graph** \mathbf{G}_q match a part of another graph \mathbf{G} ?

Query graph $\mathbf{G}_q = (\mathbf{V}, \mathbf{E})$ is a **subgraph isomorphism** of $\mathbf{G} = (\mathbf{U}, \mathbf{F})$, if there is an **injection** $f : \mathbf{V} \rightarrow \mathbf{U}$ such that

- (i) For all $v \in \mathbf{V}$ there is $f(v) \in \mathbf{U}$ such that
 $l(v) = l(f(v))$; and
- (ii) For any $v_1, v_2 \in \mathbf{V}$: $(v_1, v_2) \in \mathbf{E} \Leftrightarrow (f(v_1), f(v_2)) \in \mathbf{F}$.

Notes: 1) Usually it is required that the graphs are connected.
2) Sometimes a weaker condition suffices for (ii):
if $(v_1, v_2) \in \mathbf{E} \Rightarrow (f(v_1), f(v_2)) \in \mathbf{F}$

Subgraph isomorphism: example



- Algorithm: see Aggarwal Ch 17.2.1

Image source: Aggarwal Fig. 17.3

Maximum common subgraph (MCG)

Problem: Given G_1 and G_2 , find $G_0 = (V_0, E_0)$ such that

- (i) G_0 is a subgraph isomorphism of both G_1 and G_2 and
- (ii) $|V_0|$ is as large as possible.

- + useful for comparing graphs
 - distances between graphs
 - frequent subgraph discovery
- NP -hard problem (like subgraph isomorphism)

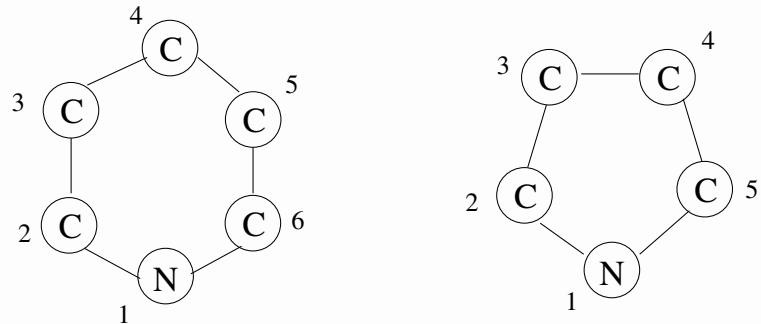
Algorithm for $MCG(G_1, G_2)$

```
function  $MCG(\mathbf{G}_1, \mathbf{G}_2, \mathcal{M}, \mathcal{M}_{best})$ 
/* Create candidates for matching node pairs */
 $C = \{(v, u) \mid v \in \mathbf{V}, u \in \mathbf{U}, l(v) = l(u), (v, u) \notin \mathcal{M}\}$ 
Prune  $C$ 
/* Recursion: */
for all  $(v, u) \in C$ 
  if valid( $\mathcal{M}, (v, u)$ ) // is  $(u, v)$  a valid extension?
     $\mathcal{M}_{best} = MCG(\mathbf{G}_1, \mathbf{G}_2, \mathcal{M} \cup (v, u), \mathcal{M}_{best})$ 
  if ( $|\mathcal{M}| > |\mathcal{M}_{best}|$ )
    return  $\mathcal{M}$ 
  else return  $\mathcal{M}_{best}$ 
```

$\mathbf{G}_1 = (\mathbf{V}, \mathbf{E})$, $\mathbf{G}_2 = (\mathbf{U}, \mathbf{F})$
Call: $MCG(\mathbf{G}_1, \mathbf{G}_2, \emptyset, \emptyset)$

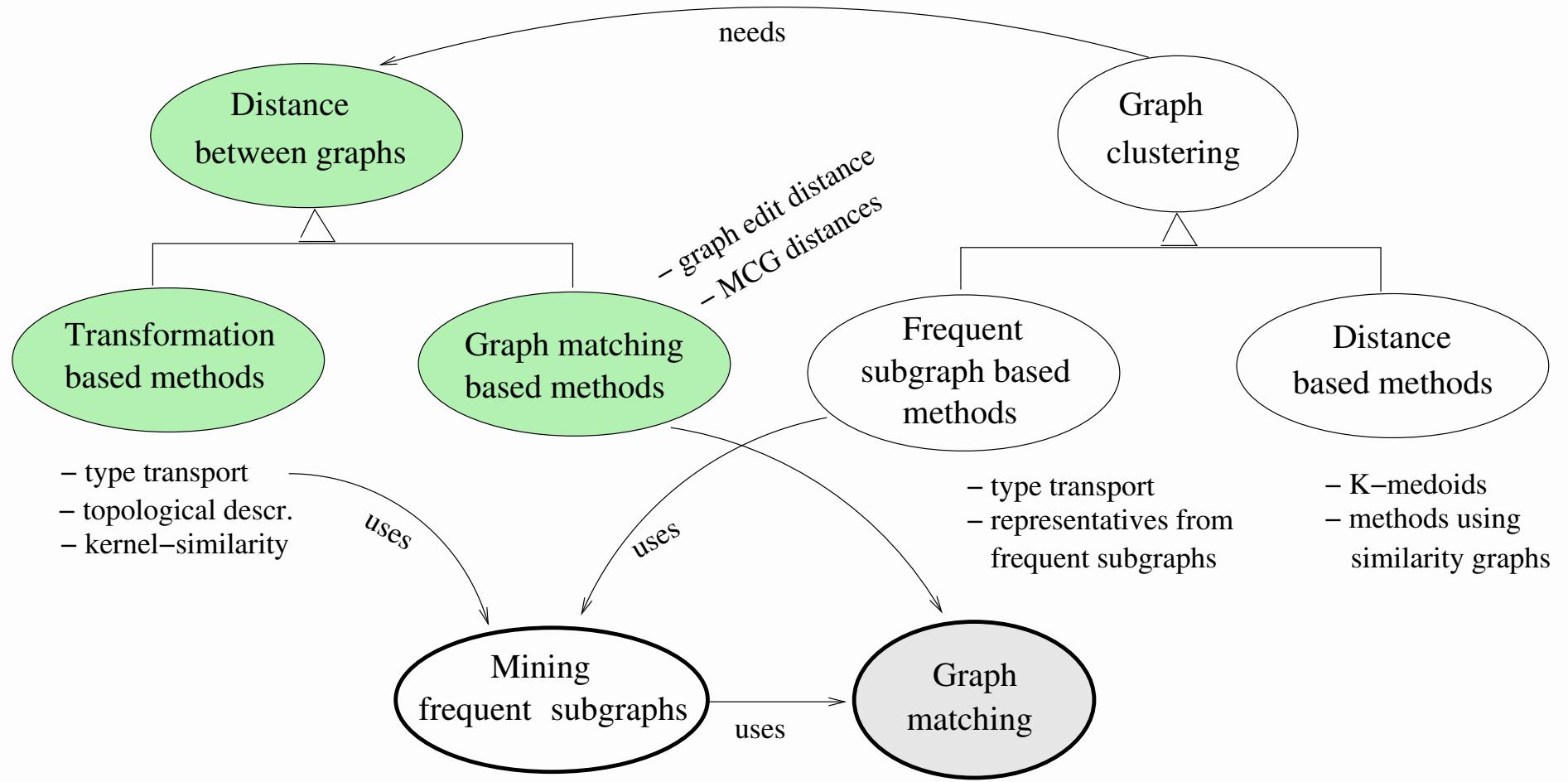
Algorithm: valid extensions

```
valid( $\mathcal{M}, (v, u)$ )  
if ( $\exists u_2 \in \mathbf{U} : ((u, u_2) \in F) \& \& ((v_2, u_2) \in \mathcal{M}) \& \& ((v, v_2) \notin E)$ ) or  
    ( $\exists v_2 \in \mathbf{V} : ((v, v_2) \in E) \& \& ((v_2, u_2) \in \mathcal{M}) \& \& ((u, u_2) \notin F)$ )  
    return 0  
else return 1
```



E.g., $\mathcal{M} = \{(1, 1), (2, 2), (3, 3), (6, 5)\}$.
(4, 4) is invalid extension – why?
Is there any valid extension?

Next to distances



Distances based on maximum common subgraphs

- Let's assume graph size = number of nodes, i.e., for $G = (V, E)$ notate $|G| = |V|$
- Let $MCS(G_1, G_2)$ =maximum common subgraph of G_1 and G_2 and $|MCS(G_1, G_2)|$ =its size

1. Unnormalized non-matching measure:

$$U(G_1, G_2) = |G_1| + |G_2| - 2 \cdot |MCS(G_1, G_2)|$$

- = number on non-matching nodes
- Problem: what if graphs have very different sizes?

Normalized MCS distances

2. Union-normalized distance $Udist \in [0, 1]$

$$Udist(\mathbf{G}_1, \mathbf{G}_2) = 1 - \frac{|MCS(\mathbf{G}_1, \mathbf{G}_2)|}{|\mathbf{G}_1| + |\mathbf{G}_2| - |MCS(\mathbf{G}_1, \mathbf{G}_2)|}$$

= number of non-matching nodes normalized by union size

3. Max-normalized distance $Mdist \in [0, 1]$

$$Mdist(\mathbf{G}_1, \mathbf{G}_2) = 1 - \frac{|MCS(\mathbf{G}_1, \mathbf{G}_2)|}{\max\{|\mathbf{G}_1|, |\mathbf{G}_2|\}}$$

- metric

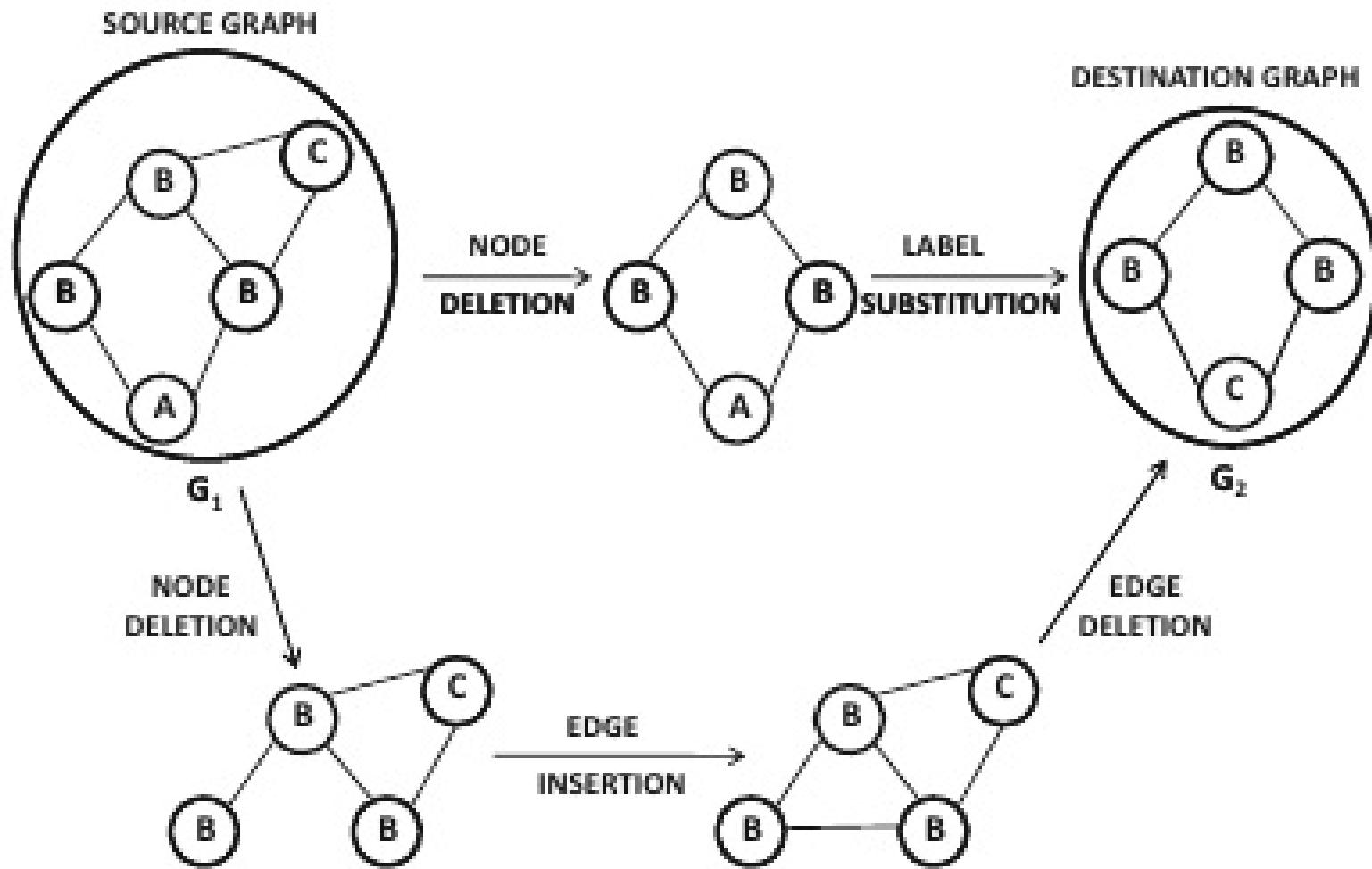
MCS distances can be computed efficiently **only for small graphs!**

Graph edit distance

What is the minimum cost of edit operations to transform G_1 to G_2 ?

- (i) node insertion
 - (ii) node deletion (deletes also incident edges)
 - (iii) edge insertion
 - (iv) edge deletion
 - (v) label substitution of nodes
-
- application-specific costs
 - may be exponentially many possible edit paths!
 - NP -hard

Graph edit distance: example

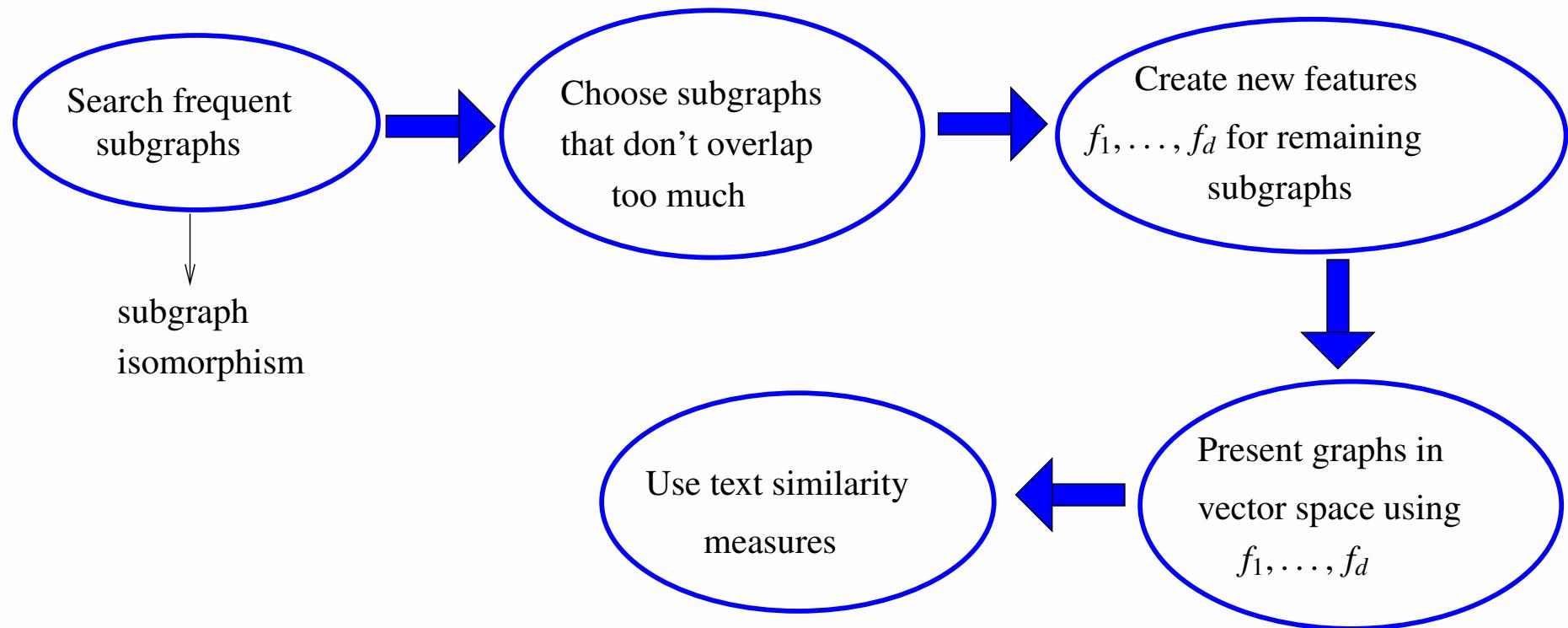


Transformation-based distances

Idea: Transform graphs into a new space where distances are easier to calculate

- a) Type transport using frequent subgraphs
- b) Topological descriptors
- c) Kernel similarity

Type transport using frequent subgraphs



f_i = number of times
ith subgraph occurs in G
or binary or tf-idf presentation

involves an NP -hard subproblem

Topological descriptors

Idea: calculate different kinds of indices from graphs \Rightarrow new numerical features \Rightarrow Use distances for numerical data

- structural information lost
- utility domain-specific (e.g., good in chemical domain)
- e.g., Wiener index:

$$W(\mathbf{G}) = \sum_{v,u \in V} d(v, u)$$

$d(v, u)$ =length of shortest path from v to u

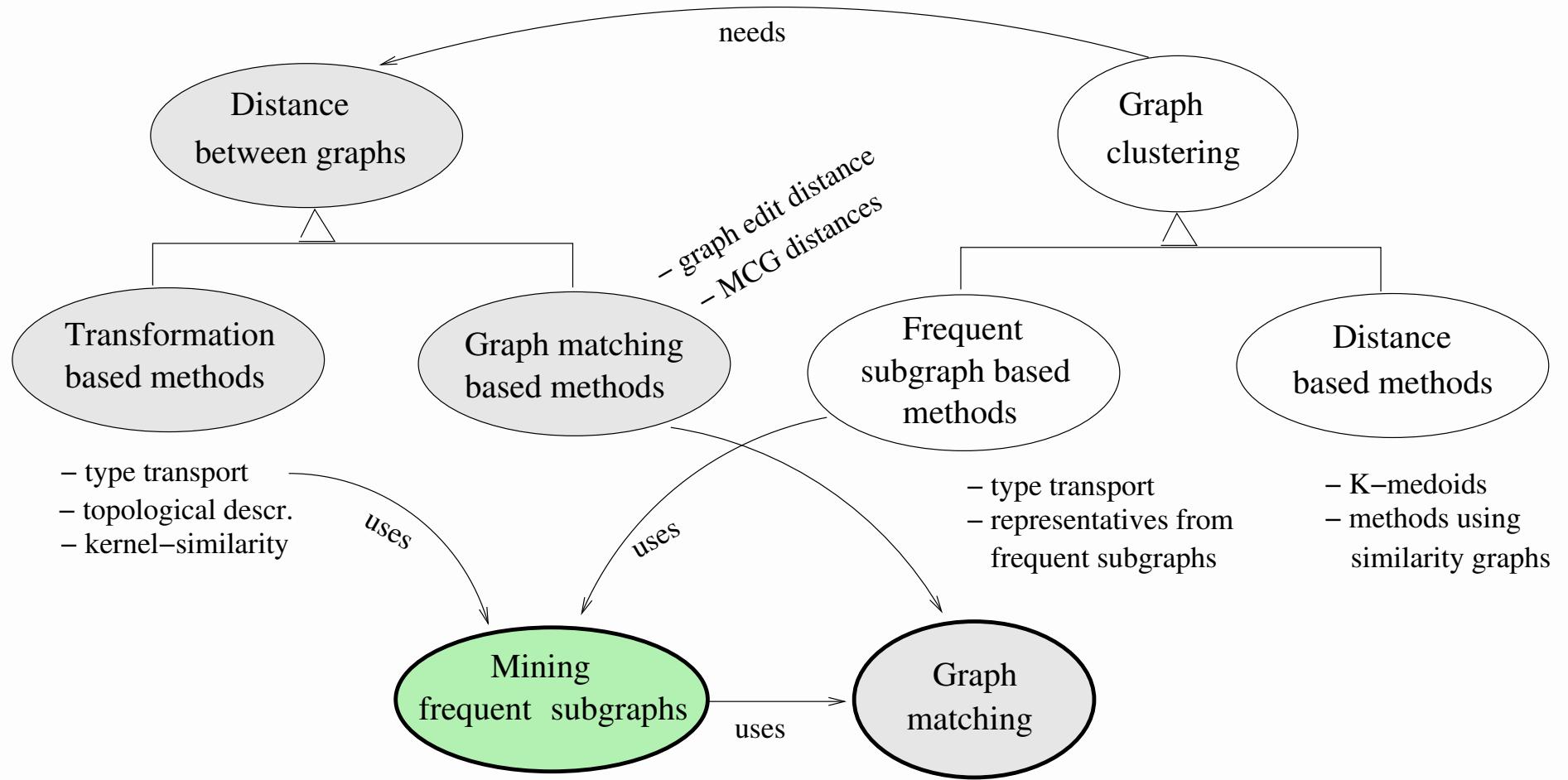
- more in Aggarwal Ch 17.3.2

Kernel similarity

Idea:

- Assume transformation Φ such that similarity of G_1 and G_2 can be measured by $\Phi(G_1) \cdot \Phi(G_2)$
- Design **kernel function** K such that $K(G_1, G_2) = \Phi(G_1) \cdot \Phi(G_2)$ and use it as a similarity measure (without transformation)
- e.g. shortest path kernel ($O(n^4)$) and random walk kernel ($O(n^6)$)
- practical for small graphs
- more in Aggarwal Ch 17.3.3

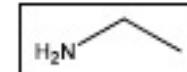
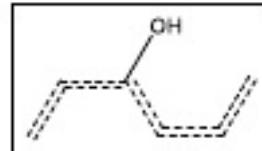
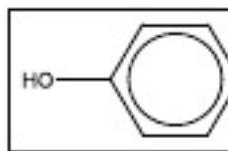
Next to frequent subgraph discovery



Frequent subgraph discovery: Motivation

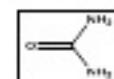
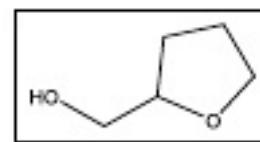
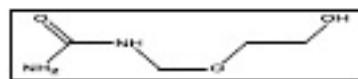
Most Discriminating Subgraphs

Predict:



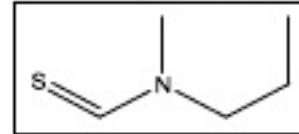
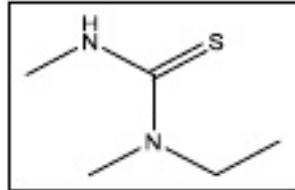
toxicity of compounds

(a) On Toxicology (PTC) Dataset



anti-HIV activity

(b) On AIDS Dataset



binding ability with
Anthrax toxin

(c) On Anthrax Dataset

Image source: <https://slideplayer.com/slide/5894097/>

Frequent subgraph discovery

Task: Given graph database, search frequent subgraphs given threshold min_{fr} .

- Search idea: utilize **monotonicity of frequency!**
- If G_1 is a subgraph of G_2 , then $fr(G_1) \geq fr(G_2)$
- similar algorithms than for frequent itemsets, but more complex
- two variants: size of graph may refer to a) number of nodes b) number of edges
⇒ how new candidates are generated

GraphApriori algorithm

\mathcal{F}_i = frequent subgraphs of size i , C_i = candidates

- $\mathcal{F}_1 = \{\mathbf{G} \mid \text{where } |\mathbf{G}| = 1, P(\mathbf{G}) \geq \min_{fr}\}; i = 1$
- while $\mathcal{F}_i \neq \emptyset$
 - generate candidates C_{i+1} from \mathcal{F}_i
 - prune $\mathbf{G} \in C_{i+1}$ if \mathbf{G} has a subgraph \mathbf{G}' such that $|\mathbf{G}'| = i$ and $\mathbf{G}' \notin \mathcal{F}_i$ (=monotonicity criterion)
 - count frequencies $fr(\mathbf{G}), \mathbf{G} \in C_{i+1}$
 - set $\mathcal{F}_{i+1} = \{\mathbf{G} \in C_{i+1} \mid P(\mathbf{G}) \geq \min_{fr}\}$
 - $i = i + 1$
- return $\cup_i \mathcal{F}_i$

GraphApriori: Candidate generation

For all $\mathbf{G}_1, \mathbf{G}_2 \in \mathcal{F}_i$, $|\mathbf{G}_1| = |\mathbf{G}_2| = i$

1. determine if \mathbf{G}_1 and \mathbf{G}_2 have a common subgraph \mathbf{G}_0 of size $i - 1$
 - may be many isomorphic matchings \Rightarrow **many alternative \mathbf{G}_0 s!**
2. for each \mathbf{G}_0 create candidate graphs of size $i + 1$
 - **node-based:** include all common + 2 non-matching nodes (with extra edge or not)
 - **edge-based:** include all $i - 1$ common edges and 2 unique edges (with extra node or not)
 - same subgraphs may be generated multiple times \Rightarrow redundancy checking

Example of node-based join

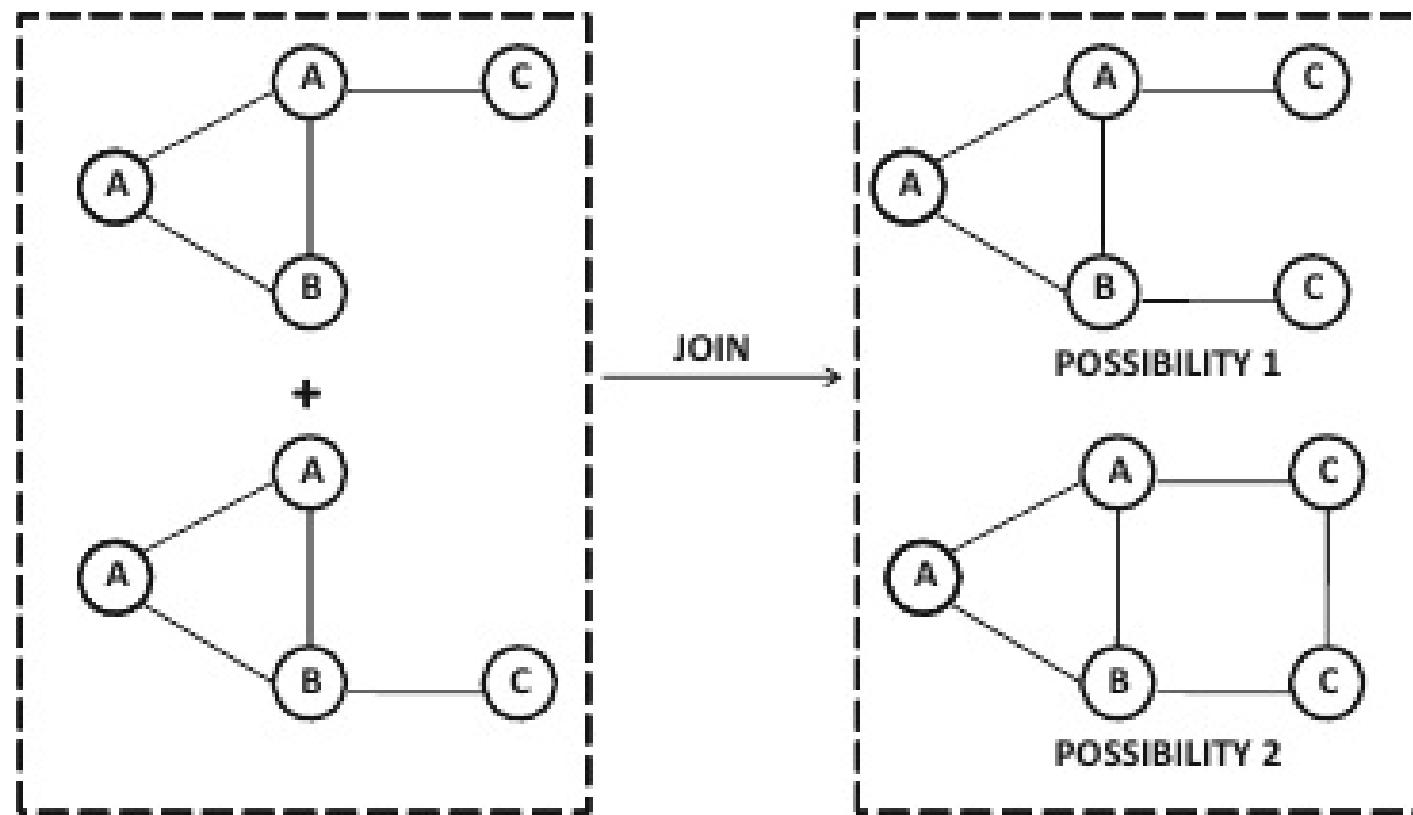


Image source: Aggarwal Fig. 17.12

Example of edge-based join

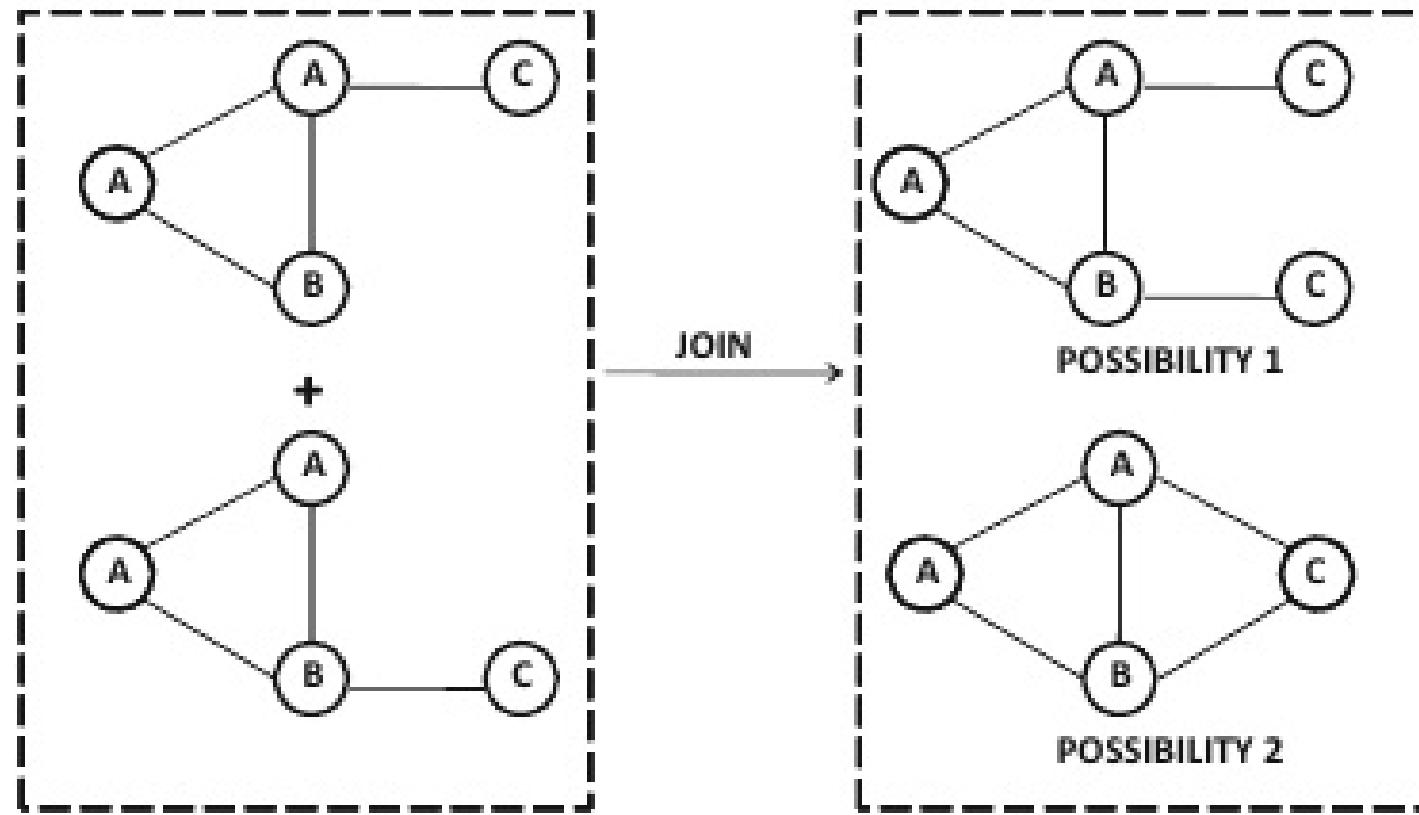


Image source: Aggarwal Fig. 17.13

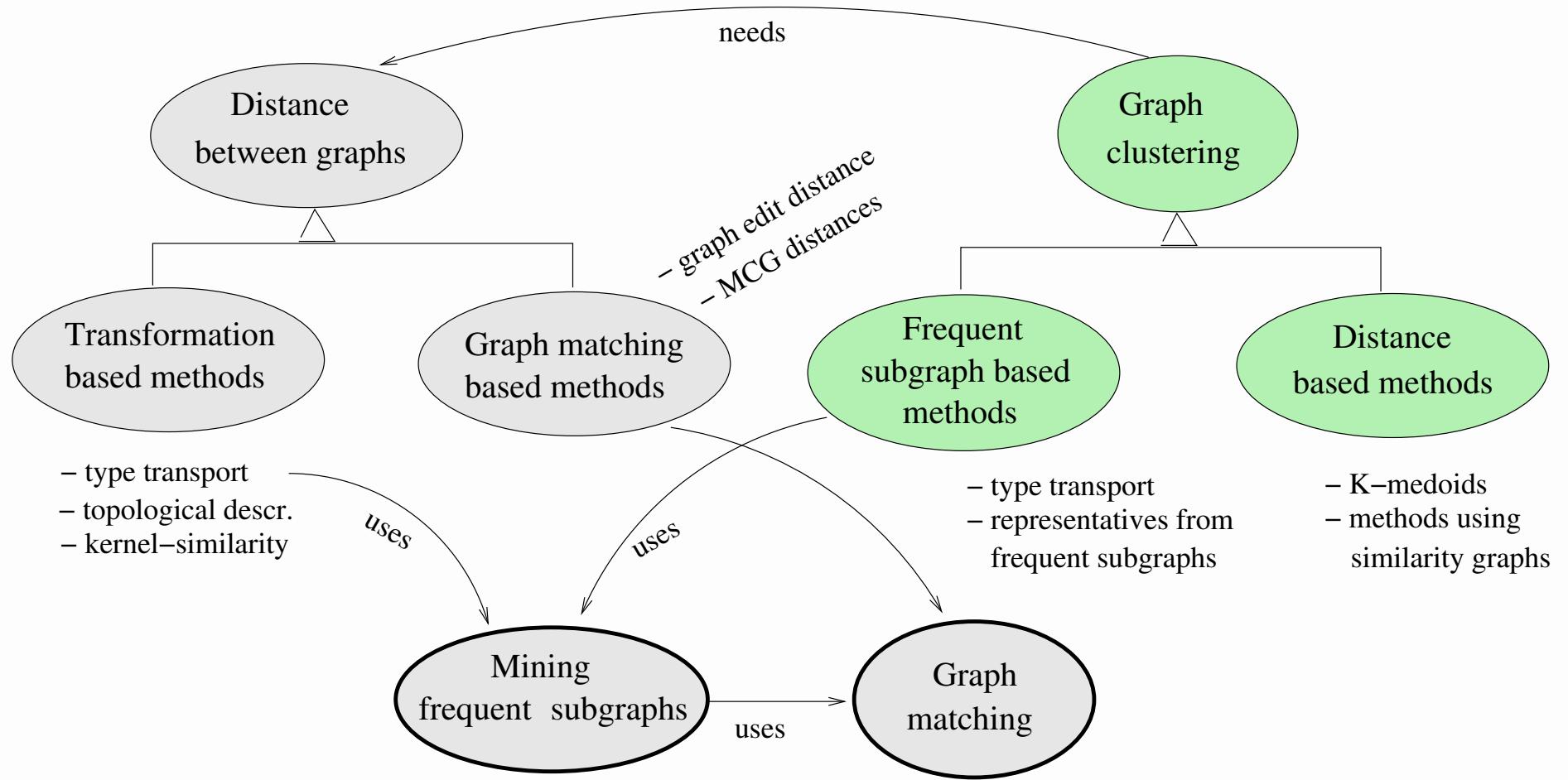
Why this is heavy?

- number of candidate patterns may be huge!
- subgraph isomorphism to identify pairs of subgraphs for joining
- graph isomorphism for redundancy checking
- subgraph isomorphism for monotonicity pruning
- subgraph isomorphism for frequency counting

Easier if

- many unique node labels
- only small subgraphs are searched
- edge-based join is used (usually less candidates)

Next to graph clustering



Distance-based clustering methods

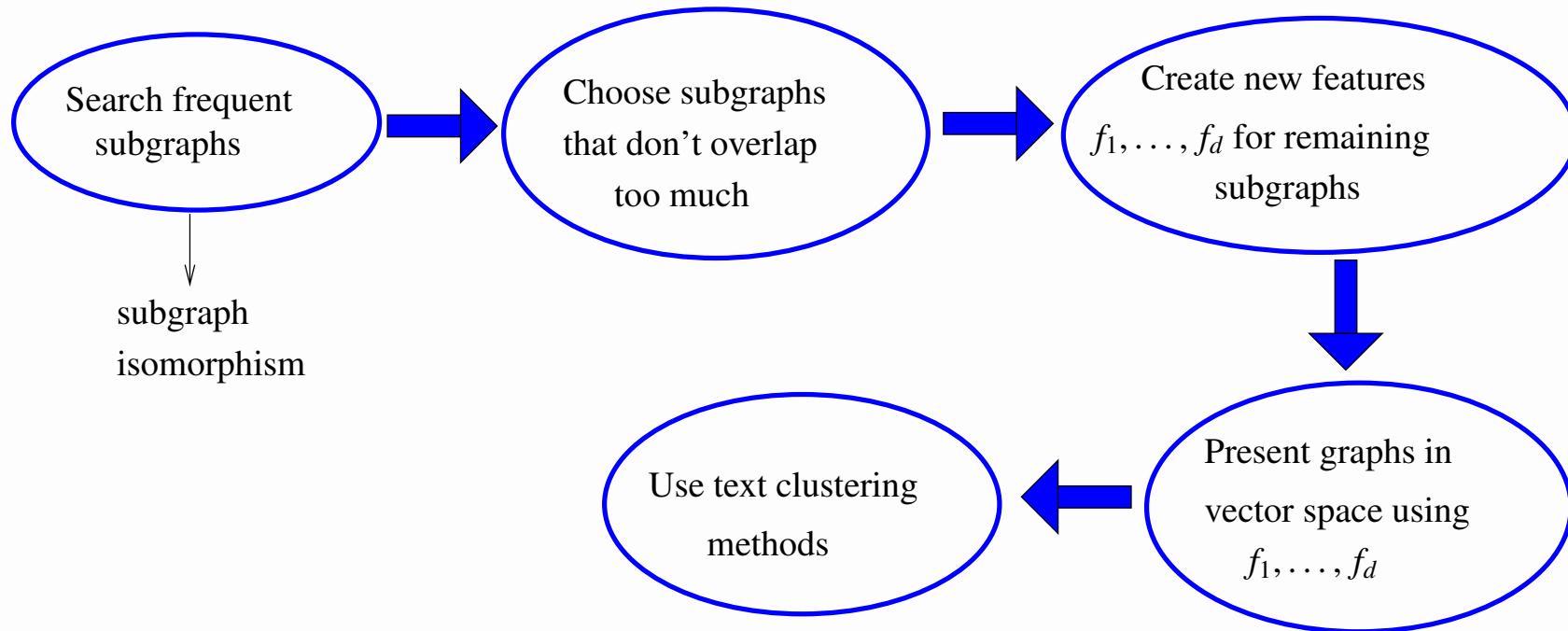
Common approaches:

1. K -medoids (needs just a distance function)
2. Spectral and other graph-based methods
 - construct a nearest neighbour/similarity graph of graph objects
 - cluster nodes of the new graph

Remember: graph distance measures very expensive to compute! → suitable for smaller graphs

Methods based on frequent subgraphs

Approach 1. Type transport: graphs → multidimensional



f_i = number of times

i th subgraph occurs in G

or binary or tf-idf representation

involves an *NP*-hard subproblem

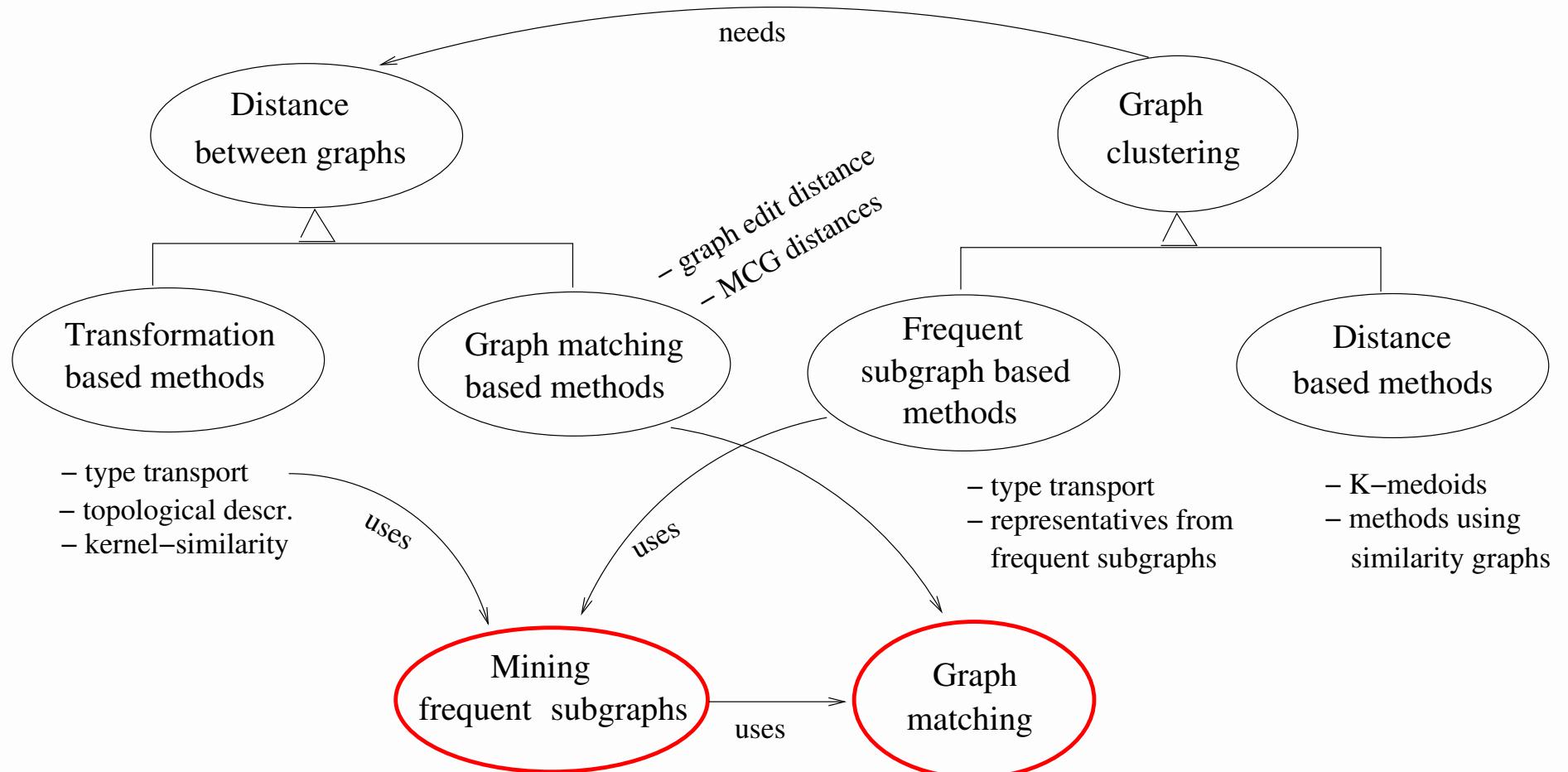
Methods based on frequent subgraphs

Approach 2. XProj: cluster representatives = **sets** of frequent subgraphs

- Initialization: Create K random clusters C_1, \dots, C_K
- for all C_i : \mathcal{F}_i = set of frequent subgraphs (of a given size) from C_i
- repeat until convergence:
 - assign each \mathbf{G}_j to C_i where $sim(G_j, \mathcal{F}_i)$ largest
 - for all C_i determine new \mathcal{F}_i

$sim(G_j, \mathcal{F}_i)$ = fraction of frequent graphs in \mathcal{F}_i that occur in \mathbf{G}_j

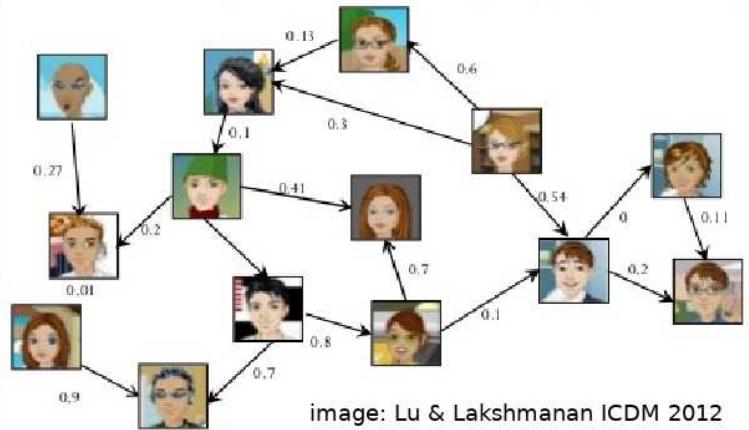
Summary



Overview of social network analysis

Emphasis:

- Properties of social networks
- Important analysis tasks
- Useful measures and solution principles



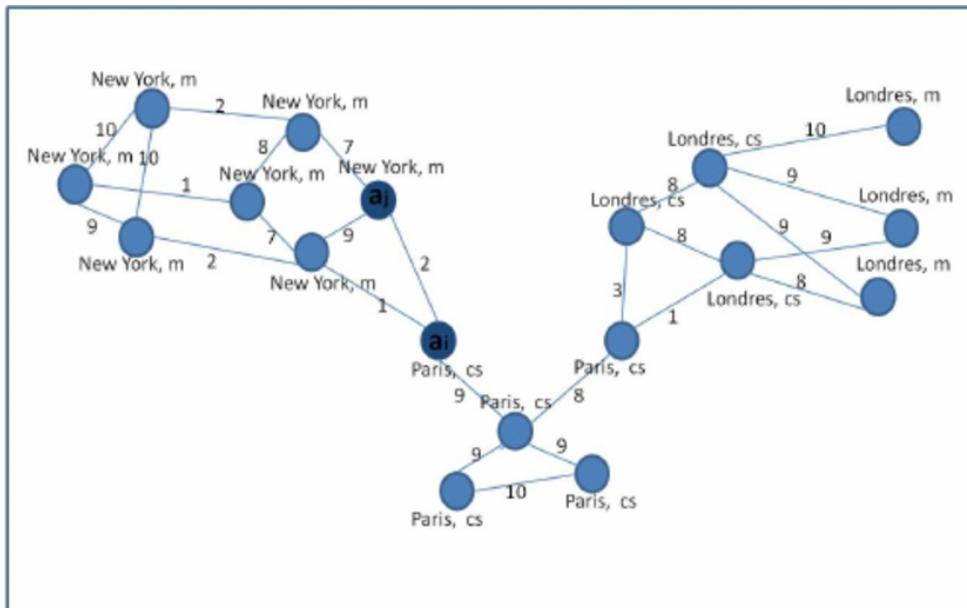
More on course CS-E5740 Complex Networks

I Introduction: Types of social networks

- online networks (Twitter, LinkedIn, Facebook)
 - indirect communication networks (telecommunications, email, chat messages)
 - media sharing sites (Youtube, Instagram, Tiktok)
 - interaction networks in professional communities (e.g., citation networks between researchers)
 - networks recorded in observational studies (e.g., interactions in a class room, between animals)
- + many more! but not always data

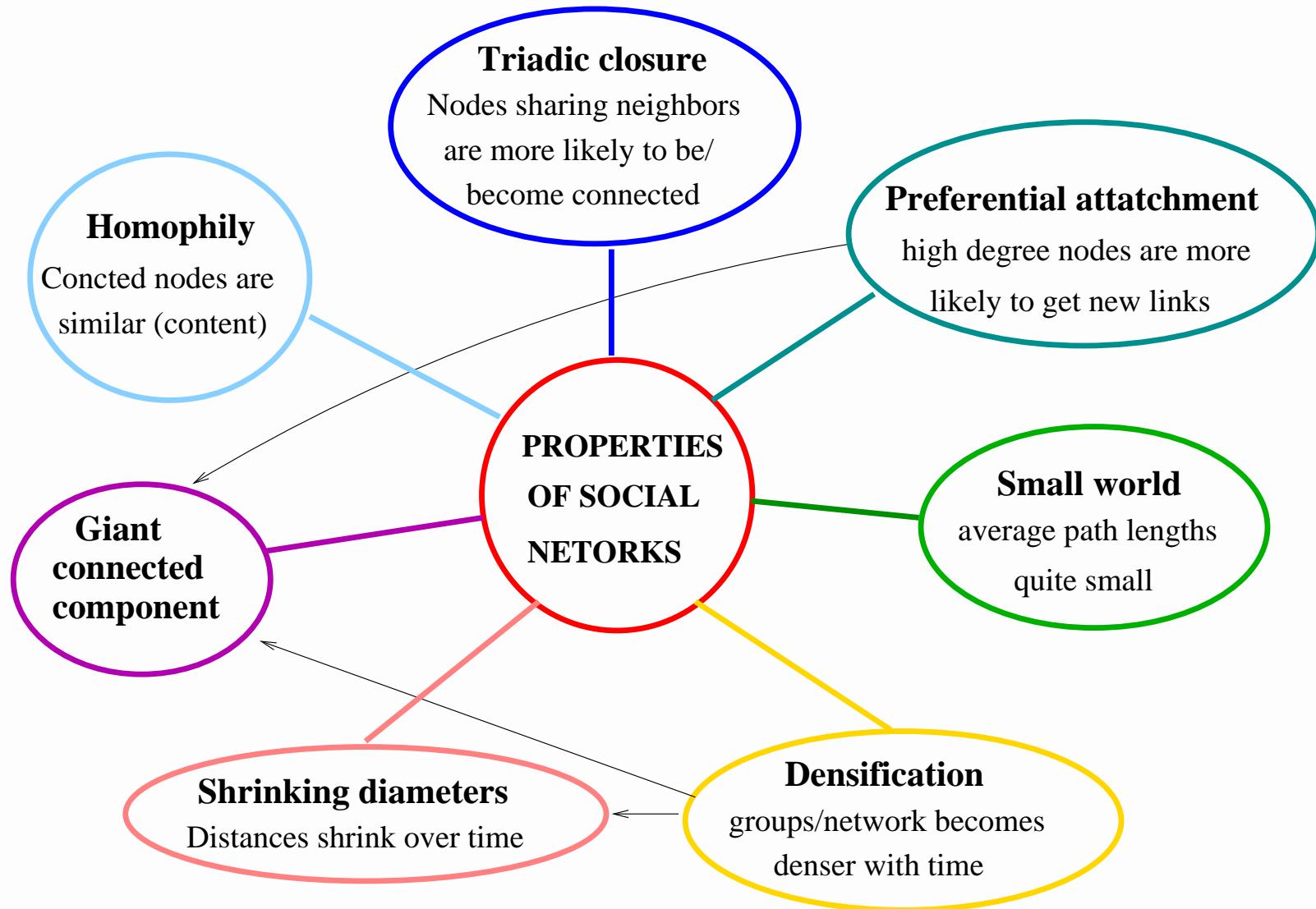
Presentation as a graph $G = (V, E)$

- V set of nodes corresponding to **actors**
 - may have labels or content (attributes, documents)
- E set of edges corresponding to links
 - undirected (friendship) or directed (“following”)
 - may have weights w_{ij}



Example by Zardi et al. (2014)
node attributes: city and education
edge weight = number of exchanged messages

Basic properties



Analysis tasks

- Social influence analysis (influential nodes and influence spread)
- Community detection (graph clustering)
- Link prediction (predict future links between nodes)
- Collective classification (predict missing node labels)

II Social influence analysis

Which nodes have most influence? How influence (information, ideas, opinions) spreads?

A valuable advertising channel!

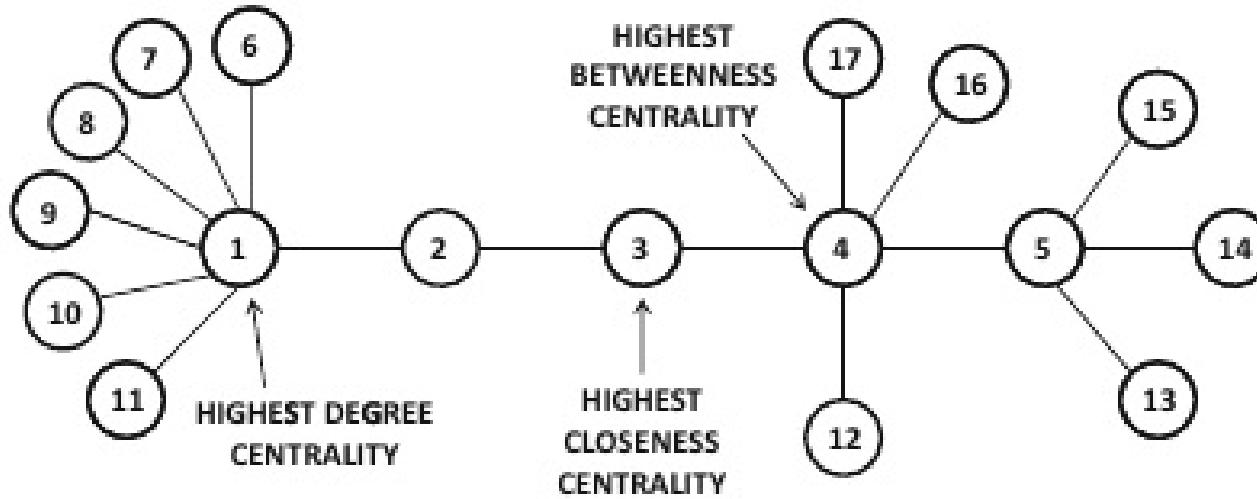
1. Measures for evaluating which nodes are influential:
 - **centrality** of a node in an undirected graph
 - **prestige** of a node in a directed graph
2. **Influence propagation or diffusion models**
 - given influence weights on edges and a model to evaluate total influence of a set of nodes
 - determine a set of **seed nodes** such that spread of influence is maximal

Measures for the centrality of node v

Degree centrality: $C_D(v) = \frac{\text{Degree}(v)}{n-1}$

Closeness centrality: $C_C(v) = \frac{1}{\text{avg}_{u \in V, u \neq v} \{ \text{Dist}(v, u) \}} = \frac{n-1}{\sum_{u \in V, u \neq v} \text{Dist}(v, u)}$

Betweenness centrality: $C_B(v) = \frac{\sum_{u, w \in V, u \neq w} \frac{\#\{\text{shortest-paths}(u, w) \text{ through } v\}}{\#\{\text{shortest-paths}(u, w)\}}}{\binom{n}{2}}$



Note: $C_c(v)$ may be calculated such $v \neq u, v \neq w$. Image: Aggarwal Fig. 19.1

III Community detection: cluster the graph

Given $\mathbf{G} = (\mathbf{V}, \mathbf{E})$. Each edge (v_i, v_j) has weight w_{ij}

- if cost c_{ij} , transform, e.g. by $w_{ij} = \frac{1}{c_{ij}}$ ($c_{ij} \neq 0$)

Common objective: Cluster \mathbf{V} into groups $\mathbf{V}_1, \dots, \mathbf{V}_K$ such that the edge-cut cost

$$cost(\mathbf{V}_1, \dots, \mathbf{V}_k) = \sum_{(v_i, v_j) \in E, v_i \in \mathbf{V}_p, v_j \in \mathbf{V}_q, p \neq q} w_{ij}$$

is minimal.

- many variants and extra constraints!
- in general NP -hard problem, but polynomially solvable, if $\forall i, j : w_{ij} = 1$, $K = 2$ and no balancing requirements

Example

Clustering based on both structural and content-based features

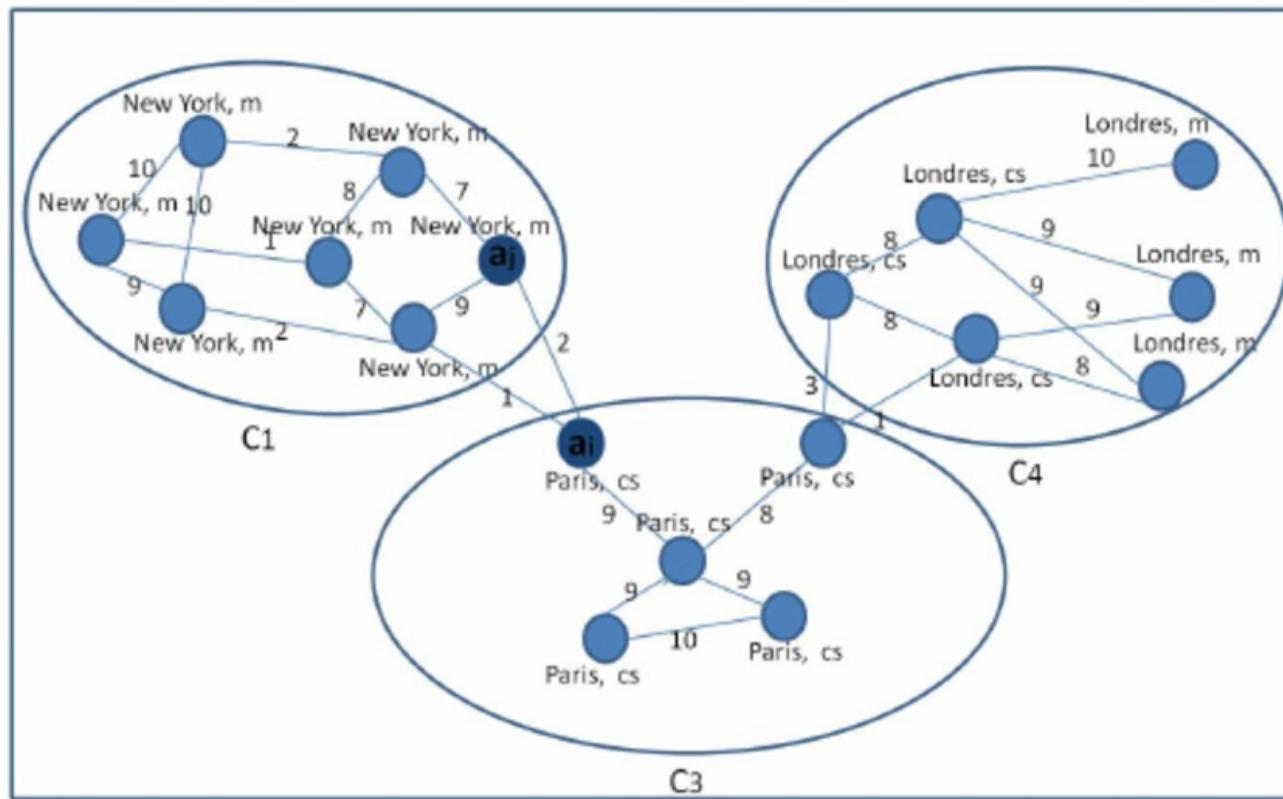


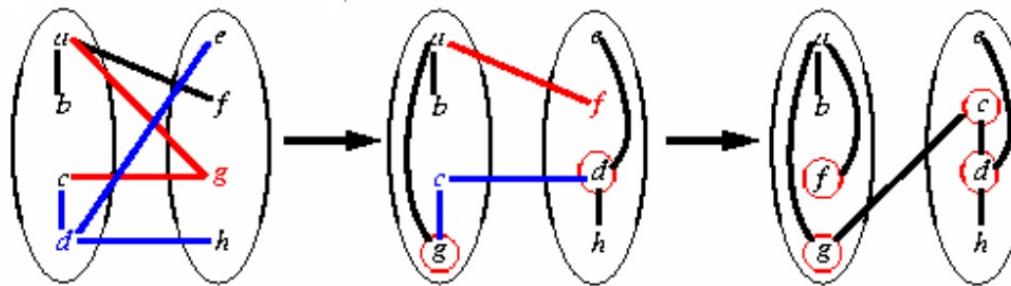
Image source: Zardi et al.: A Multi-agent homophily-based approach for community detection in social networks, ICTAI 2014

Some community detection methods

1. Spectral clustering

2. Kernighan-Lin: balanced 2-way partitioning

- at each iteration, test a set of possible swap sequences and choose the one with greatest improvement



Step #	Vertex pair	Cost reduction	Cut cost
0	-	0	5
1	{d, g}	3	2
2	{c, f}	1	1
3	{b, h}	-2	3
4	{a, e}	-2	5

Image source: Chang 2004

3. *Girwan-Newman algorithm*

- remove “bridge edges” until K connected components remain
- edges with high **betweenness**: large proportion of shortest paths go through them

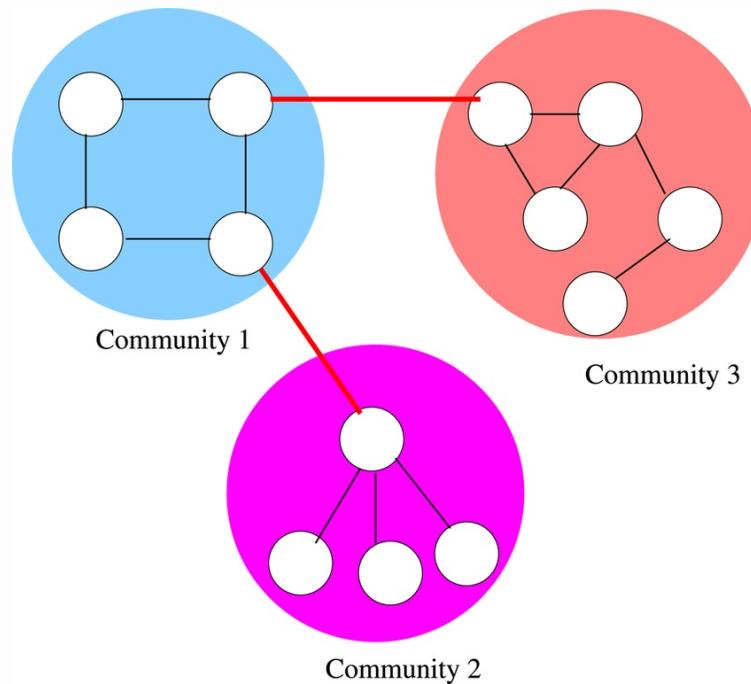


Image source: Namtirtha et al. 2023

4. METIS algorithm

1. Coarsen the graph by combining tightly interconnected nodes and parallel edges
2. Partition the coarsened representation (easier)
3. Refine partitioning when expanding graphs back

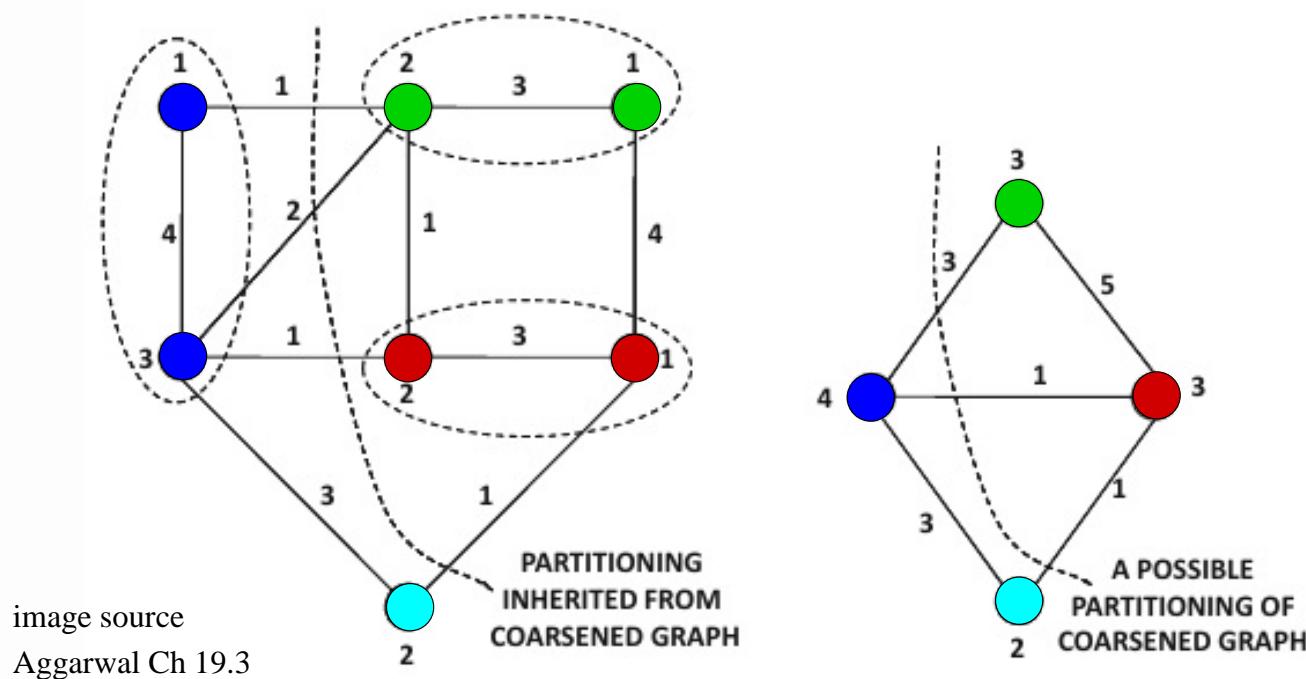


Image source: Aggarwal Fig. 19.6

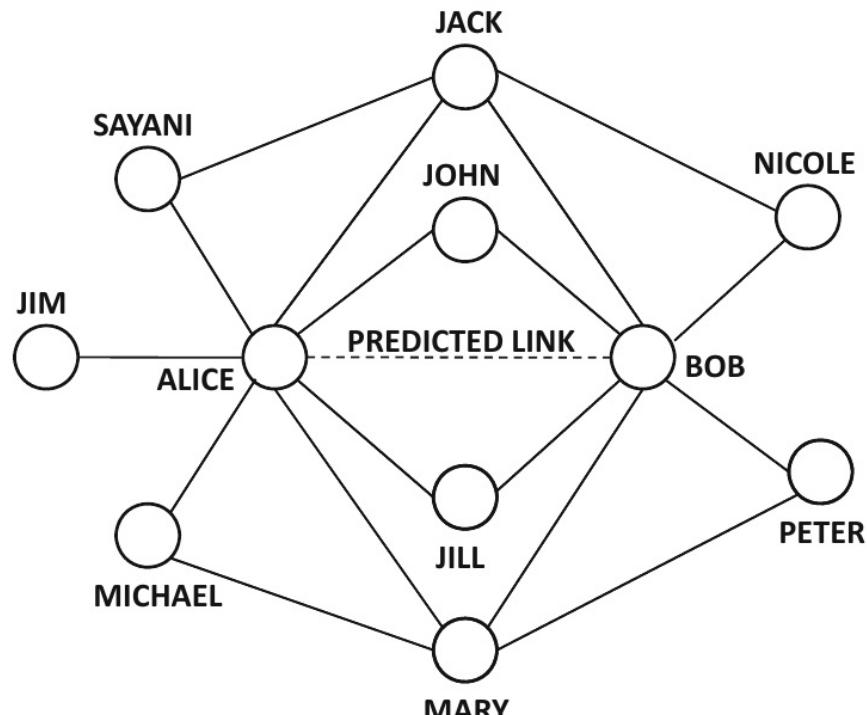
IV Link prediction and node similarity

Utilize especially **structural** features!

Approaches:

1. Evaluate potential connections with **node similarity measures**
 - + easy and fast to compute
2. Learn a classifier for predicting links or their absence
 - + more accurate
 - computationally more expensive
3. Use missing value estimation methods (like matrix factorization)

Neighbourhood-based node similarity measures



(a) Many common neighbors
between Alice and Bob

(normalized) number of common neighbours

- not good, if number of common neighbours small

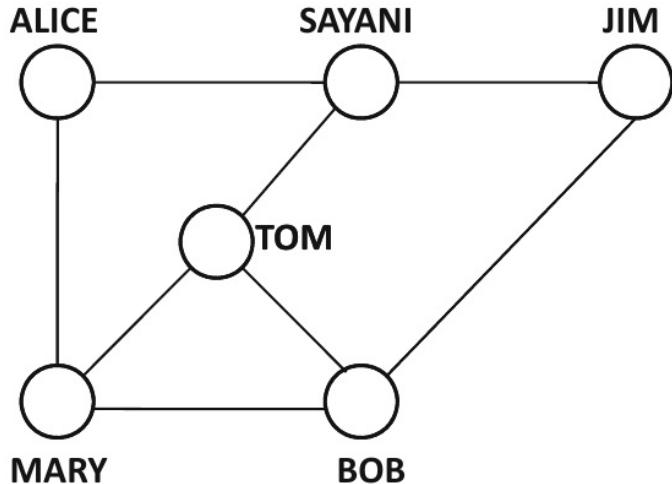
- $Jaccard(v_i, v_j) = \frac{|S_i \cap S_j|}{|S_i \cup S_j|}$

- $AdamicAdar(v_i, v_j) = \sum_{v_k \in S_i \cap S_j} \frac{1}{\log(|S_k|)}$

$$S_i = \{v_k \mid v_k \text{ neighbour of } v_i\}$$

Walk-based node similarity measures

Is Alice more similar to Bob or Jim?



(b) Many indirect connections between Alice and Bob

- Personalized PageRank with teleportation to v_i
- SimRank
- Katz measure

$$Katz(v_i, v_j) = \sum_{t=0}^{\infty} \beta^t \cdot n_{ij}^{(t)}$$

$n_{ij}^{(t)}$ = number of walks of length t between v_i and v_j
 $\beta < 1$ discount factor (punishes long walks)

Image sources

- Chang (2004): Unit 4: Circuit partitioning (lecture slides). EDA course, National Taiwan University.
[http://cc.ee.ntu.edu.tw/~ywchang/Courses/
EDA04/lec4.pdf](http://cc.ee.ntu.edu.tw/~ywchang/Courses/EDA04/lec4.pdf)
- Namtirtha et al. (2023): Placement Strategies for Water Quality Sensors Using Complex Network Theory for Continuous and Intermittent Water Distribution Systems. *Water Resources Research* 59(7), doi:10.1029/2022WR033112.
- Zardi et al. (2014): A Multi-agent homophily-based approach for community detection in social networks, *IEEE 26th Int. Conf. Tools with Artificial Intelligence*, doi: 10.1109/ICTAI.2014.81.

Data randomization for assessing the results of data mining

CS-E4650 Methods of Data Mining

Lecture November 7, 2023

Heikki Mannila

heikki.mannila@aalto.fi

Background

- Data analysis methods are very useful tools
- There are possible problems in using them
- Finding something which is only an artifact of the data or the analysis process, not representative of any real phenomena
- Also, issues related to bias, privacy, etc.

Simple ways of using randomization

- Add some noise to the data
- Does the result change dramatically?
- Remove a part of the data
- How much does the result change?
- If small changes in the data cause large changes in the results, it is usually good to understand why

What does a result mean? Example

- Even if the result seems robust, it does not mean that the result is informative or useful
- Example: Clustering: a clustering algorithm will return a clustering
- Even in the case when there is no cluster structure
- Trivial example: two-dimensional completely regular data
- What kind of cluster structure do we get?
- For example, by using k-means
- Does the result tell us something useful of the data?
- If possible, plot your data!

In this lecture

- We look at some simple ideas for randomizing data to help in assessing the results
- There is some nice theory behind these methods
- We do not discuss the theory much
- Additional reading: Assessing Data Mining Results via Swap Randomization, A. Gionis et al., ACM Transactions on Knowledge Discovery from Data, Vol. 1, No. 3, Article 14 (December 2007), [ACMJ346-06.tex](#)

Significance testing

- A statistic of interest $f(D)$ on the data
- A null hypothesis about the data D : a probability model that can be used to tell how likely or unlikely a value $f(D)$ is
- If probability of such a value is small, then it might be that the null hypothesis is not true
- Example: coin flipping

Significance testing of results using randomization

- Describe a randomized process for generating the data D
- Test statistic: a quantity of interest $f(D)$
- Generate many independent versions D_i of the data
- Test whether the value $f(D)$ of the test statistic on real data is extreme compared to the values $f(D_i)$
- Resulting in an estimate of how likely that is
- Comparison with traditional significance testing?
 - Randomized process replaces the probabilistic model
 - The randomized model can be arbitrarily complex
 - Good and bad

Significance testing has its problems

- Problems related to multiple testing
- If you do sufficiently many analyses, you'll find something even from random data
- Examples:
 - Spurious correlations
 - xkcd Significant <https://xkcd.com/882/>
- Several methods for handling this
 - Bonferroni correction; false discovery rate (FDR) approach

Significance testing has its problems

- One might do many different analyses without realizing it
- “Garden of forking paths” –
 - Selecting subsets of the data
 - Selecting properties of interest in the data
 - Soon one has actually done many analyses
- These problems remain in the randomization approach

Contents

- Basic idea of randomizing data
 - Example: 0-1 sequence analysis
- Randomizing data in prediction tasks
 - Example: ecological presence-absence data
- Swap randomization for 0-1 matrices
 - Example from in ecological data
- Bootstrapping (very briefly)
- Back to sequences

Basic approach in randomization

- You compute something from the data D : value $b = f(D)$
- For $i=1,\dots,n$
 - D_i = result of randomizing D
 - compute $b_i = f(D_i)$
- Compare the value b obtained from the real data to the set of values b_1, \dots, b_n
- If b is right there among the others, then the value $b=f(D)$ is no big news
- If b is quite extreme among the b_i 's, then perhaps the value of $b=f(D)$ is interesting

A simple example

- Monitoring a process
- At each timepoint we see either 0 (all OK) or 1 (something wrong)
- We are interested in finding out whether there are surprisingly long sequences of consecutive 1s in the sequence
- Example: a sequence S, which we know nothing about

Sequence analysis

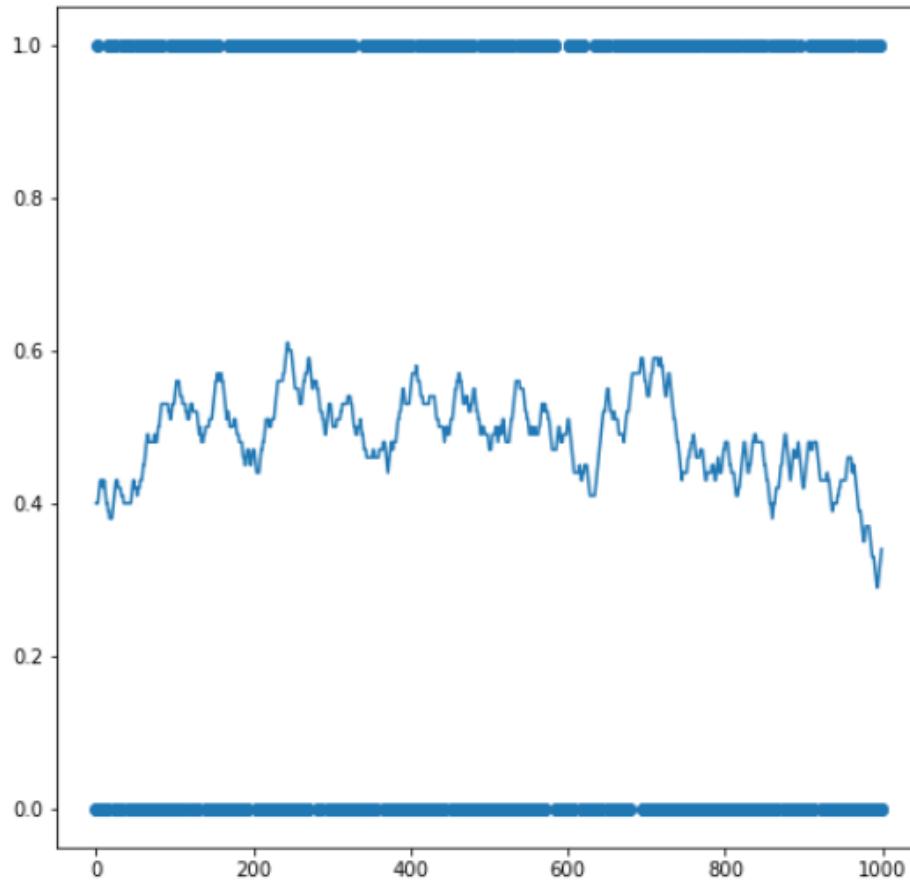
Descriptive statistics

```
In [5]: len(s), (s==0).sum(), (s==1).sum()
```

```
Out[5]: (1000, 513, 487)
```

So 1000 elements, about equally 0s and 1s

Sequence analysis – plot the data



Sequence analysis – longest run of consecutive ones

```
In [14]: z = longestrunc(S)
          print('The original sequence S had a run of ', z, 'consecutive 1s')
```

The original sequence S had a run of 17 consecutive 1s

```
In [15]: pS = np.random.permutation(S)
          pz = longestrunc(pS)
          print('A permuted version of S had a run of ', pz, 'consecutive 1s')
```

A permuted version of S had a run of 9 consecutive 1s

Is 17 consecutive one a lot or not?

Sequence analysis

Permute 1000 times and collect data

```
: iter = 1000
lenres = np.zeros(iter)
for i in range(iter):
    lenres[i]=longestrun(np.random.permutation(S))

L = longestrun(S)
print(L, lenres.mean(),lenres.std(), lenres.max())
# How many times was the result on randomized data smaller than on the true data
print((lenres>=L).sum())
```

```
17 8.929 1.8520148487525685 21.0
5
```

A simple example, cont.

- Quantity of interest F : length of the longest substring of 1s
- For the original data S we have $F(S)=17$
- We randomized the sequence S by permuting it
- The permuted sequence
 - Has the same number of 1s and 0s as the original sequence S
 - But the order has been lost
- The longest sequence of 1s in the 1000 permuted sequences was of length 21
- For 5 permutations the length of the longest sequence was at least 17

Basic approach in randomization (see slide 11)

- If $b=f(D)$ is right there among the others, then the value is no big news
- If b is quite extreme among the $b_i=f(D_i)$, then perhaps $b=f(D)$ is interesting
- Compute
$$z = |\{i \mid 1 \leq i \leq n \text{ & } f(D) > f(D_i)\}| / n$$
- Or
$$z = |\{i \mid 1 \leq i \leq n \text{ & } f(D) < f(D_i)\}| / n$$

In the example: $5/1000 = 0.005$

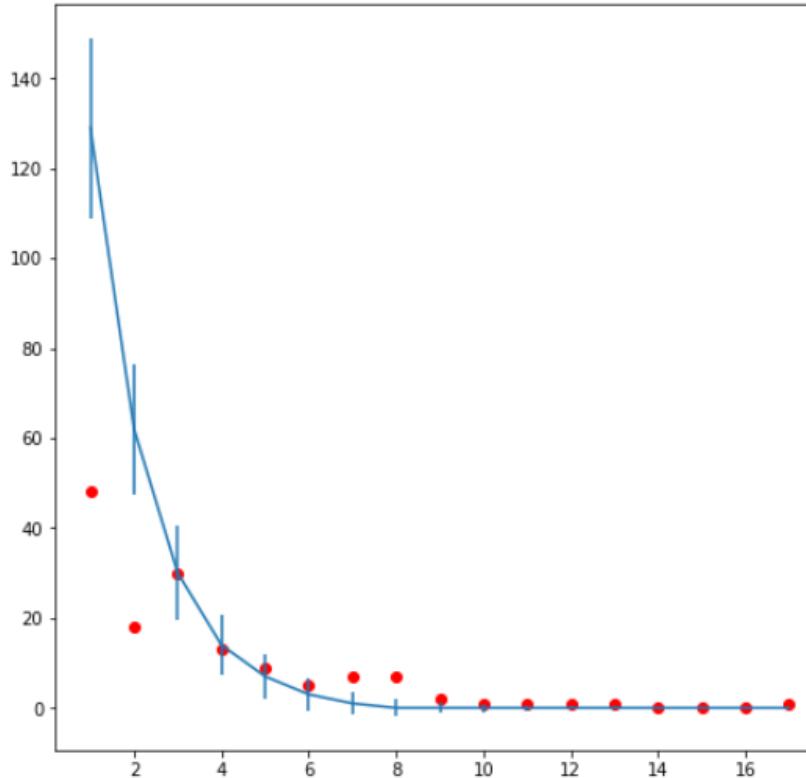
Wait a minute ...

- Couldn't we have estimated the probability of finding a substring with k 1s in a random permutation fairly easily?
- Yes, in this example that would have been possible
- For more complex examples, e.g.,
 - Distribution of all-1 substrings of size k for varying k
 - Longest subsequence with density at least a given bound
- It is harder to determine the probability analytically
- Quite easy using the randomization approach

Sequence analysis – distribution of the lengths of runs of 1s

Why does the distribution of the lengths of the 1s look like it does?

One should check the correctness!



Sequence analysis – longest subsequence with density at least 0.65? A slow implementation

```
In [28]: iter = 100
ssres = np.zeros(iter)
for i in range(iter):
    ssres[i], maxi, maxj = ldss(np.random.permutation(S),0.65)
```

```
In [29]: truess, maxi, maxj = ldss(S,0.65)
print(truess, ssres.mean(),ssres.std(), ssres.max())
# How many times was the result on randomized data smaller than on the true data
print((ssres>=truess).sum())
```

69 56.35 17.73943347460679 143.0 $24/100 = 0.24$
24

Garden of forking paths?

What does the example tell us?

- There are longer all-1 substrings in the data than in a random permuted sequence [with the same number of 1s]
- The distribution of all-1 substring of varying lengths differs from that in a randomly permuted sequence
- The longest subsequence with density at least 0.65 is about the same length as in a randomly permuted sequence
- Garden of forking paths?

Where did the sequence come from?

- A 2-state Markov chain
- In state 0, output 0 with probability 0.7 and stay in state 0
- In state 0, output 1 with probability 0.3 and go to state 1
- In state 1, output 1 with probability 0.7 and stay in state 1
- In state 1, output 0 with probability 0.3 and go to state 0

```
def genseq(n,a,b):  
    res = np.zeros(n)  
    state = 0  
    for i in range(n):  
        r = np.random.uniform()  
        if state==0 and r < a:  
            res[i]=0  
        elif state==0 and r > a:  
            res[i]=1  
            state=1  
        elif state==1 and r < b:  
            res[i]=1  
        elif state==1 and r > b:  
            res[i]=0  
            state=0  
    return res
```

```
N = 1000  
S = genseq(N,0.7,0.7)
```

Randomization in a prediction task

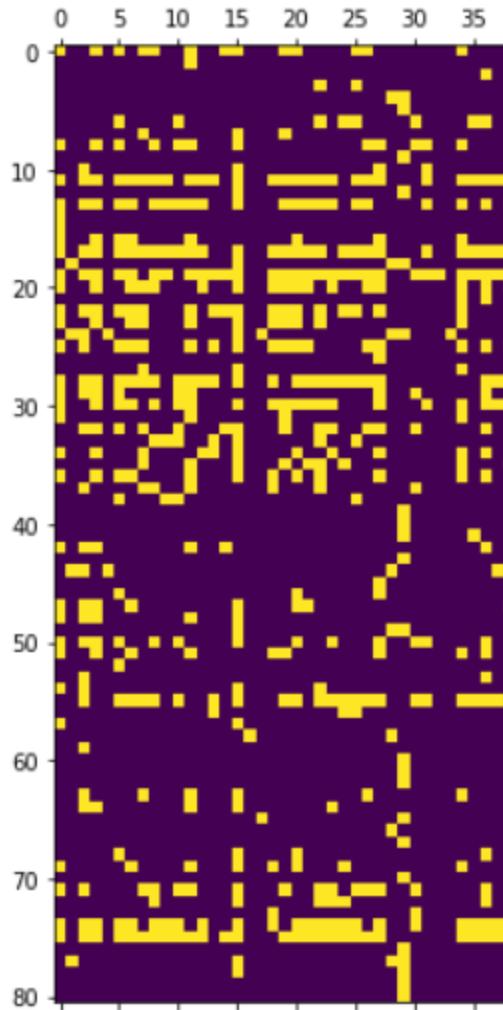
- In the previous example we computed some useful statistics from the data
- Consider now a prediction task: given data matrix X , predict target value y
- We want to find out whether the rows of X give us some information on the values of y
- How to use randomization here?
- Permute the values of y
- Cut all connection between X and y
- **Maintain the distribution of the values of y**

Prediction task - data

	Aoteapsyche sp1 FW_013_05	Aphrophila novaezelandiae	Archicauliooides diversus	Austraclima jollyae	Austrosimulum australense	Costachorema xanthoptera	Cricotopus I	Cricotopus II	Deleatidium sp1 FW_013_05	Eukiefiriella
Achnanthes lanceoloata	1	0	0	0	1	0	1	0	1	1	1
Achnanthes linearis	1	1	0	0	1	0	1	0	1	1	1
Achnanthes minutissima	0	0	0	0	0	0	0	0	0	0	0
Achnanthes saxonica	0	0	0	0	0	0	0	0	0	0	0
Achnanthes sp1 FW_013_05	0	0	0	0	0	0	0	0	0	0	0
...

- Reference: Thompson, R., Townsend, C. (2003) IMPACTS ON STREAM FOOD WEBS OF NATIVE AND EXOTIC FOREST: AN INTERCONTINENTAL COMPARISON. Ecology, 84(1), pp. 145–161
- Source: [https://esajournals.onlinelibrary.wiley.com/doi/abs/10.1890/0012-9658\(2003\)084%5B0145:IOSFWO%5D2.0.CO;2](https://esajournals.onlinelibrary.wiley.com/doi/abs/10.1890/0012-9658(2003)084%5B0145:IOSFWO%5D2.0.CO;2),
- Species: 98
- Predators (Columns) Preys (Rows)
- <https://www.web-of-life.es/map.php>

Prediction task – plot the data



Basic check

- Suppose we are interested in predicting the value of column 11
- The data seems to be fairly sparse (lots of 0s)
- How accurate is predicting “0” for every row?

```
In [50]: (1 - otagonpdata[:, colofinterest].sum() / otagonpdata.shape[0]).round(2)
```

- We get 68% accuracy without looking at the data in any detail

Prediction task: logistic regression

```
colofinterest = 11
y = otagonpdata[:,colofinterest] # column colofinterest
X = np.delete(otagonpdata.copy(),colofinterest,1) # delete column colofinterest from a copy
#print(y.shape, X.shape)
```

```
from sklearn.linear_model import LogisticRegression
clf = LogisticRegression(random_state=0).fit(X, y)
clf.score(X, y)
```

```
0.9259259259259259
```

```
# So we could predict column 11 by using the other columns with 93 % accuracy
```

```
perm_y = np.random.permutation(y)
perm_clf = LogisticRegression(random_state=0).fit(X, perm_y)
# print(y, perm_clf.predict(X), perm_y - perm_clf.predict(X))
perm_clf.score(X, perm_y)
```

```
0.8024691358024691
```

```
... and we could predict a random column with the same number of 1s with 85 % accuracy
```

```
1 - colsums[11]/81
```

```
0.6790123456790124
```

```
# And just by saying "each entry is 0" would give us 68 % accuracy
```

In the demo example

- The accuracy on the randomized data was almost as high as for the original data
- The reason: lots of predictors, few 1s in the column to be predicted
- Conclusion: we should not be too enthusiastic about the prediction results
- (What happens for the other columns in the data?)

An aside: a paper from 2017/2020

3446776
(acm.org)

DOI:10.1145/3446776

Understanding Deep Learning (Still) Requires Rethinking Generalization

By Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals

Abstract

Despite their massive size, successful deep artificial neural networks can exhibit a remarkably small gap between training and test performance. Conventional wisdom attributes small generalization error either to properties of the model family or to the regularization techniques used during training.

Through extensive systematic experiments, we show how these traditional approaches fail to explain why large neural networks generalize well in practice. Specifically, our experiments establish that state-of-the-art convolutional networks for image classification trained with stochastic gradient methods easily fit a random labeling of the training data. This phenomenon is qualitatively unaffected by explicit regularization and occurs even if we replace the true images by completely unstructured random noise. We corroborate these experimental findings with a theoretical construction showing that simple depth two neural networks already have perfect finite sample expressivity as soon as the number of parameters exceeds the number of data points as it usually does in practice.

We interpret our experimental findings by comparison with traditional models.

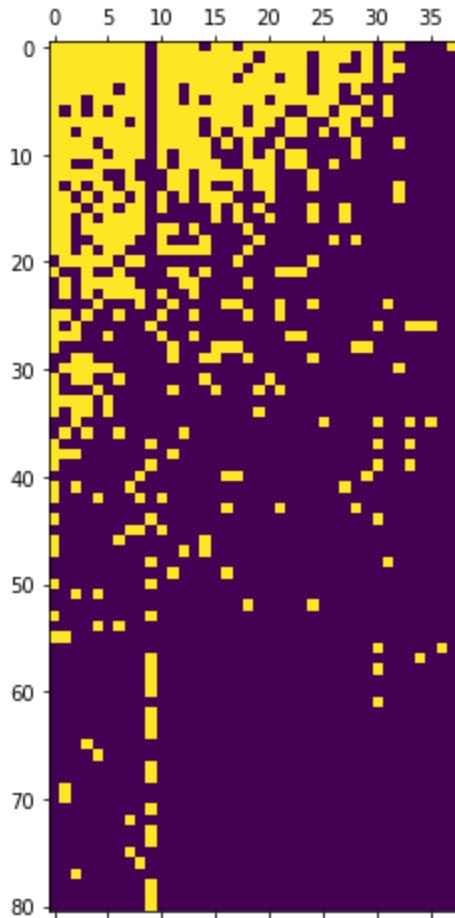
We supplement this republication with a new section at the end summarizing recent progresses in the field since the original version of this paper.

More complex randomizations

- In the previous examples we randomized *a sequence or a column* of the data
- What remained constant in the randomizations?
 - Total number of 1s
 - Counts of different values in the target column y
- Sometimes it is useful to randomize *the whole data matrix*
- For 0-1 data it is natural to maintain the row and column sums
- Number of 1s in each row and in each column
- Why would we be interested in this? Skewed distributions

Skewed distributions

- Previous data
- Rows and columns reordered by row sum and column sum
- Very skewed distributions of row and column sums



Example

X and Y are correlated
in the data.

In the second dataset, their correlation can be attributed to the highly skewed row sums in the data set.

Example (cont.)

- X and Y are correlated
- If we consider only columns for X and Y the correlation is clear
- If we look at the larger matrix, we notice that we can perhaps explain the correlation between X and Y by the number of 1s in the rows
- X and Y are correlated because they both tend to occur in the rows with many 1s
- This might be useful to notice if the data is, e.g., ecological data
- Two species X and Y are correlated because they occur in environments where there are many species

Swap randomization

- Randomization idea:
 - Given 0-1 matrix E
 - Produce randomized versions E_i of E
 - Having the same row and column sums as E
- How can we do this?

Swap randomization

- Given a matrix, change the data by applying the operation below:

$$\begin{array}{c} \begin{array}{cc} A & B \\ \vdots & \vdots \\ \cdots & 1 & \cdots & 0 & \cdots \\ \vdots & & \vdots & & \vdots \\ u & \cdots & 0 & \cdots & 1 & \cdots \\ v & \vdots & & \vdots & & \vdots \end{array} & \Leftrightarrow & \begin{array}{cc} A & B \\ \vdots & \vdots \\ \cdots & 0 & \cdots & 1 & \cdots \\ \vdots & & \vdots & & \vdots \\ u & \cdots & 1 & \cdots & 0 & \cdots \\ v & \vdots & & \vdots & & \vdots \end{array} \end{array}$$

Fig. 1. A swap in a 0–1 matrix.

- This maintains the number of 0s and 1s in rows and columns

Swap randomization

- Randomization: do enough swaps to get a random matrix with the same row and column sums as the original data
- As in the earlier examples, compute the quantity of interest both the original data and for the randomized data

Toy example: matrices D1 and D2

```
: iter = 500
origXYcorr = np.corrcoef(D1, rowvar=False)[0,1]

XYcorrs= np.zeros(iter)

for i in range(iter):
    D1s, succ = swaprandomization(D1,10000)
    XYcorrs[i] = np.corrcoef(D1s, rowvar=False)[0,1]

print(origXYcorr, XYcorrs.mean(), XY
0.55 -0.04175 0.25939484863813317
```

```
iter = 500
origXYcorr2 = np.corrcoef(D2, rowvar=False)[0,1]

XYcorrs2= np.zeros(iter)

for i in range(iter):
    D2s, succ = swaprandomization(D2,10000)
    XYcorrs2[i] = np.corrcoef(D2s, rowvar=False)[0,1]

print(origXYcorr2, XYcorrs2.mean(), XYcorrs2.std())
```

0.55 0.5198499999999999 0.14651698706975919

Thus

- [Computing correlations for such small (0-1) datasets is not good practice, but we use it to illustrate the point. Instead, one could compute, e.g., the dot product of columns X and Y.]
- X and Y are highly correlated.
- Randomize by looking at all matrices with the same row and column counts as D1
- The correlation is for data obtained from D1 by randomization is low
- For data obtained from D2 by randomization the correlation is hight
- For D2, the correlation of variables X and Y is explained by the row and column counts of D2

Swap randomization

- Theorem [Ryser 1957]: Assume D and E are $n \times m$ matrices with the same row and column sums. Then there is a sequence of swap operations that transforms D to E.
 - HJ Ryser: "Combinatorial properties of matrices of zeros and ones" - Canadian Journal of Mathematics, 1957.
- Not an easy proof, but not terribly difficult.
- Graph formulation:
 - Vertices: matrices with the same row and column sums
 - Edges: between D and E, if doing one swap to D gives E
- Ryser's theorem: this graph is connected

Swap randomization

- Theorem, informal: if you do sufficiently many random swap operations starting from D , then the resulting matrix is a random sample from the set of all matrices with the same row and column sums as D .
- Thus: we can produce random matrices with the same row and column sums as D
- And do randomization testing using those.

Huh? “Sufficiently many”?

- The number of random swaps needed to achieve “full mixing” is a deep question
- A random walk on the graph where the nodes are the matrices with the given row and column sums
- Experimentally it seems that about $5*L$ swaps is enough, if the matrix D has L 1s.

How to do swap randomization

- Easy to code the simplest possible algorithm
 - Generate randomly row numbers u and v
 - Generate randomly column numbers A and B
 - If the submatrix has the desired form, do the swap
 - Repeat until sufficiently many successful swaps
 - Slow
 - Really slow for large data

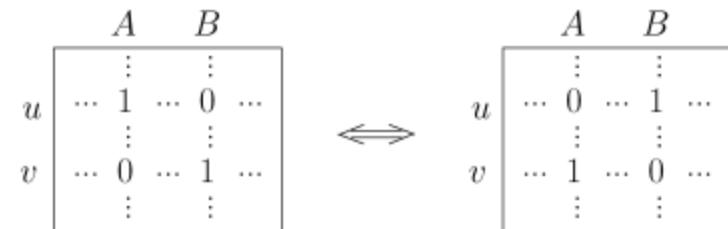


Fig. 1. A swap in a 0–1 matrix.

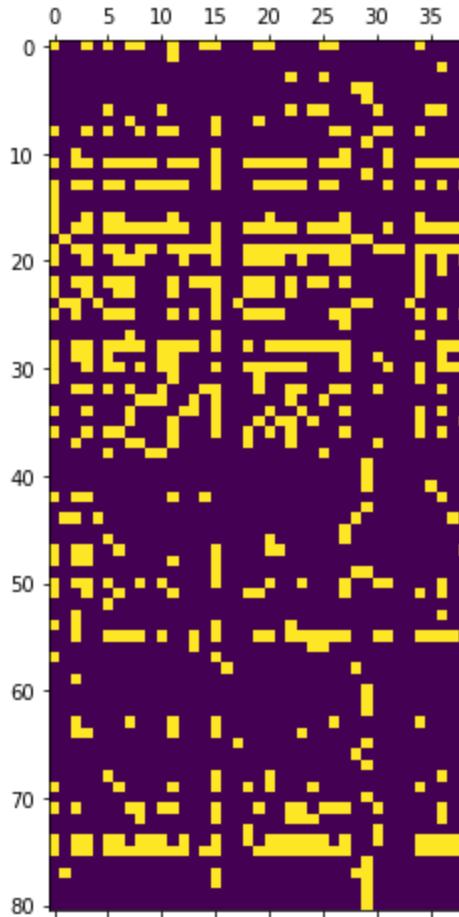
How to do swap randomization faster

- Many algorithms exist
- Not simple to guarantee that the method gives a random sample of the matrices with the given row and column sums
- See, e.g., A fast and unbiased procedure to randomize ecological binary matrices with fixed row and column totals | Nature Communications from 2014

Clustering

- Clustering methods have already been covered in the course
- Randomization for clustering

- Are there clusters in the data?
- The naïve randomization:
probability of 1 is the same in the randomized data as in the original data



Example: clustering

- Clustering methods have already been covered in the course
- Randomization for clustering
- Example

Trivial randomization for clustering

```
: ones = (otagonpdata==1).sum()  
print(ones)
```

577

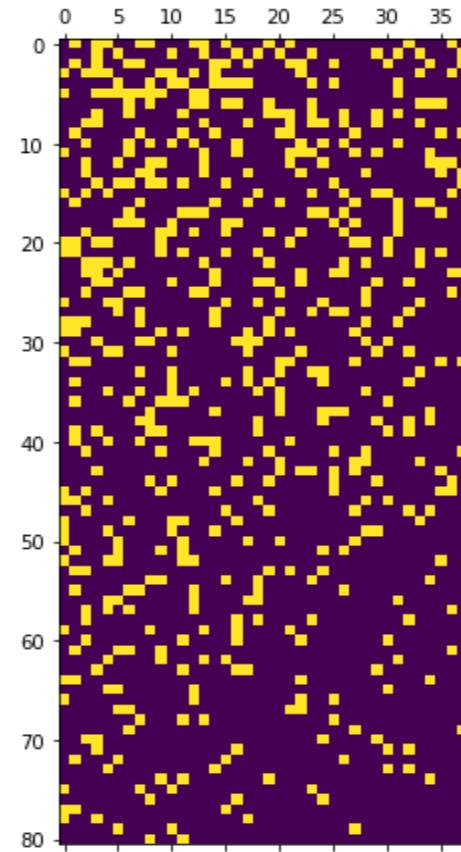
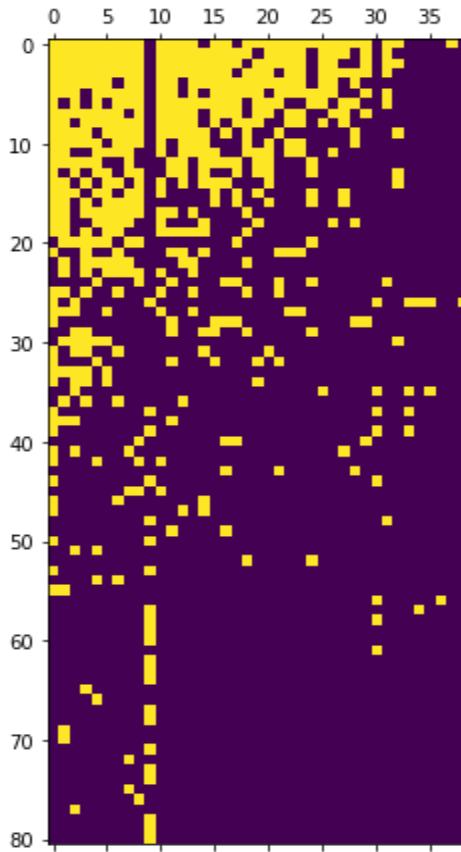
```
: # Frequency of 1s in the data  
(rc, cc) = otagonpdata.shape  
freq = ones/(rc*cc)  
print('Frequency of 1s in the data', ones, rc, cc, freq)
```

Frequency of 1s in the data 577 81 39 0.18265273820829375

```
: randm = (np.random.random_sample(size=(rc,cc))<freq).astype(int)  
  
print(randm.shape, (randm==0).sum(), (randm==1).sum(), (randm==1).sum()/(rc*cc))  
randm
```

(81, 39) 2611 548 0.17347261791706237

Trivial randomization for clustering: original and randomized reordered – row and column sums can change



Trivial randomization for clustering

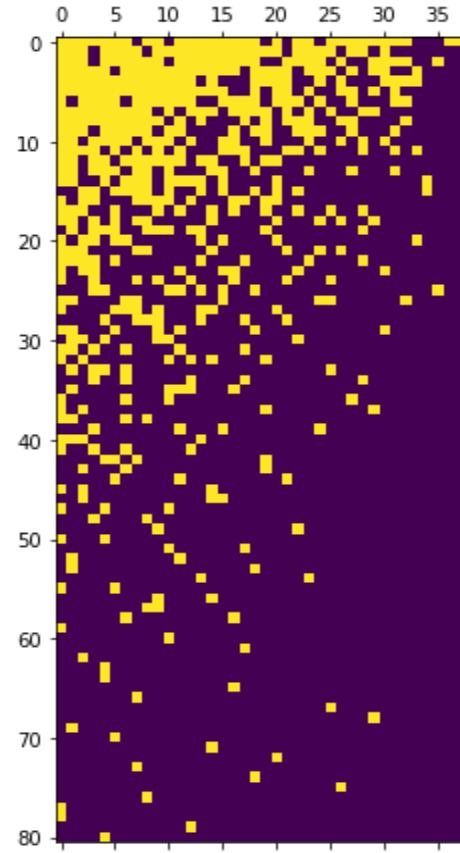
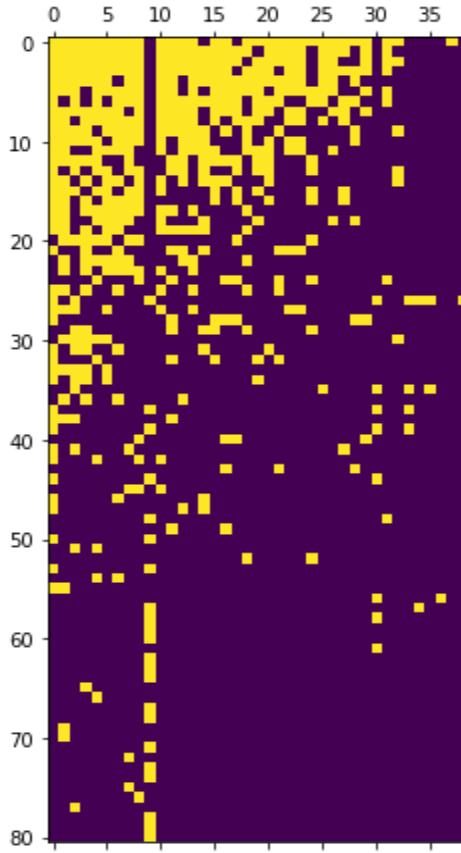
```
kmeans = KMeans(n_clusters=3).fit(otagonpdata)
scoredata = -kmeansr.score(otagonpdata)
scoredata
```

```
293.6834138724523
```

```
iter = 500
clusres = np.zeros(iter)
for i in range(iter):
    randm = (np.random.random_sample(size=(rc,cc))<freq).astype(int)
    kmeansr = KMeans(n_clusters=3).fit(randm)
    clusres[i]=-kmeansr.score(randm)
```

```
Original data 293.6834138724523 randomized mean 425.72046638298644 std 12.824839253082015
Number of cases with better score 0
```

Swap randomization for clustering: row and column sums stays the same



Swap randomization for clustering

```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=4).fit(otagonpdata)
scoredata = kmeans.score(otagonpdata)
-scoredata
```

233.39814814814812

```
iter = 100
sclusres = np.zeros(iter)
swaprdata, junk = swaprandomization(otagonpdata,500000)

for i in range(iter):
    print(i)
    swaprdata, junk = swaprandomization(swaprdata,500000)
    kmeansr = KMeans(n_clusters=4).fit(swaprdata)
    sclusres[i] = kmeansr.score(swaprdata)
```

```
scoredata, sclusres.mean(), sclusres.std(), (sclusres<scoredata).sum()
```

(-233.39814814814812, 276.1954270272572, 3.2930663706204175, 0)

Hence

- The clustering on the real data has a better score than those for swap randomized data
- The difference is a lot smaller than for the trivial randomization
- Thus the cluster structure might not be completely due to the skewed row and column distributions in the data

Larger experiments (Gionis et al 2007)

Table VII. Results on Clustering

Dataset	k	E	mean	std	Z	p
S1	10	1777.3	3669.9	11.1	170.43	0.01
	20	1660.7	3303.2	11.3	145.33	0.01
S2	10	4075.4	4084.4	11.6	0.77	0.22
	20	3686.2	3691.3	12.1	0.42	0.36
COURSES	10	17541.6	24405.1	30.2	227.09	0.01
	20	16062.0	23588.4	31.9	235.92	0.01
PALEO	10	1040.7	1401.7	4.8	74.74	0.01
	20	800.1	1193.9	5.9	67.09	0.01
RETAIL	10	23920.9	24086.0	135.2	1.22	0.10
	20	22276.3	22481.1	235.3	0.87	0.18

k : number of clusters used in k -means; E : clustering error in the original dataset; mean: mean clustering error in the sampled datasets; std: standard deviation of the clustering error in the sampled datasets; Z: distance of E from mean, measured in standard deviations; p : empirical p -value.

- Dataset sizes: S1, S2: 1000 x 20, Courses 2405 x 5021, Paleo 124 x 139, Retail 88162 x 16470

Bootstrapping

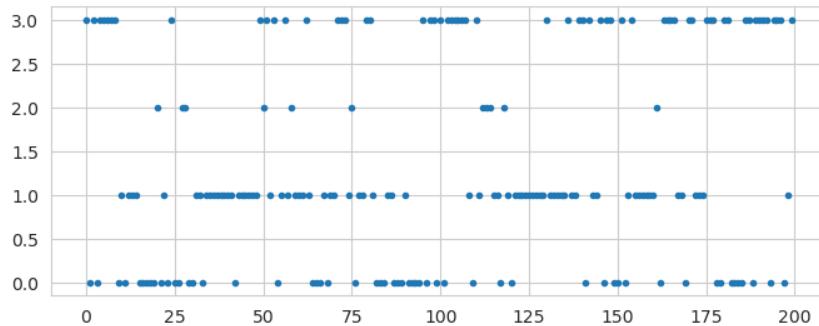
- Also called resampling; a slightly different idea
- A statistic of interest f , again
- Want to understand the distribution of $f(D)$
- Assume $D = \{x_1, x_2, \dots, x_n\}$
- Do many times:
 - Sample n entries from D with resampling, result D_j
 - Same entry can (and will) be many times in D_j
 - Compute $f(D_j)$
- Compute the average and std of $f(D_j)$'s

Back to sequences: another simple example

- Sentence embeddings
- A document with 740756 characters
- Look at fragments of size 500, no overlap; 1402 fragments
- Do sentence embedding (large language model style)
- For each fragment we get a 384-dimensional vector
- Is there some continuity in the vectors?
- Approach: cluster the vectors; see if we stay in the same cluster for longer time than for a randomized clustering

Approach A: cluster

- Cluster
- Count how many times we go from one cluster to another
- Sum of the diagonal: how many times stayed in the same

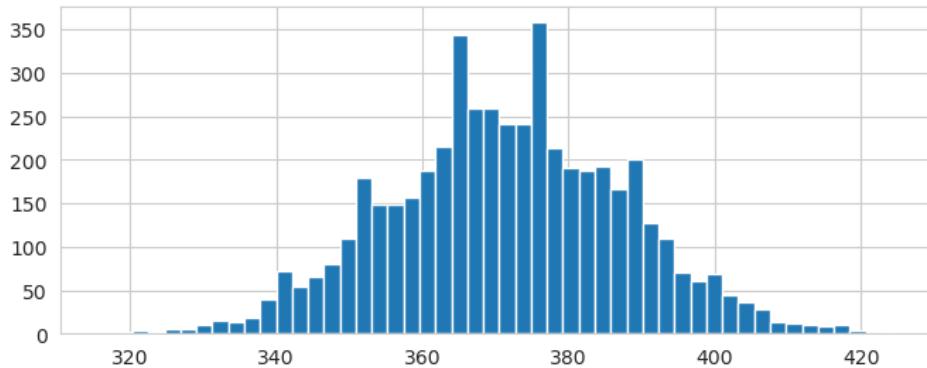
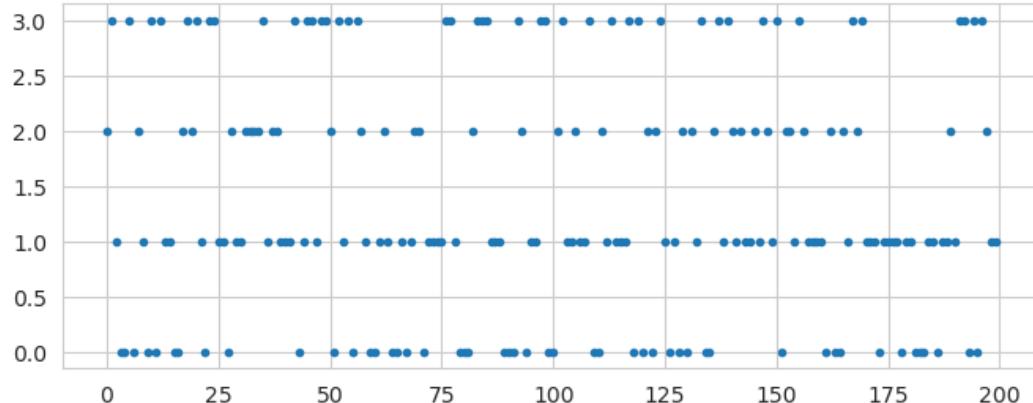


```
: def changes(labels):  
  
    k = np.max(labels)+1  
    res = np.zeros((k,k))  
    for i in range(len(labels)-1):  
        res[labels[i],labels[i+1]] += 1  
    return res  
  
:  
from sklearn.cluster import KMeans  
  
for n_clusters in range(4,5):  
    kmeans = KMeans(n_clusters=n_clusters, n_init='auto').fit( # , random_state=0).fit(  
        embeddings  
    )  
    print(changes(kmeans.labels_))  
    print(np.diag(changes(kmeans.labels_)).sum())  
  
[[ 163.   87.   49.   85.]  
 [ 84.  236.   67.   80.]  
 [ 44.   62.   75.   44.]  
 [ 94.   82.   34.  115.]]  
589.0
```

Approach A: cluster

- Randomize the cluster labels
- For each randomization count the sum of the diagonal
- For the true data the sum was 589

```
huu = np.random.permutation(kmeans.labels_)
plt.plot(huu[:200],'.')
[<matplotlib.lines.Line2D at 0x2ac360c0e710>]
```



Wrapping up

- Randomization: a tool for assessing data mining results
- Widely applicable, but not a solution to every problem
- Problem 1: how to decide what to randomize?
 - Determines our null model – what are we comparing against
- Problem 2: how do we know that the randomization algorithm really works correctly
 - Not trivial
- Randomization approach does not remove the problems of multiple testing or garden of forking paths
- xkcd: Correlation

Mining text data

1. Overview
 2. Preprocessing text data
 3. Representations (bag-of-words)
 4. Text clustering and its applications
 5. Extra: Word embeddings



1. Overview: Text data

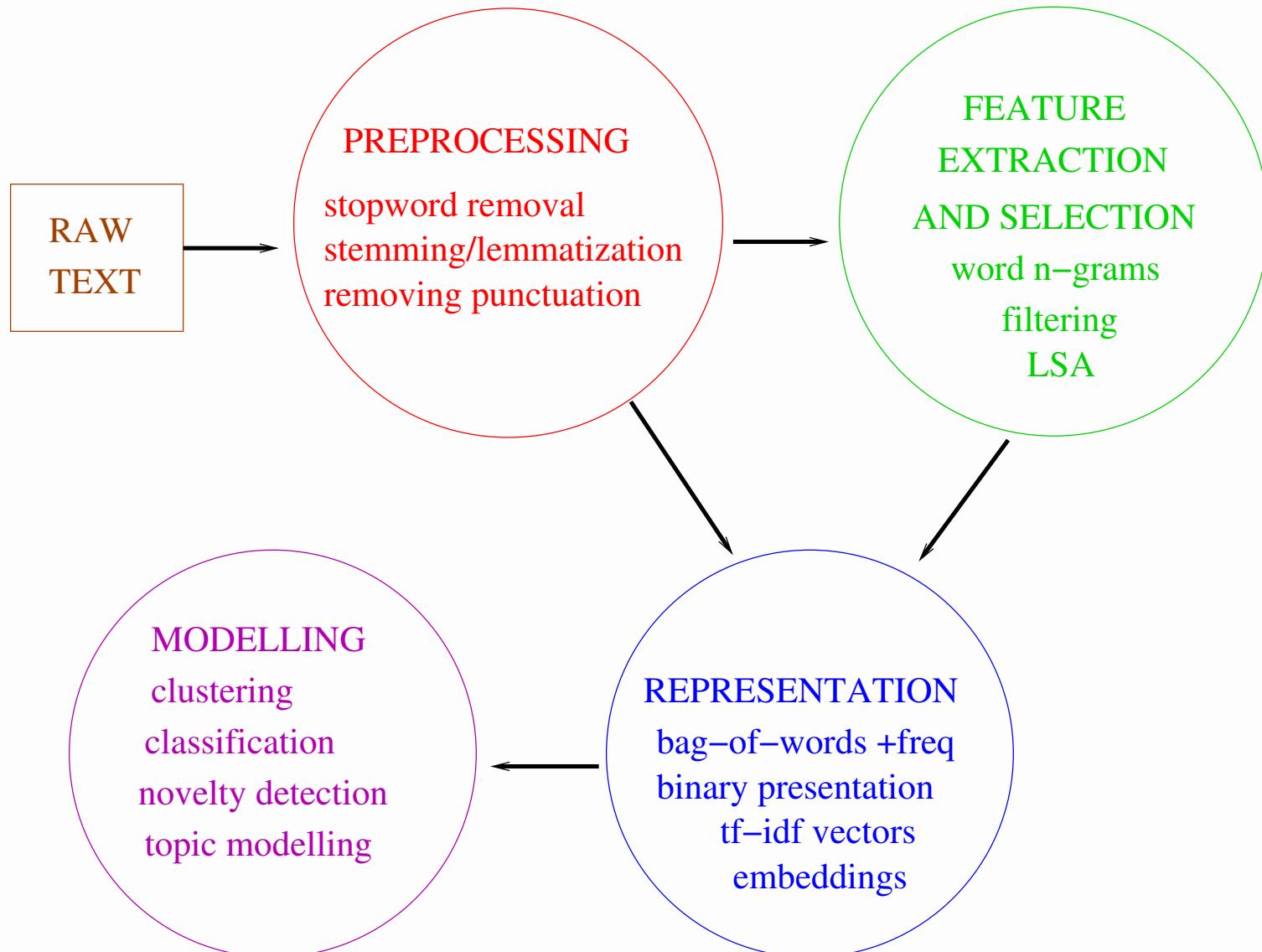
Corpus = collection of text documents ($\mathbf{d}_1, \dots, \mathbf{d}_n$)

Lexicon = set of words (w_1, \dots, w_m)

Document

- originally **ordered**: sequence of words (string), e.g.,
 $\mathbf{d}_1 = (w_{11}, w_{12}, \dots, w_{1q})$, length q
- **bag-of-words model**: unordered set of words (+ frequencies), e.g., *Our cat likes the neighbour's cat.* → $\{our: 1, cat: 2, likes: 1, the: 1, neighbour's: 1\}$
- **vector space presentation** as a numerical (or binary) vector $\mathbf{x} = (x_1, \dots, x_m)$
 - very sparse! (many 0s)
 - occurrence of words more important than absence

Main steps and example tasks



2. *Main tasks of preprocessing*

- **Tokenization** (usually word=token)
- **Lower-casing**
- **Remove stopwords** (may be stemmed or not, may contain apostrophes (*it's, you'd*))
- Reduce inflected forms into a base form:
 - **Stemming**: cut suffixes (*running* → *run*)
 - **Lemmatization**: derive dictionary form (*was* → *be*, *mice* → *mouse*)
- **Remove punctuation**. Decide how to handle
 - digits (numbers sometimes informative)
 - dashes in compound words (*cat-food* → *catfood* or *cat food*?)

Stopwords

- frequently occurring words with little information about semantic content
 - not discriminative
 - articles, prepositions, pronouns, auxiliary verbs, ...
 - multiple lists available
 - Note: relevance of words depends on the context!
→ check & edit!

•
•
•
cannot
cant
co
computer
con
could
couldnt
cry
de
describe
detail
•

Warning:
may be
relevant

Stemming and lemmatization

Stemming cuts suffixes → root form

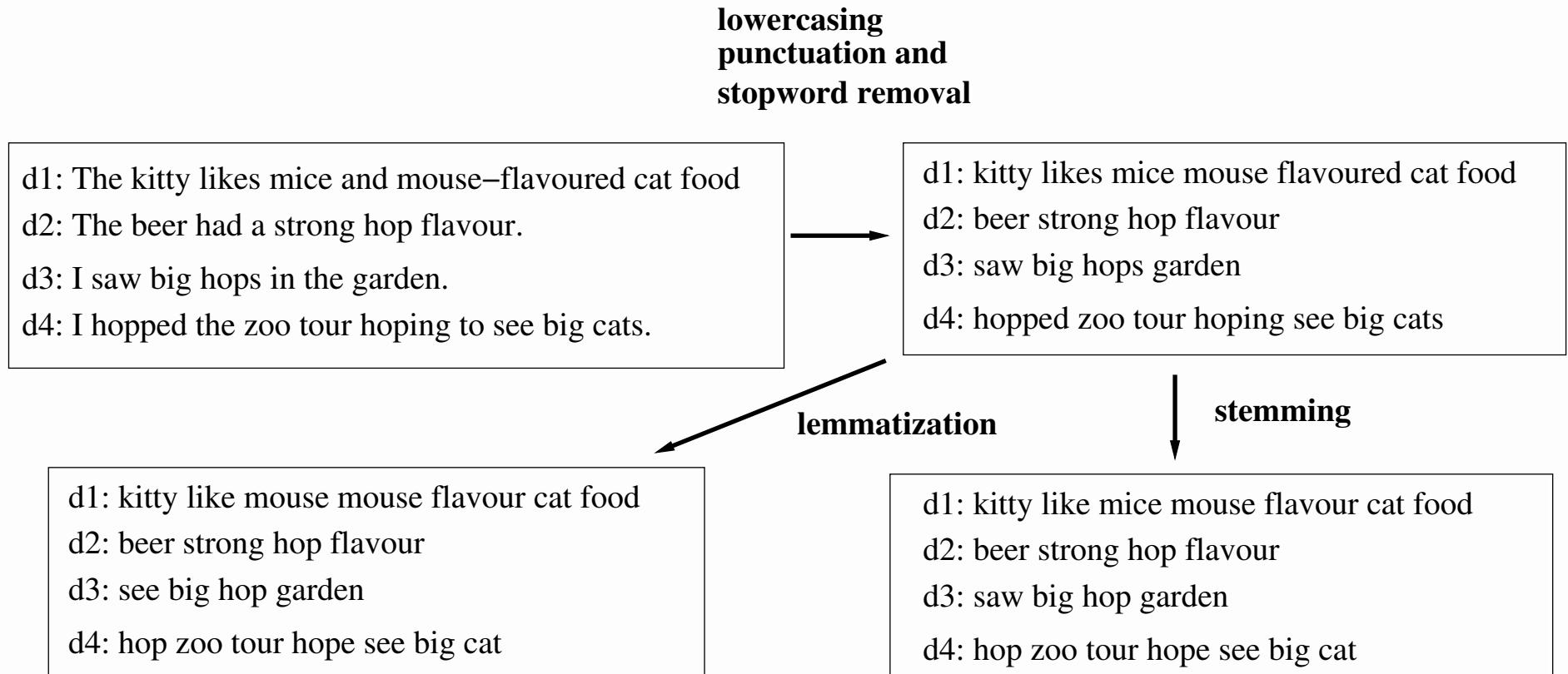
- Overstemming: word is over-truncated → false synonyms (e.g., *hopping* and *hope* → *hop*)
- Understemming: too little truncation → same root not detected (e.g., *alumnus* → *alumnu* vs. *alumni* → *alumni*)
- rule-based, possibly use look-up tables

Lemmatization derive dictionary form

- utilize context, part-of-speech tagging ^a, look-up tables
- more difficult! → slower
- potentially better accuracy

^anoun, verb, adjective, ...; singular/plural; verb tense; subject, object, ...

Example: simple preprocessing



Example: document-term matrix

d1: kitty like mouse mouse flavour cat food

d2: beer strong hop flavour

d3: see big hop garden

d4: hop zoo tour hope see big cat

Typically very sparse matrix!
Present compactly!

	kitty	like	mouse	flavour	cat	food	beer	strong	hop	see
d1	1	1	2	1	1	1				
d2				1			1	1	1	
d3									1	1
d4					1				1	1

	big	garden	zoo	tour	hope
d1					
d2					
d3	1	1			
d4	1		1	1	1

Note: 0s often skipped in visual presentation (not stored in real structure)

3. Representation (VSM=vector space model)

1. **Frequency model:** $x_{ij} = fr(w_j|\mathbf{d}_i)$ + possibly normalize to $\|\mathbf{x}\| = 1$ ($fr(w_j|\mathbf{d}_i)$ =number of times w_j occurs in \mathbf{d}_i)
2. **Boolean (binary) model:** $x_{ij} = 1$, if w_j occurs in \mathbf{d}_i and 0 otherwise
3. **Tf-idf representation:** $tf \cdot idf$, where tf =term frequency, idf =inverse document frequency
 - basic form $tfidf(w_j, \mathbf{d}_i) = fr(w_j|\mathbf{d}_i) \cdot \log \frac{n}{n_j}$
 - $df(w_j) = \frac{n_j}{n}$, $idf(w_j) = \frac{n}{n_j}$, n_j =number of documents containing w_j
 - When $tfidf$ is maximal?
 - many variants! → Check what your library calculates!

Some *tf-idf* variants

- tf can be normalized: $\frac{fr(w_j|\mathbf{d}_i)}{\text{len}(\mathbf{d}_i)}$
- frequency damping: $tf(w_j, \mathbf{d}_i) = \log(fr(w_j|\mathbf{d}_i))$ or $\sqrt{fr(w_j|\mathbf{d}_i)}$
- Note: **sklearn** offers **length-normalized** ^a versions of either $fr(w_j|\mathbf{d}_i) \cdot \left(\log\left(\frac{1+n}{1+n_j}\right) + 1 \right)$ or $fr(w_j|\mathbf{d}_i) \cdot \left(\log\left(\frac{n}{n_j}\right) + 1 \right)$
 - always $idf > 0$
 - $\frac{1+n_j}{1+n}$ smoothed estimate of relative document frequency

^aeither $\sum_j x_{ij}^2 = 1$ or $\sum_j |x_{ij}| = 1$ or none

Example: Basic tf-idf

$$tf = fr(w_j | \mathbf{d}_i) = 1 \text{ or } 2 \text{ (mouse)}$$

$$n_j = df(w_j) \in \{1, 2, 3\} \Rightarrow idf(w_j) \in \{2, 1, 0.415\}.$$

	kitty	like	mouse	flavour	cat	food	beer	strong	hop	see
d1	2.0	2.0	4.0	1.0	1.0	2.0				
d2				1.0			2.0	2.0	0.415	
d3									0.415	1.0
d4					1.0				0.415	1.0

	big	garden	zoo	tour	hope
d1					
d2					
d3	1.0	2.0			
d4	1.0		2.0	2.0	2.0

Are single words good features?

Basic features: (stemmed) words (excluding stopwords)

1. too frequent words don't separate documents
 - stopword removal and *idf* help
 - common word may be part of useful feature, e.g., *data mining, data management, text data* → word n-grams
2. unique or rare words don't reveal similarity between documents
 - words may still be synonyms (*kitty, cat*) → WordNet
 - or related → LSA and word embeddings can help
 - different spelling (*neighbour, neighbor*) or errors
3. polysemy problem: similar looking word (or stem) may have different meanings (*hop*)

Create features for word n-grams?

word n-gram = sequence of n words that often occur together (phrases, collocations)

- monogram (unigram, 1-gram): single word “*data*”
- bigram: two words: “*data analysis*”
- trigram: three words: “*Bayesian data analysis*”

results a lot of features! \Rightarrow filtering by

- frequency (keep if $fr(w_1 w_2)$ high)
- significance of association (e.g., $MI(w_1, w_2)$, $\chi^2(w_1, w_2)$)
- dimension reduction (e.g., LSA)

Note: character n-grams = n consecutive characters

Latent semantic analysis (LSA) = SVD applied to text data

- radical dimension reduction (e.g., 100 000 → 300)
- helps with synonymy and a bit with polysemy
- similar/related terms tend to be combined, e.g.,
 $\{\text{car, truck, flower}\} \rightarrow \{1.3452 \cdot \text{car} + 0.2828 \cdot \text{truck, flower}\}$

$$A_{n \times d} = \hat{U}_{n \times r} \Sigma_{n \times d} V^T_{d \times d} \xrightarrow{\hat{V}} A\hat{V}$$

The diagram illustrates the Singular Value Decomposition (SVD) of a matrix A . The matrix A is shown as a pink rectangle labeled A and $n \times d$. It is decomposed into three components: \hat{U} , Σ , and V^T . \hat{U} is a pink rectangle labeled \hat{U} and $n \times r$, where r is the rank of A . Σ is a light blue rectangle divided into two horizontal sections: a pink top section labeled $\hat{\Sigma}$ and $r \times r$, and a light blue bottom section. V^T is a light blue rectangle divided into two horizontal sections: a pink top section labeled \hat{V}^T and $r \times d$, and a light blue bottom section. Below the components are their respective dimensions: \hat{U} is $n \times n$, Σ is $n \times d$, and V^T is $d \times d$.

Recall: If old features F_1, \dots, F_d and i th singular vector $(\mathbf{V}_{i1}, \dots, \mathbf{V}_{id})^T$, i th new feature $\sum_{j=1}^d \mathbf{V}_{ij} F_j$. LSA: Recap Aggarwal 2.4.3.3. Image: Perunicic (2017).

4. Clustering text data

Given **vector-space representation** of text data:
any common clustering method

- K -representatives, spectral, hierarchical, probabilistic EM-algorithm, affinity propagation
- distance/similarity: prefer **cosine similarity**!
remember: if data normalized to $\|\mathbf{x}\| = 1$,
 $L_2^2(\mathbf{x}_1, \mathbf{x}_2) = 2(1 - \cos(\mathbf{x}_1, \mathbf{x}_2)) \rightarrow L_2$ ok
- dimension reduction (PCA or LSA) can help

Suggestion: Use K -means always as a baseline to compare other methods!

Example: cos-sim vs. L_2 distance

d1: kitty like mouse mouse flavour cat food

d2: beer strong hop flavour

d3: see big hop garden

d4: hop zoo tour hope see big cat

\cos_T =cos-sim of tfidf vectors, \cos_B =cos-sim of binary vectors,
 L_2 =Euclidean distance between tfidf vectors:

pair	\cos_T	\cos_B	L_2	note
(d1, d2)	0.0603	0.204	6.097	
(d1, d3)	0	0	6.014	No common words!
(d1, d4)	0.0469	0.154	6.571	
(d2, d3)	0.0229	0.250	3.873	
(d2, d4)	0.0146	0.189	4.899	
(d3, d4)	0.2245	0.567	4.123	

Special techniques for clustering text

1. Modifications of K -representatives

- replace cluster centroids by **cluster digests** = most frequent words (e.g., 200–400)
- better strategies to select seeds!

2. Co-clustering

- Idea: cluster words and documents simultaneously
- cluster $C_i = (R_i, V_i)$; V_i =most relevant words (cluster digest) in documents R_i

Note: “cluster digest” has slightly different meanings in different contexts (intention to describe the cluster)

Scatter/Gather: better K -means for text

1. Construct K seeds with agglomerative **hierarchical clustering**. Two alternatives:
 - a) **Buckshot**: choose \sqrt{Kn} random samples $\rightarrow K$ clusters $\rightarrow K$ seeds
 - b) **Fractionation**: divide data into $\frac{n}{m}$ buckets $\rightarrow \nu m$ clusters per bucket ($\nu \in]0, 1[$) \rightarrow concatenate cluster documents \rightarrow repeat until K clusters left
2. Apply **K -means**
 - centroid=concatenation of documents + remove infrequent words

Scatter/Gather: better K-means for text

3. (Optionally) Refine clusters:

a) Split incoherent clusters

- coherence score: $\text{avg}(\text{sim}(\mathbf{x}, \mathbf{c}))$ or $\text{avg}(\text{sim}(\mathbf{x}_i, \mathbf{x}_j) \mid \mathbf{x}_i, \mathbf{x}_j \in \mathbf{C})$
- if too low → Buckshot with $K = 2 +$ recluster rest

b) Join similar clusters

- topical words overlapping significantly

Cutting et al. (1992). Scatter/Gather: A cluster-based approach to browsing large document collections.

Co-clustering (biclustering)

Idea: rearrange rows and columns of doc-term matrix such that most non-zero entries form blocks.

	CHAMPION	ELECTRON	TROPHY	RELATIVITY	QUANTUM	TOURNAMENT
D ₁	2	0	1	1	0	3
D ₂	0	2	0	1	3	0
D ₃	1	3	0	1	2	0
D ₄	2	0	2	0	0	3
D ₅	0	2	1	1	3	0
D ₆	1	0	2	0	0	3

(a) Document-term matrix

	CHAMPION	TROPHY	TOURNAMENT	ELECTRON	RELATIVITY	QUANTUM
D ₁	2	1	3	0	1	0
D ₂	2	2	3	0	0	0
D ₃	1	2	3	0	0	0
D ₄	0	0	0	2	1	3
D ₅	1	0	0	3	1	2
D ₆	0	1	0	2	1	3

(b) Re-arranged document-term matrix

=Bipartite graph partitioning problem!

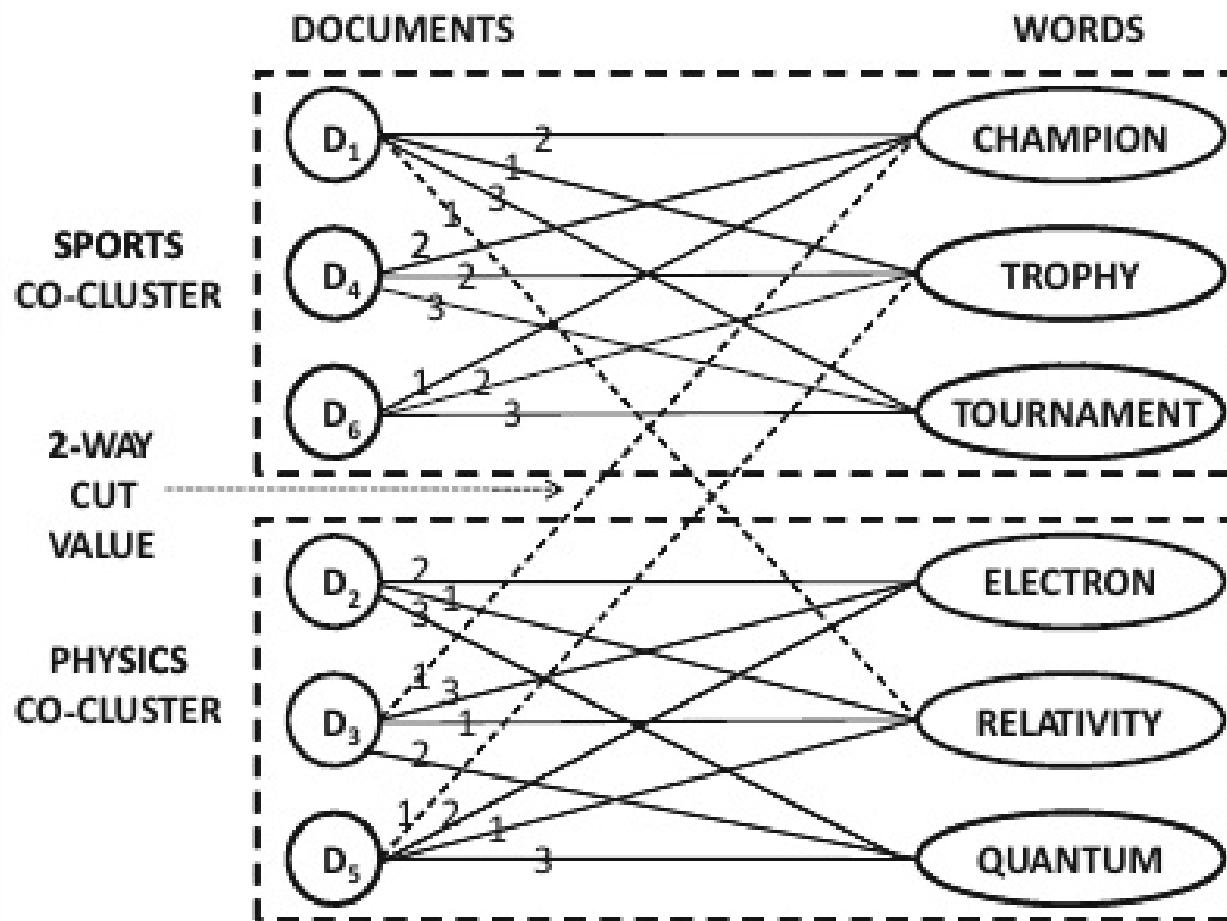


Image source Aggarwal Fig. 13.2

Graph partitioning

Construct graph $\mathbf{G} = (\mathbf{U} \cup \mathbf{V}, \mathbf{E}, \Gamma)$

- \mathbf{U} = nodes for documents (for doc. \mathbf{d}_i node u_i)
- \mathbf{V} = nodes for words (for word w_j node v_j)
- \mathbf{E} = edges
 $(u_i, v_j) \in \mathbf{E}$, if word w_j occurs in \mathbf{d}_i
- Γ = weights
 $\gamma_{ij} = fr(w_j|\mathbf{d}_i)$ or $\gamma_{ij} = tfidf(w_j, \mathbf{d}_i)$

Objective: Partition $\mathbf{U} \cup \mathbf{V}$ into groups such that edge-cut cost ($\sum \gamma_{ij}$ between groups) minimal (+ extra constraints)

Solution: Spectral clustering or other graph partitioning methods (vs. community detection)

Application 1: clustering for classification

Centroid-based classifier:

- Cluster documents of each class (size n_i) into $k_i \propto n_i$ clusters
 - **cluster digest**=most common words of the centroid
 - For \mathbf{d}_{new} :
 - determine K nearest digests (clusters)
 - report the dominant label
-
- + fast alternative to K -NN classifier
 - + handles **synonymy** (similar words → same centroid) and **polysemy** (different meanings → different centroids)

Application 2: Novelty detection

Problem: Temporal stream of text documents, when a new topic appears? (e.g., news)

Simple solution:

- Maintain a sample of documents, \mathcal{D}
- For \mathbf{d}_{new} calculate $sim(\mathbf{d}_{new}, \mathbf{d})$ for all $\mathbf{d} \in \mathcal{D}$
- If novelty score = $\frac{1}{\max_d sim(\mathbf{d}_{new}, \mathbf{d})}$ high, report \mathbf{d}_{new}

Problem: Pairwise similarity cannot handle synonymy or polysemy \Rightarrow Utilize **micro-clustering**

Novelty detection with micro-clustering

Idea: Maintain K document clusters C_1, \dots, C_K .

For C_i : \mathbf{c}_i = centroid = **cluster digest**, $fr(w_j|C_i)$ word frequencies,
 t_i = time stamp when C_i updated

- given \mathbf{d}_{new} , determine nearest centroid \mathbf{c}_i
- if $(sim(\mathbf{d}_{new}, \mathbf{c}_i) \geq \theta)$
 - add \mathbf{d}_{new} to C_i
 - update $fr(w_j|C_i)$, \mathbf{c}_i (most frequent words) and t_i
- else
 - report \mathbf{d}_{new} as novelty
 - create a new cluster $C = \{\mathbf{d}_{new}\}$ (with new time stamp)
 - remove an old cluster C_i with earliest t_i

5. Extra: Word embeddings

Idea: Present words as numerical vectors such that distances reflect semantic similarity

2D presentation of embeddings:

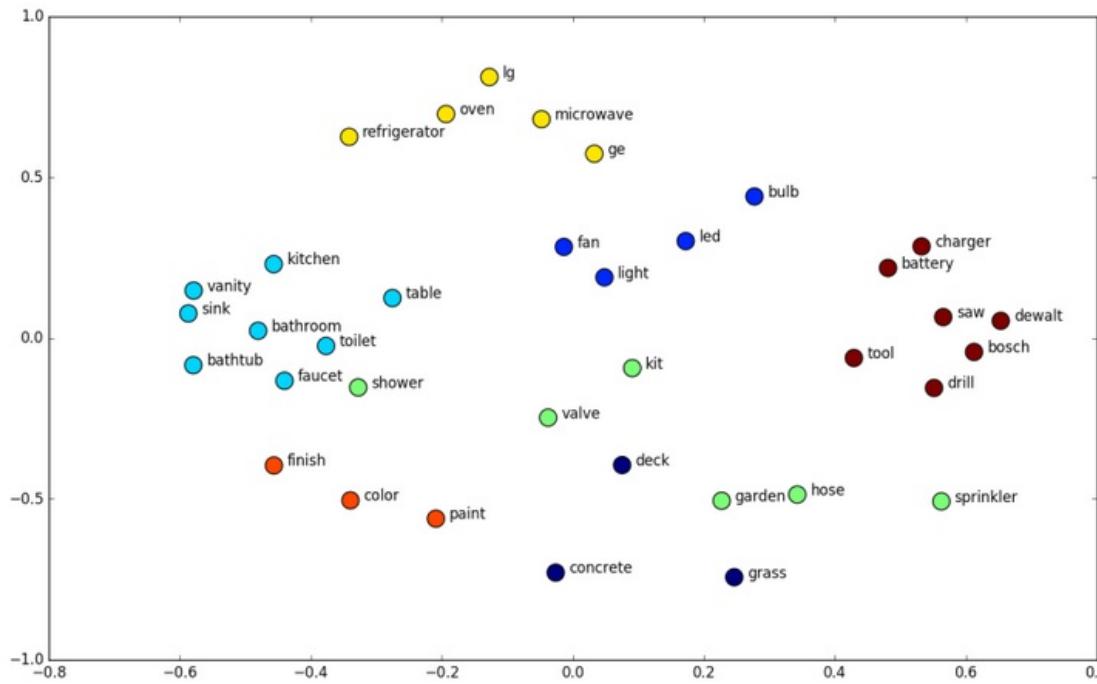


image source Barla (2021)

Word embedding examples

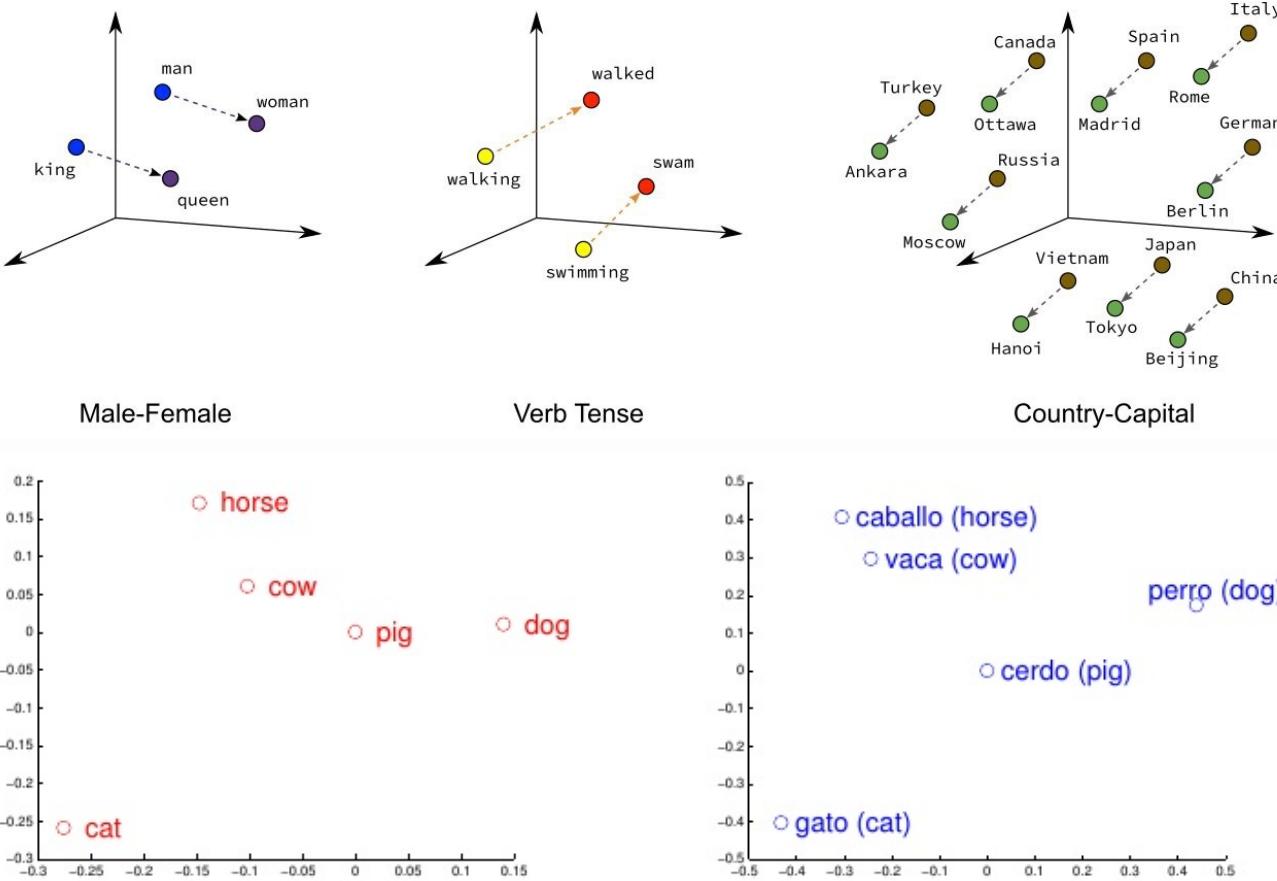


image sources Zhang (2020) and Mikolov et al. (2013)

Word embeddings: popular approaches

1. **Matrix factorization:** typically truncated SVD of word co-occurrence (or term-doc) matrix
2. **Word2vec models:** learn a shallow neural network and use its weights
 - a) **CBOW:** predict target word given context words
 - b) **Skip-gram:** predict context words given target word
 - good embeddings require large training sets
but models can be transferred to other similar contexts

Mikolov et al. (2013)

Word embeddings: popular approaches

3. **GloVe**: learn embedding vectors \mathbf{x}_i , $\tilde{\mathbf{x}}_i$, such that

$$\mathbf{x}_i \cdot \tilde{\mathbf{x}}_j \propto \log(P(w_j|w_i))$$

- $\tilde{\mathbf{x}}_i$ context embedding → later combine with \mathbf{x}_i
- + fast to learn
- + requires less data

Pennington et al. (2014)

Note: *Hype* and *best* are not synonyms. Test always simpler presentations (bag-of-words), too!

Summary

- text presented in vector space (as numerical vectors)
 - simplest: document=bag-of-words with binary occurrence, frequencies, or *tfidf* of words
- preprocessing important → features. Be careful!
- Goal: try to capture important (mid-frequency) words and phrases
 - prune out stopwords, stemming/lemmatization, detect collocations/n-grams, maybe spell-checking
- dimension reduction often useful (+ LSA helps with synonymy/polysemy)
- clustering: K -representatives + modifications, spectral, hierarchical, co-clustering, ...

References & image sources

- Barla (2021): The ultimate guide to word embeddings
<https://neptune.ai/blog/word-embeddings-guide>
- Cutting et al. (1992): Scatter/Gather: A cluster-based approach to browsing large document collections. ACM SIGIR Conference, pp. 318-329.
- Mikolov et al. (2013): Efficient estimation of word representations in vector space. ICLR and arXiv:1301.3781
- Mikolov et al. (2013): Exploiting similarities among languages for machine translation. arXiv:1309.4168

References & image sources

- Pennington et al. (2014): GloVe: Global vectors for word representation. Empirical Methods in Natural Language Processing (EMNLP).
- Perunicic (2017): How are principal component analysis and singular value decomposition related?
<https://intoli.com/blog/pca-and-svd/>
- Zhang (2020): Legal applications of neural word embeddings <https://towardsdatascience.com/legal-applications-of-neural-word-embeddings-556b7515012f>

Common sources of erroneous, inaccurate or misleading results in data mining

Wilhelmiina Hämäläinen

November 28, 2023

Figure 1 presents the knowledge discovery process with an additional step 0 (gathering data). Step 0 is usually not included to the process, since the data miner is typically invited when the data has been gathered. However, many problems could be avoided if the data gathering had been done carefully.

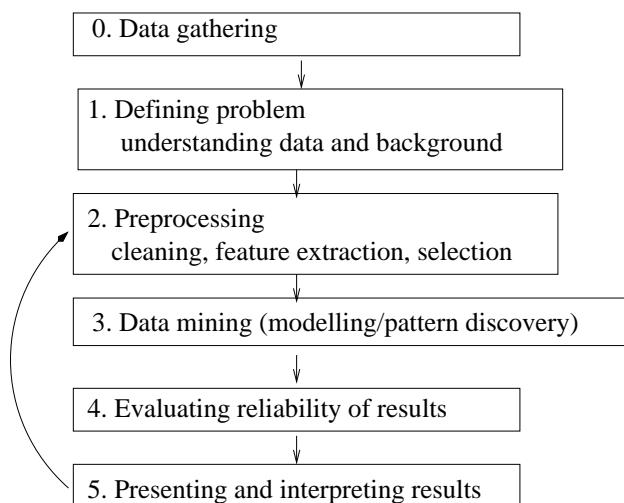


Figure 1: Steps of the KDD process extended with step 0, gathering data.

Here are important problem sources at each phase:

0. Gathering data: This step is the ultimately the main source of errors and inaccuracies in data, stemming from, e.g., faulty or inaccurate devices, human respondents giving incorrect/missing information for private reasons, and human errors in manual recording of data. The data

may also be too sparse or not representative to the true population, which leads to many problems – the model parameters cannot be estimated accurately, the discoveries will lack statistical validity (may not hold in future data) and the results may be misleading (describe some exceptional subpopulation instead of the assumed target population).

1. Defining the problem: The computational problem (and related assumptions) may be misspecified, because the data miner and client do not understand each other (people in different fields have their own terminology and may even use the same term for different meanings). In the worst case, the data miner solves a wrong problem, e.g., searches frequent co-occurrence associations when the client wanted statistical associations. Another problem is misspecification of **data types**, which can lead to serious errors at later steps (remember: everything that looks numbers is not ratio-scale numerical data – even categorical values can be coded by numbers and there are numerical datatypes, like circular variables, that require special treatment).
2. Preprocessing: Real world data contains nearly always some anomalies and data cleaning is seldom perfect, leaving possibly erroneous values or incorrectly imputed missing values. Outliers can bias results seriously, depending on the modelling technique (e.g., Pearson correlation coefficients are inaccurate, K -means cannot detect real clusters), unless detected and dealt appropriately.

Feature extraction and selection have a crucial effect on results, leading to e.g., missed and trivial patterns or suboptimal clusterings. One common error is to ignore the data types and apply PCA or SVD dimension reduction on non-numerical or circular features. Similar problems occur if the curse of dimensionality is ignored, because it tends to produce misleading similarity and distance evaluations and emphasize irrelevant (possibly erroneous) features.

3. Data mining: At this step, the main source of errors is ignorance of methods and their assumptions. One may choose a method that doesn't solve the specified problem (e.g., search only condensed presentations of frequent associations instead of significant statistical associations) or whose assumptions don't fit the data (e.g., certain shape of clusters, absence of outliers). Many errors stem from programs and they should be tested carefully and one should also prepare for exceptions (like data formatting errors). One common source of errors is to use ready made library functions without checking what they actually calculate or how

the parameters should be set. One should also remember that existing programs do contain bugs or may not work correctly in pathological cases. Here one could also mention accuracy problems related to presentation of floating point numbers – small inaccuracies can accumulate and produce seriously erroneous results.

4. Evaluating reliability of results: Validation of results is extremely important, because the methods tend to return something even from random data. One would need some guarantees that the patterns are not due to chance, but likely to hold also in future data. One problem is that the validation method itself may be wrongly selected or have unrealistic assumption (e.g., on the distribution of data). Evaluating quality of clustering is especially difficult, because the validation method may assume a different clustering objective than intended. E.g., many famous internal validation indices cannot detect good spectral clustering, because the clusters are arbitrarily shaped.
5. Presenting and interpreting results: The results can be misinterpreted, because of misleading presentation. Visualizations are often inaccurate or even misleading (e.g., 2-D presentations of multidimensional data may not show real clustering). One should also supply all information that is needed for interpretation (e.g., obtaining excellent clustering index value is misleading, if all data points formed singleton clusters). One should also understand what conclusions can be drawn from models and patterns. A classical error is to assume that statistical dependence means causality, sometimes with serious consequences (vs. Simpson's paradox).

Recap lecture

1. Exam and tips for preparation
2. Main learning goals
 - common sources of errors and misleading results
 - list of main learning goals
3. Selected recap topics (based on the questionnaire)

Exam Wed 13.12. 2023 13:00–16:00

Undergraduate Centre, A-sali (Aalto-sali) Y202a

- **non-programmable calculator** capable of roots, trigonometric and logarithmic calculations needed
- all advanced mathematical formulations will be provided in the exam paper
- both explanation and calculation/proof tasks (may be in the same task)
 - e.g., brief explanations what X means and why is it important or how it differs from Y
 - e.g., given toy data calculate/simulate/show something (measures, clustering, frequent sets/their condensed representations, association rules, graph patterns, ...)

Tricks for (deep) learning

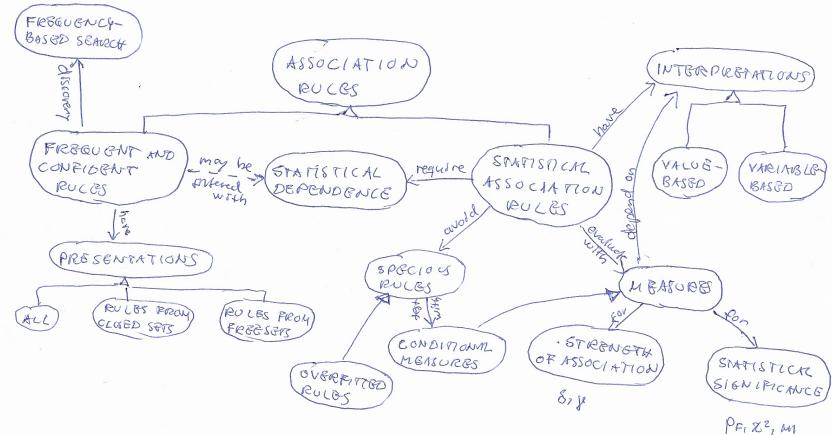
1. Motivate yourself

- Why this is important i) in general ii) to me some day?

2. Create big pictures and see connections!

- summary tables comparing approaches or methods
- concept map showing main concepts and their relations
- connect new to something familiar (e.g., single linkage clustering \leftrightarrow connected components)

method	data type	cluster type	benefits	drawbacks
K-representatives				
<i>K</i> -means				
<i>K</i> -medoids				
...				
Hierarchical				
single-link				
...				
Graph-based				
Density-based				
Probabilistic				



Tricks for (deep) learning

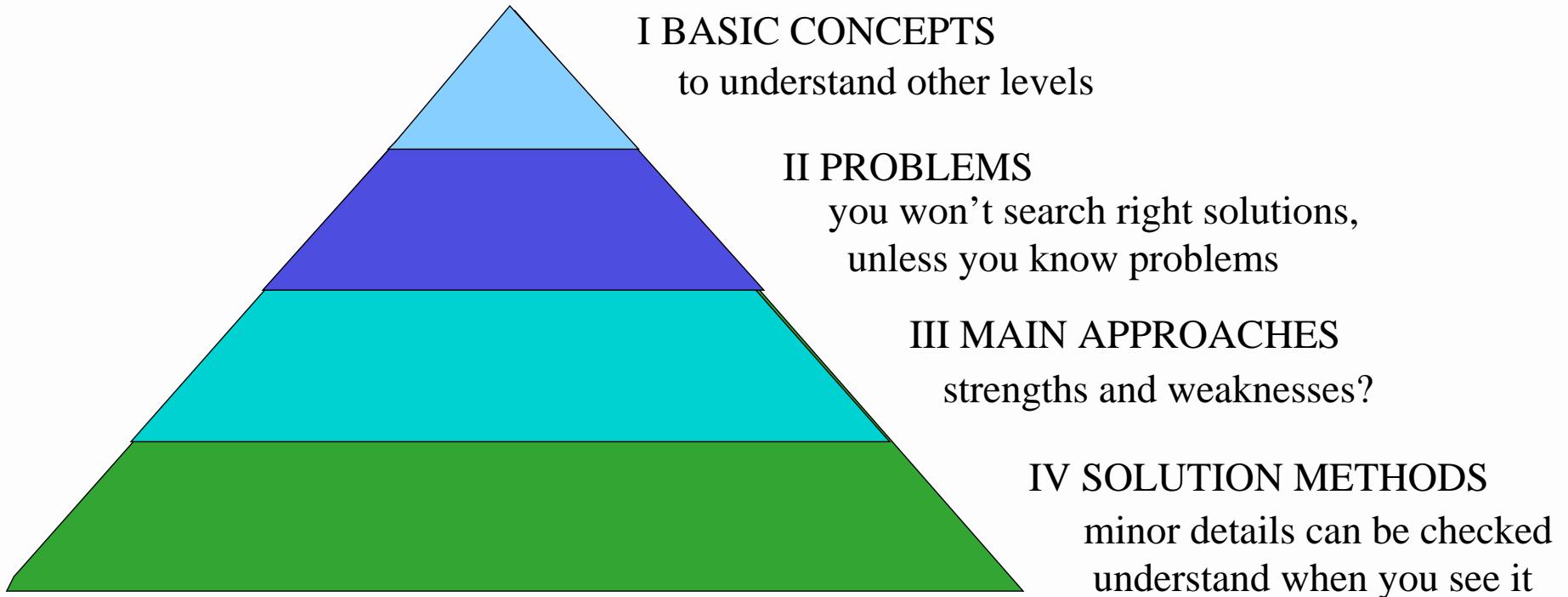
3. Try active learning

- Invent new problems to solve (How to apply *A* in *B*?)
- Invent and calculate examples
- Ask yourself questions (**what-why-how?** What is good/bad here? What happens if...)
- Write things on your own words and notations, make schematic drawings

4. Set yourself learning goals (I should understand *A*)

- evaluate which goals you have met
- soon a list of main learning goals → check and tailor for yourself
- check the recap topic wish (asking what-why-how or good/bad)

Learning DM: Different levels



Most important learning goal

Become **aware of error sources** at each step of the DM process and do less erroneous modelling (aim error-free!)

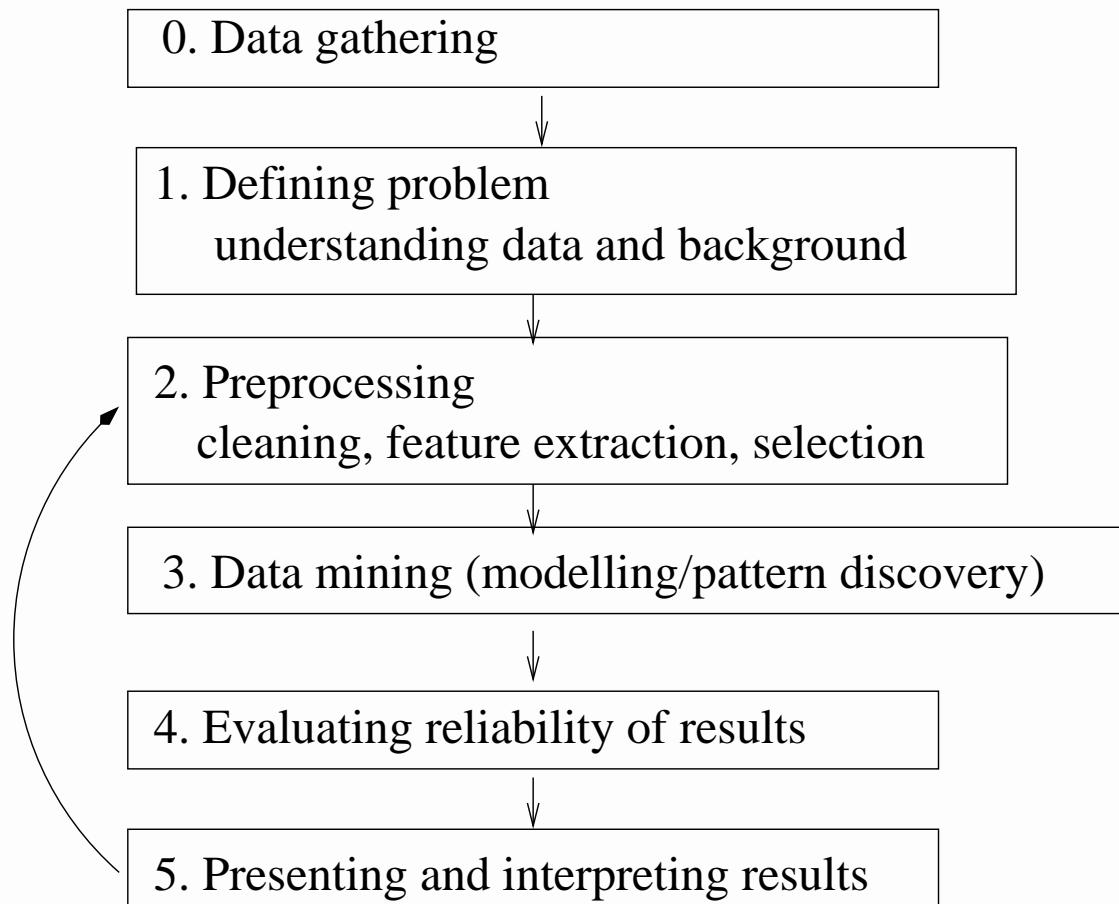
If you cannot solve something correctly, it is better to know/tell errors than believe the solution is correct

Always ask:

- **Does this make sense?**
 - does data look sensible?
 - is method X sensible (here/at all)?
 - do results look sensible?
 - is the validation method sensible?
- **What could go wrong here?**

Recall: KDD process

with additional step 0



Most common sources of errors

- trusting accuracy of data sources/search engines
- errors and anomalies in data
- ignoring data types (e.g., PCA on categorical data)
- being unaware of model/algorithm assumptions (or not checking if they fit the problem)
- blind use of ready-made libraries (what do they calculate?)
- not testing/verifying own programs
- not checking all intermediate results
- no validation/wrong validation
- unjustified conclusions/misleading presentation of results

Always suspect everything! (~ Cartesian doubt)

Main topics and learning goals

- I **DM process**: Be able to plan a valid DM project.
 - Know error sources and strategies
- II **Preprocessing**: Know tasks and strategies (missing data, scaling, **discretization**, dimension reduction (**PCA & SVD**)
- III **Distance and similarity**: Know common **measures** (for numerical, categorical and mixed; sets, strings, text and graphs). How to construct **similarity graphs**? What is the **curse of dimensionality** and remedies to it?
 - When a distance measure is metric and why it is useful?

Main topics and learning goals

- IV **Clustering:** Know clustering objectives and what elements affect clustering results. How to study clustering tendency? What methods to try for a given data and clustering objective? **K-representatives, hierarchical and spectral clustering**, their assumptions, strengths and weaknesses. How to validate clustering?
- V **Association mining:** Know **different types of association patterns**. How to evaluate strength and statistical significance of association? How to utilize **monotonicity** in search? **Apriori** algorithm and computational strategies. Problem of overfitted and specious rules. Condensed representations of frequent sets and their problems.

Main topics and learning goals

- VI **Other pattern discovery:** How to search new pattern types with monotonic properties (**Generic Apriori**)? How to evaluate significance of discovered patterns with **randomization testing**? What is the multiple hypothesis problem?
- VII **Graph mining:** What are **graph isomorphism, isomorphic subgraphs and MCG**? How to measure distance between graphs, mine graph patterns or cluster graphs?
- VIII **Social network analysis:** What are the main tasks of social network analysis and approaches to solve them? Especially, how to detect communities or evaluate centrality or similarity of nodes?

Main topics and learning goals

- IX **Web mining and search:** Know how search engines work. How to rank documents (esp. **PageRank** and **HITS**)?
- X **Recommender systems:** Know main approaches, especially **collaborative filtering**.
- XI **Text mining:** How to preprocess text and present in vector space? How to get informative features? Special problems and techniques (esp. for clustering).

Recap topics

- Computationally demanding DM tasks
- Generic Apriori
- Generating rules from sets
- Evaluating statistical associations rules
- Idea of spectral clustering
- Co-clustering with a spectral example
- LSA (recaps also SVD)
- Mid-frequency terms and phrases
- Multiple hypothesis testing problem
- Types of association patterns
- Other topics? (existing material/questions)

Computationally demanding DM tasks

No polynomial time algorithms known (most are *NP-hard*):

problem	What to do?
optimal feature selection	use heuristics (like greedy)
exact K -means clustering	use heuristic K -means alg.
best classification rule	usually scalable (if direct search), restrict complexity, use \min_{fr}
best association rule	
(sub)graph isomorphism	
MCS-distances and graph edit distance	impractical for larger graphs, consider alternative (transformation-based) distances
frequent subgraph discovery	restrict complexity, use higher \min_{fr}
graph partitioning	solve relaxed problem (spectral) or use heuristic community detection alg.

Computationally demanding DM tasks

- exponential search space not necessarily a problem, if effectively pruned and the data is nice (e.g., sparse transaction data)
- polynomial time algorithms can also be expensive!
 - if the same routine is repeated multiple times (e.g., all pairwise distances)
 - hierarchical (at least $O(n^2)$) and spectral ($O(n^3)$) clustering can be heavy for large data (vs. K-means $O(nKq)$, q =number of iterations)
- space (memory) may also be the bottleneck

Emphasis: Know when to expect heavy computation and if your method is solving the exact problem or some approximation/constrained version

Generic Apriori

- Decide general pattern type (e.g., sets, graphs, sequences).
- Define subpattern relationship (subset, subgraph, subsequence). Notate $\beta \subseteq \alpha$, if β is a subpattern of α .
- Define complexity of the pattern. Notate $|\alpha|$. (We say that α is an i -pattern, if $|\alpha| = i$, $i = 1, 2, \dots$)
- Decide interesting binary property Φ and check it is **monotonic!** (Otherwise Apriori doesn't work!)

ϕ monotonic if for all $\beta \subseteq \alpha$: if $\Phi(\beta) = 0$, then $\Phi(\alpha) = 0$

Examples of monotonic properties Φ

1. Frequent subgraphs (given min_{fr})

$$\Phi(\mathbf{G}) = \begin{cases} 1 & \text{if } fr(\mathbf{G}) \geq min_{fr} \\ 0 & \text{otherwise} \end{cases}$$

2. Classification rules $\mathbf{X} \rightarrow C$ (C fixed) with potentially high δ (given min_δ)

$$\Phi(\mathbf{X} \rightarrow C) = \begin{cases} 1 & \text{if } P(\mathbf{X}C)P(\neg C) \geq min_\delta \\ 0 & \text{otherwise} \end{cases}$$

(Note: Φ is for pruning by monotonicity. For actual discoveries, check that $\delta(\mathbf{X}, C) \geq min_\delta$. See Exercise 3.3)

Generic Apriori

Notations: \mathcal{F}_i = i -patterns having property Φ , C_i = candidate i -patterns

Algorithm:

$$i = 1; \mathcal{F}_1 = \{\alpha \mid \Phi(\alpha) = 1, |\alpha| = 1\}$$

while $\mathcal{F}_i \neq \emptyset$

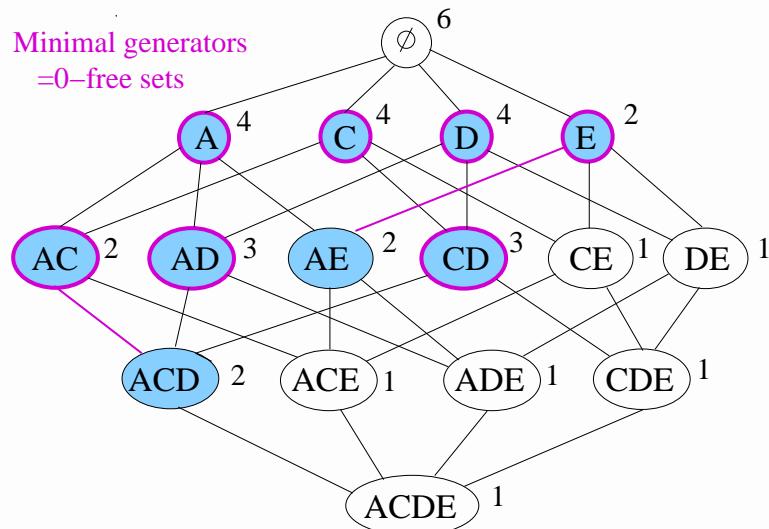
- Generate candidates C_{i+1} from \mathcal{F}_i
 - Prune $\alpha \in C_{i+1}$ if $\exists \beta \subseteq \alpha, |\beta| = i, \beta \notin \mathcal{F}_i$
 - Evaluate Φ for all $\alpha \in C_{i+1}$
 - Set $\mathcal{F}_{i+1} = \{\alpha \in C_{i+1} \mid \Phi(\alpha) = 1\}$
 - $i = i + 1$
- } (monotonicity)

Return $\cup_i \mathcal{F}_i$

Generating rules from sets $X \in \mathcal{F}$

Idea: For all $|X| \geq 2$ and all $C \in X$, test rule $X \setminus \{C\} \rightarrow C$ (and possibly $X \setminus \{C\} \rightarrow \neg C$) with the given measures

Example: given frequent sets with $fr \geq 2$ (blue), find positive consequent rules with $\phi \geq 0.6$ and $\delta \geq 0.1$.

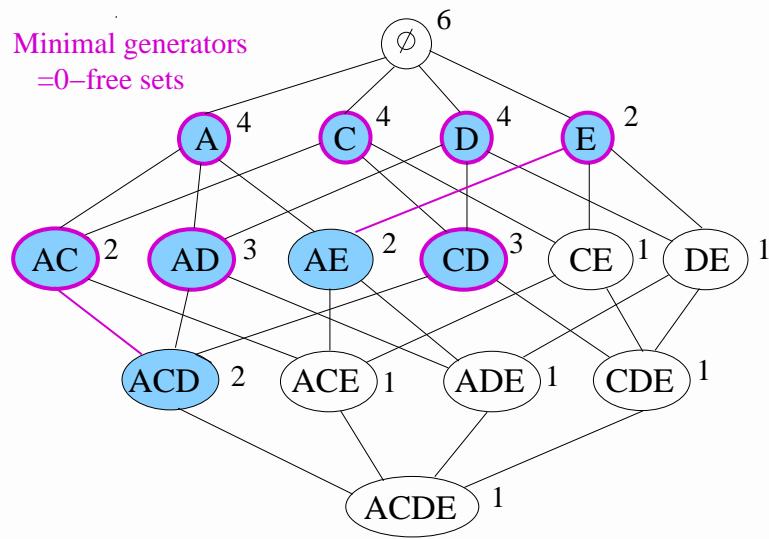


	ϕ		ϕ	δ
$A \rightarrow C$	$\frac{2}{4}$	$C \rightarrow A$	$\frac{2}{4}$	$-\frac{1}{9}$
$A \rightarrow D$	$\frac{3}{4}$	$D \rightarrow A$	$\frac{3}{4}$	$\frac{1}{18}$
$A \rightarrow E$	$\frac{2}{4}$	$E \rightarrow A$	$\frac{2}{2}$	$\frac{1}{9}$
$C \rightarrow D$	$\frac{3}{4}$	$D \rightarrow C$	$\frac{3}{4}$	$\frac{1}{18}$
$AC \rightarrow D$	$\frac{2}{2}$			$\frac{1}{9}$
$AD \rightarrow C$	$\frac{2}{3}$			0
$CD \rightarrow A$	$\frac{2}{3}$			0

Generating rules from sets $X \in \mathcal{F}$

Example continued: Identify 0-free frequent sets with $fr \geq 2$ and find positive consequent rules with $\phi \geq 0.6$ and $\delta \geq 0.1$.

- Recall: X 0-free if for all $Y \subsetneq X$ holds $P(Y) > P(X)$



	ϕ		ϕ	δ
$A \rightarrow C$	$\frac{2}{4}$	$C \rightarrow A$	$\frac{2}{4}$	$-\frac{1}{9}$
$A \rightarrow D$	$\frac{3}{4}$	$D \rightarrow A$	$\frac{3}{4}$	$\frac{1}{18}$
$C \rightarrow D$	$\frac{3}{4}$	$D \rightarrow C$	$\frac{3}{4}$	$\frac{1}{18}$

No rules found!

Lesson: Condensed representations can miss the best association rules!

Evaluating statistical associations rules

Given rule $\mathbf{X} \rightarrow C=c$, $c \in \{0, 1\}$. Evaluate

1. Statistical dependence: Is

$$\delta(\mathbf{X}, C=c) = P(\mathbf{X}, C=c) - P(\mathbf{X})P(C=c) > 0 \text{ or}$$

$$\gamma(\mathbf{X}, C=c) = \frac{P(\mathbf{X}, C=c)}{P(\mathbf{X})P(C=c)} > 1? \text{ If not, prune out.}$$

2. Evaluate significance, using measure M and its threshold.
(E.g., required $MI \geq \min_{MI}$) If not significant, prune out.
3. Evaluate overfitting: does $\mathbf{X} \rightarrow C=c$ improve significantly
(using conditional measure M_c) all $\mathbf{Y} \rightarrow C=c$, $\mathbf{Y} \subsetneq \mathbf{X}$?
 - i) If $P(C=c|\mathbf{X}) \leq P(C=c|\mathbf{Y})$ for some $\mathbf{Y} \subsetneq \mathbf{X}$, prune out.
Why?
 - ii) If $M_c(\mathbf{X} \rightarrow C=c | \mathbf{Y} \rightarrow C=c)$ not sufficient (given threshold)
for some $\mathbf{Y} \subsetneq \mathbf{X}$, prune out. (E.g., $MI_C < \min_{MI_C}$)

Note: 3ii) is for value-based associations. For variable-based associations,
you should check also $M_c(\neg\mathbf{Y} \rightarrow C \neq c | \mathbf{X} \rightarrow C \neq c)$ (extra stuff).

Idea of spectral clustering

1. Create similarity graph \mathbf{G} (or use a social network).
2. Present \mathbf{G} in (low-dimensional) vector space as \mathbf{Y} such that distances $d(\mathbf{y}_i, \mathbf{y}_j)$ ($\mathbf{y}_i, \mathbf{y}_j \in \mathbf{Y}$) reflect edge weights w_{ij} in \mathbf{G} .
 - create graph Laplacian \mathbf{L} (normalized or unnormalized)
 - take first K eigenvectors of \mathbf{L} with smallest λ s $\stackrel{a}{\rightarrow} \mathbf{u}_1, \dots, \mathbf{u}_K$
 - create data matrix \mathbf{Y} using $\mathbf{u}_1, \dots, \mathbf{u}_K$ as columns
 - if \mathbf{L}_{sym} , normalize rows to unit length $\stackrel{b}{\rightarrow}$
3. Cluster \mathbf{Y} (usually K -means).

^aexcept $\lambda = 0$, since then $\mathbf{y} \propto (1, 1, \dots, 1)$

^bWith \mathbf{L}_{rw} , normalize columns to unit length, unless already done.

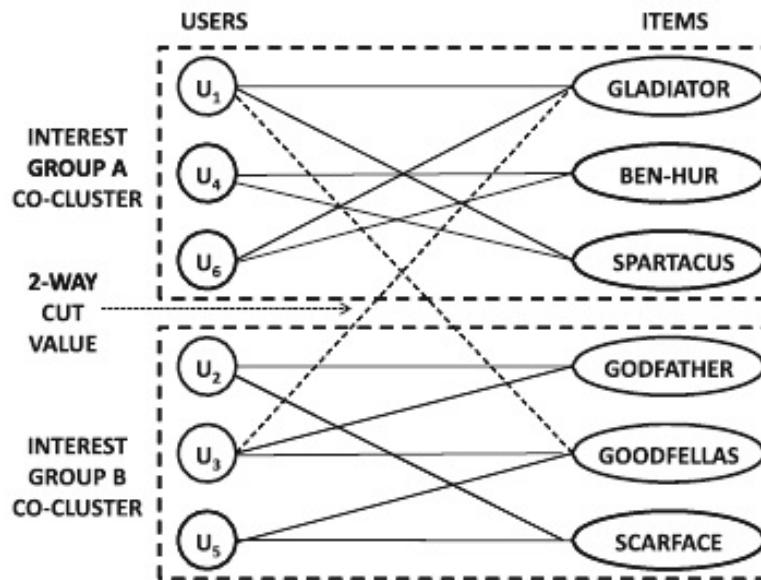
Co-clustering with a spectral example

When you want to cluster both rows and columns, especially

- users and items (recommender systems)
- documents and words (text mining)
- most effective when sparse data

INTEREST GROUP A CO-CLUSTER	GLADIATOR	BEN-HUR	SPARTACUS	GODFATHER	GOODFELLAS	SCARFACE
U ₁	1		1		1	
U ₄		1	1			
U ₆	1	1				
U ₂				1		1
U ₃	1			1	1	
U ₅					1	1

(a) Co-cluster



(b) User-item graph

Example from Aggarwal Fig 18.6.

Write 12×12 adjacency matrix $\mathbf{W} \rightarrow$ Laplacian \mathbf{L}

Co-clustering with a spectral example

Using unnormalized Laplacian. 2nd eigenvector (column 2) clusters users (rows 1–6) and items (rows 7–12).

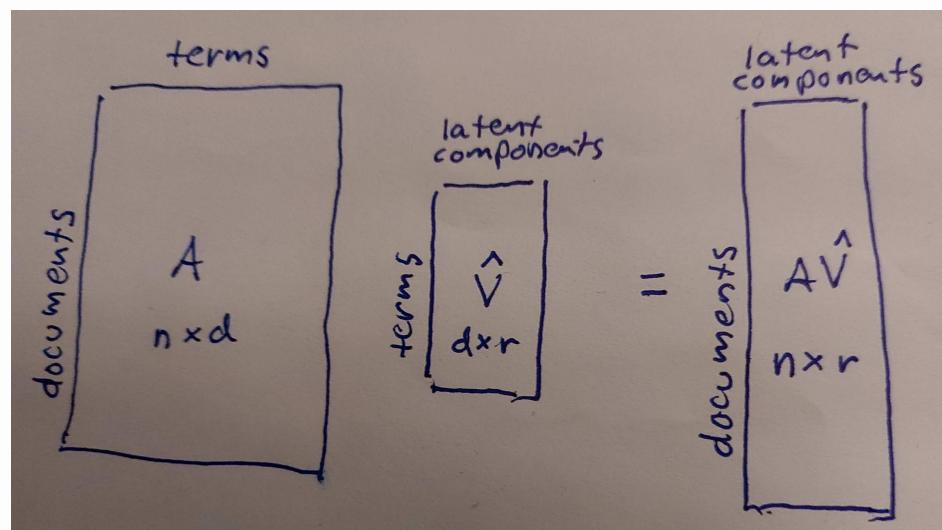
```
Eigenvalues: [1.45040194e-15 2.67949192e-01 1.00000000e+00 1.00000000e+00  
1.18639350e+00 2.00000000e+00 3.00000000e+00 3.00000000e+00  
3.47068342e+00 3.73205081e+00 4.00000000e+00 5.34292308e+00]  
E-vectors  
[[ 0.28867513 -0.10566243 0.40507055 -0.05083812 0.11190063 -0.28867513  
-0.00561909 -0.40820962 -0.13413108 -0.39433757 0.28867513 -0.46849451]  
[ 0.28867513 -0.39433757 0.24656238 -0.32538233 0.23530186 0.28867513  
0.35632945 0.19923853 0.43581073 -0.10566243 0.28867513 -0.06857143]  
[ 0.28867513 -0.28867513 -0.15850817 0.37622045 -0.42674499 -0.28867513  
-0.35071035 0.20897109 0.20512889 0.28867513 0.28867513 -0.16065758]  
[ 0.28867513 0.39433757 -0.40507055 0.05083812 0.23530186 -0.28867513  
-0.00561909 -0.40820962 0.43581073 0.10566243 -0.28867513 -0.06857143]  
[ 0.28867513 0.10566243 0.24656238 0.32538233 0.11190063 0.28867513  
0.35632945 0.19923853 -0.13413108 0.39433757 -0.28867513 -0.46849451]  
[ 0.28867513 0.28867513 0.15850817 -0.37622045 -0.42674499 0.28867513  
-0.35071035 0.20897109 0.20512889 -0.28867513 -0.28867513 -0.16065758]  
[ 0.28867513 -0.10566243 0.24656238 0.32538233 -0.11190063 -0.28867513  
0.35632945 0.19923853 0.13413108 -0.39433757 -0.28867513 0.46849451]  
[ 0.28867513 -0.39433757 0.40507055 0.05083812 -0.23530186 0.28867513  
-0.00561909 -0.40820962 -0.43581073 -0.10566243 -0.28867513 0.06857143]  
[ 0.28867513 -0.28867513 0.15850817 -0.37622045 0.42674499 -0.28867513  
-0.35071035 0.20897109 -0.20512889 0.28867513 -0.28867513 0.16065758]  
[ 0.28867513 0.28867513 -0.15850817 0.37622045 0.42674499 0.28867513  
-0.35071035 0.20897109 -0.20512889 -0.28867513 0.28867513 0.16065758]  
[ 0.28867513 0.10566243 0.40507055 -0.05083812 -0.11190063 0.28867513  
-0.00561909 -0.40820962 0.13413108 0.39433757 0.28867513 0.46849451]  
[ 0.28867513 0.39433757 0.24656238 -0.32538233 -0.23530186 -0.28867513  
0.35632945 0.19923853 -0.43581073 0.10566243 0.28867513 0.06857143]]
```

LSA (*latent semantic analysis*)

LSA = SVD applied to document-term (or term-document) matrix

$$A_{n \times d} = \hat{U}_{n \times r} \mid \Sigma_{r \times r} \mid \hat{V}^T_{r \times d}$$

$U_{n \times n}$ $\Sigma_{n \times d}$ $V^T_{d \times d}$



- new presentation to documents $A\hat{V} = \hat{U}\hat{\Sigma}$
- and for terms $A\hat{U} = \hat{V}\hat{\Sigma}$
- reduces dimensionality!
- helps with synonymy (similar terms tend to be combined)
- may help with polysemy (if one main meaning)

Note: If term features F_1, \dots, F_d and i th singular vector $(V_{i1}, \dots, V_{id})^T$, i th new feature $F'_i = \sum_{j=1}^d V_{ij} F_j$.

Image: Perunicic (2017).

LSA toy example

document-term matrix with tf-idf weights

Documents:

(cat)² (lion)² (kitty)

(cat)² (lion)³ (kitty)²

(cat) (lion) (kitty) (jaguar)

(jaguar)³ (car)² (engine)

(jaguar)² (car) (engine)

(jaguar)² (car) (engine)²

$$\mathbf{D} = \begin{bmatrix} 2 & 2 & 1 & 0 & 0 & 0 \\ 2 & 3 & 2 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0.585 & 0 & 0 \\ 0 & 0 & 0 & 1.755 & 2 & 1 \\ 0 & 0 & 0 & 1.170 & 1 & 1 \\ 0 & 0 & 0 & 1.170 & 1 & 2 \end{bmatrix}$$

Here $tfidf(w_j, d_i) = fr(w_j|d_i) \cdot \log \frac{n}{n_j}$.

Note: $idf = \log(2) = 1$ for all words except for “jaguar”,
 $idf(jaguar) = \log(1.5) = 0.585$

LSA toy example: New presentation for documents

```
Singular values
[[5.34774741 4.09614197 1.06647632 0.60457418 0.52357289 0.07911012]
V^T
[[-5.54058090e-01 -6.97497899e-01 -4.50769743e-01 -5.26295418e-02
-1.71779747e-02 -1.60623926e-02]
[-2.67257237e-02 -3.90118618e-02 -1.78272243e-02 5.91518361e-01
5.79873106e-01 5.57941941e-01]
[ 1.51382506e-04 3.99531936e-02 -3.74464454e-02 -3.05214148e-01
-4.77894588e-01 8.21865299e-01]
[ 6.56873417e-01 -6.33949084e-02 -6.93413670e-01 -2.06029595e-01
2.02649099e-01 1.26897596e-02]
[-4.29978205e-01 4.52274325e-01 -1.27507972e-01 -5.65003826e-01
5.20427081e-01 6.50747650e-02]
[ 2.75574675e-01 -5.49370110e-01 5.45899068e-01 -4.38751600e-01
3.50938537e-01 9.26523906e-02]]
New data DV_2 using 2 singular vectors
[[-2.95388172 -0.1493024 ]
[-4.10214936 -0.20614148]
[-1.73311401  0.26247343]
[-0.14278319  2.75580288]
[-0.09481693  1.82989153]
[-0.11087932  2.38783347]]
```

New features approximately
*-0.55cat -0.70lion -0.45kitty and
0.59jaguar +0.58car +0.56engine*

How and why to increase frequency of mid-frequency terms and phrases?

1. **Too frequent** terms not specific, do not separate documents ⇒
 - remove stopwords and other too frequent words
 - idf (in $tf-idf$)
 - specialize terms using n-grams (e.g., “*data analysis*” and “*Bayesian data analysis*”, if “*data*” useless)
2. **Too rare** terms useless for detecting similarities ⇒
 - stemming and lemmatization
 - spell-checking
 - detect synonyms (e.g., WordNet)

Multiple hypothesis testing problem

Recall:

- Statistical hypothesis test: if $p \leq \alpha$, reject the null hypothesis (pattern is declared significant)
- type I error = null hypothesis is true but the pattern passes the test (a spurious discovery)
- In a single test, probability of type I error is $P(\text{error I}) \leq \alpha$

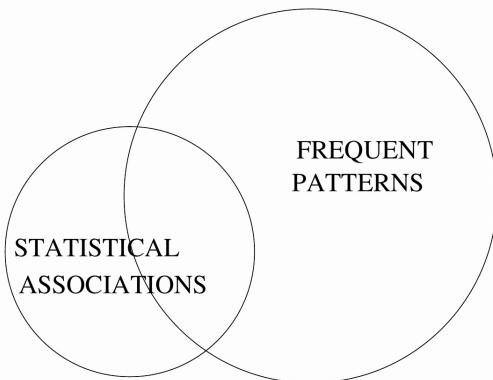
Problem: If you test m true null hypotheses and each test has $P(\text{error I}) = \alpha$, then $m \cdot \alpha$ spurious patterns will pass the test! \Rightarrow Classical thresholds $\alpha = 0.05$ or $\alpha = 0.01$ cannot be used in DM!

\Rightarrow multiple hypothesis testing correction methods (Bonferroni, Šídák, Hochberg, FDR correction methods, ... – not covered in the course)

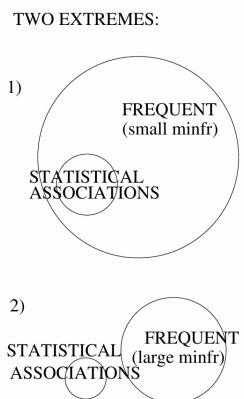
Short note: Types of association patterns

- Sets: frequent sets or statistical dependency sets
- Rules: frequent co-occurrence or statistical dependence (between condition and consequence)

Two approaches find generally different patterns!



If sufficiently small min_{fr} is feasible, you can filter statistical associations afterwards (but this can be heavy!)



Value-based and variable-based interpretation of association $X \rightarrow C = c$

Let I_X be an indicator variable:
 $I_X = 1$, if X holds, and $I_X = 0$, otherwise.

Important: Are we interested in association between values $I_X=1$ and $C=c$ or between binary variables I_X and C ?

⇒ different goodness measures and different results!

- **lift** γ measures strength of association between values
- **leverage** δ measures also strength of association between variables

Other topics (existing material/questions?)

- PCA assumptions: lec3
- Clustering tendency: lec3
- Efficient frequency counting (database projections, tid lists)
lec8
- Specious associations lec8
- K -representatives: lec4
- Linkage metrics: lec4

4. Yule-Simpson's paradox and other specious associations

Statistical dependence is a necessary but not a sufficient condition of causal relation!

- Often a **majority** of dependencies are **specious** (illusory, spurious, apparent) associations
- e.g., **cake → exam failure** was a sideproduct of **alcohol → exam failure** and **alcohol → cake**

Principal component analysis (PCA) assumptions

1. High **variance** reflects important structures of data.
2. Data can be presented well as a **linear** combination of suitable **orthogonal** basis vectors (PCs).

How to study clustering tendency and choose features?

Approaches:

1. Visual inspection of pairwise distance distributions
 - only hints
2. Filtering methods, e.g.,
 - Entropy-based measures
 - Hopkins statistic
3. Wrapper models + cluster validation indices
 - e.g., average silhouette, Calinski-Harabasz, Davies-Bouldin, and external indices