Knowledge Discovery in Databases

SS 2016

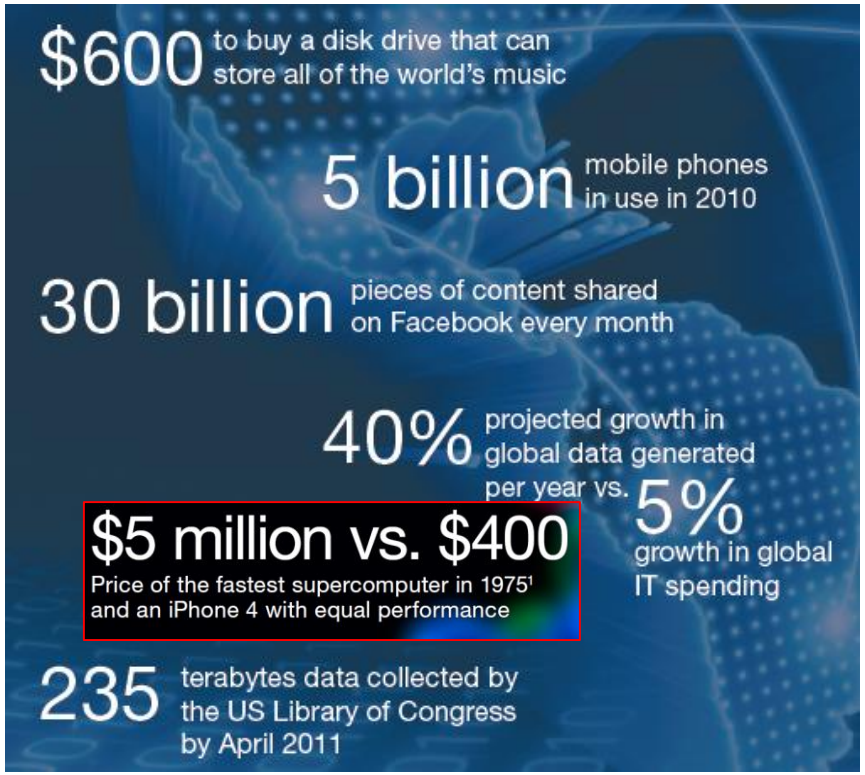
Epilog: Further Topics

Lecture: Prof. Dr. Thomas Seidl

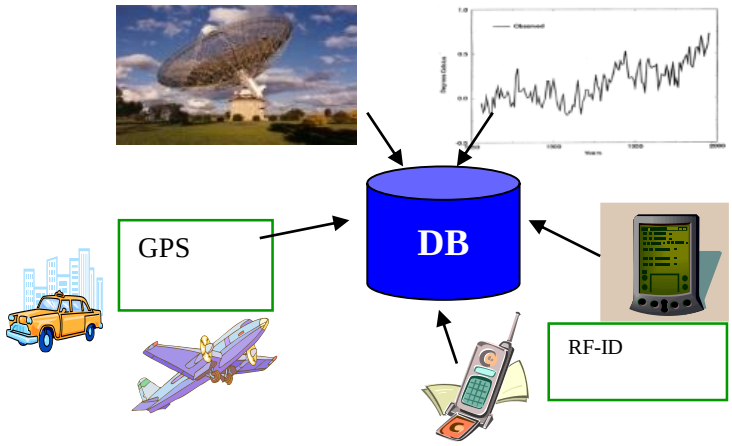
Tutorials: Julian Busch, Evgeniy Faerman,
Florian Richter, Klaus Schmid

We are drowning in data … but starving for information

- Exponential grows in data



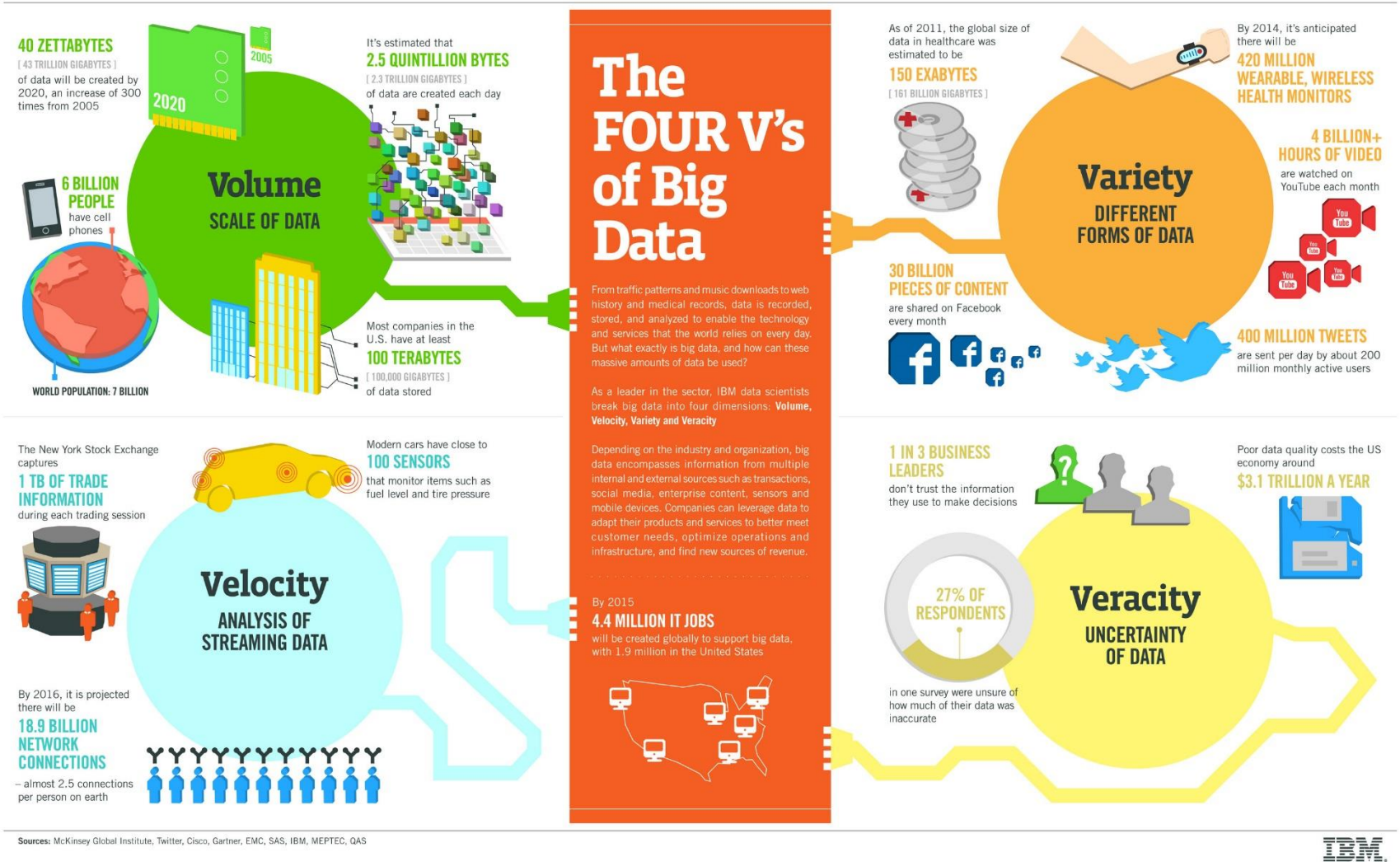
J. Leskovec, A. Rajaraman, J. Ullman: Mining of Massive Datasets, http://www.mmds.org



http://www.popsci.com/announcements/article/2011-10/november-2011-data-power

- Data contains value and knowledge

Four Vs of Big Data



Cluster computing

- Vertical scalability (scale up)
  - Increase capacity by adding more resources to single machine (processors, storage, memory)

- Horizontal scalability (scale-out)
  - Increase capacity by adding of more machines

https://www.greentree.com/latest-news/avoiding-cumulus-congestus

Cluster computing

- Distributed storage:
  - Distributed file systems (GFS (google), HDFS (hadoop), S3 Amazon,…)
  - NoSQL Databases

CAP Theorem: Any shared-data system can have at most two of the three desired properties!

Cluster computing

- MapReduce
  - Programming abstraction for parallel processing of large datasets proposed by Google
  - Programmer specifies the program as sequence of consecutive map and reduce functions
  - Programs are automatically parallelized and executed on cluster
  - Runtime system takes care of data partitioning and parallel execution

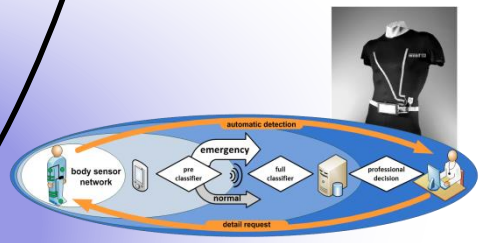
Stream Mining: Applications and tasks

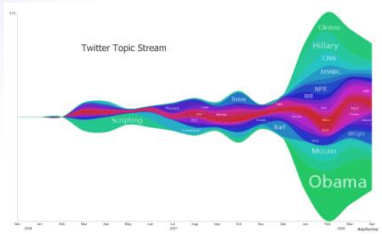
Classification

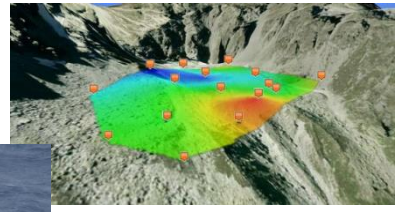
Modeling

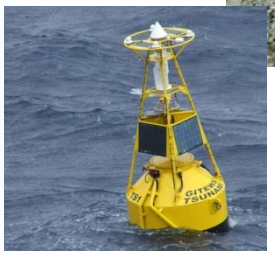
constant data rate

varying data rate









Outlier detection

Stream properties and resulting requirements

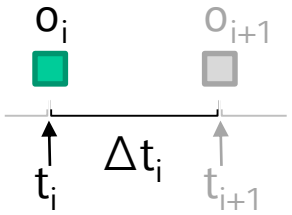
- Infinite stream
  - Single scan, missing random access
- Limited time
  - Fast access, cheap methods, sampling
- Limited memory
  - Compression
- Evolving distribution
  - Aging & updating models
- Noisy data
  - Noise handling
- Varying data rates → varying time allowances
  - Handle lowest time allowance, reduce idle times

Check list for stream algorithms:

- Single scan, missing random access
- Fast access, cheap methods
- Compression
- Aging & updating models
- Noise handling
- Handle varying data rates

Definitions

- Stream
  - A stream $S: \mathbb{N}_0 \rightarrow \mathbb{N}_0 \times Q: i \rightarrow (t_i, o_i)$ is an infinite sequence of objects $o_i \in Q$ from a d-dimensional input space Q and $t_i \in \mathbb{N}_0$, $t_i \leq t_j \forall i < j$ is the discrete arrival time of object o_i.
- Stream algorithms
  - Online algorithms – the input is given one at a time
  - Budget algorithms – tailored to a specific "real-time" budget b
  - Anytime algorithms – provide a result after any amount of processing time
- Inter-arrival time
  - The inter-arrival time between two consecutive objects o_i and o_{i+1} is denoted as $\Delta t_i = t_{i+1} - t_i$, i.e. $0 \leq \Delta t_i \in N$.
- Constant and varying streams
  - A stream S is called constant $\leftrightarrow$ $\Delta t_i = \Delta t_j \; \forall i, j$



Sampling and Buffering

- Sampling: draw a representative sample of the data distribution
- Apply a strategy to pick objects (online random sampling)
- Maintain a finite sample

- Good for computing statistics like expected values
- Impossible for anomaly detection or practical tasks like sorting

- Buffering: insert newly incoming objects into a buffer (FiFo queue) and process objects consecutively from the buffer
- Can reduce idle times
- Probability of failure (buffer overflow) depends on buffer size, budget, arrival distribution and length of the data stream

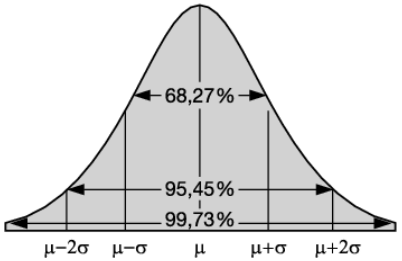
Stream statistics: mean, standard deviation, correlation

- Let $o_{i,j} \in \mathbb{R}$ be the value of object o_i in dimension j
- The sample mean μ_j of all objects $o_1 \ldots o_n$ seen so far is

$$\mu_j = \frac{\sum_{i=1}^{n} o_{i,j}}{n}$$

- The standard deviation σ_j of dimension j can be computed as

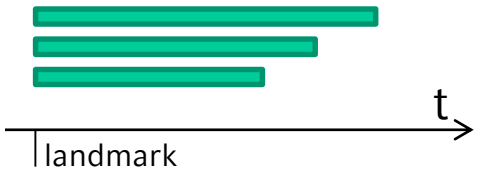
$$\sigma_j = \sqrt{\frac{\sum_{i=1}^{n} o_{i,j}^2}{n} - \left(\frac{\sum_{i=1}^{n} o_{i,j}}{n}\right)^2}$$



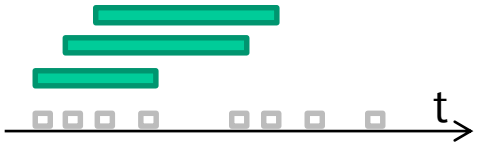
- All necessary statistics $(n, \sum o_{i,j}, \sum o_{i,j}^2, \sum o_{i,j} \cdot o_{i,k})$ can be maintained incrementally

Aging mechanisms

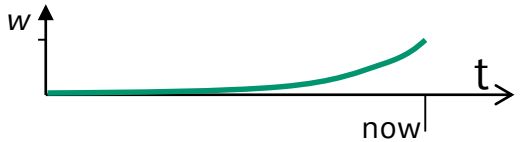
- Landmark window
  - Restart, reset



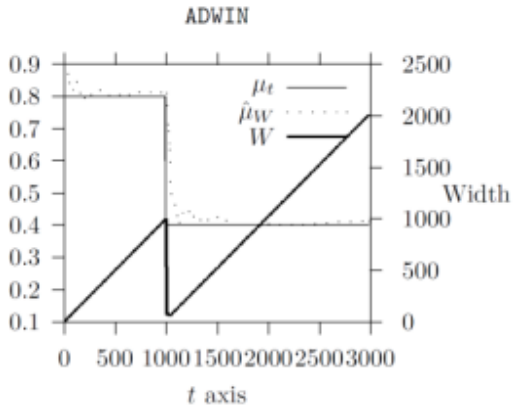
- Sliding window
  - Binary weighting

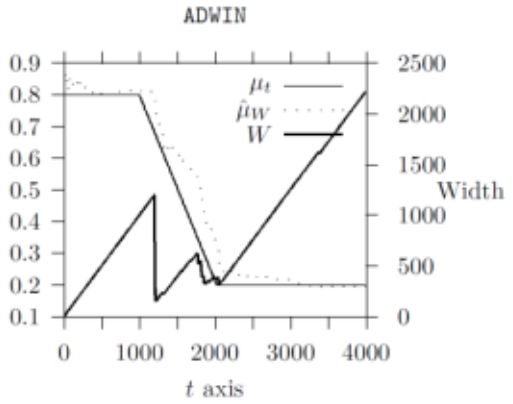


- Damped window
  - Usually linear or exponential decay

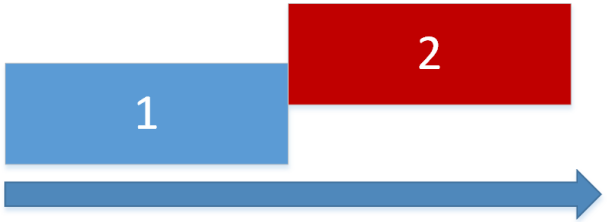


- Adaptive windowing
  - Compression wrt. mean shift

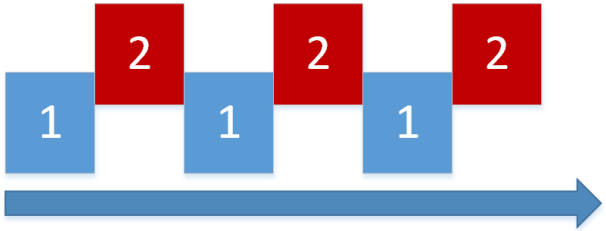




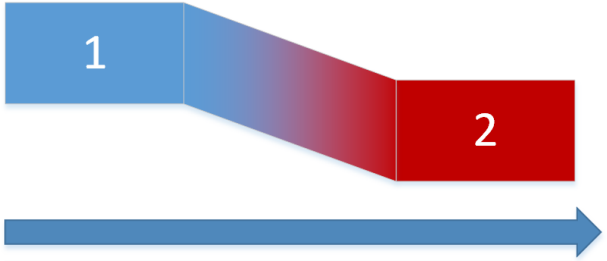
Concept drift



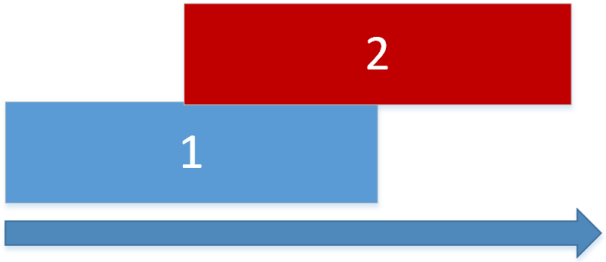
Sudden Drift:
Crisis situation
Emergency
Production issues
Industrial problems



Recurring Drift:
Seasonal impacts
Recurring events/expositions
Manufacturing cycles
Mostly planned rearrangements



Incremental Drift:
Step-wise adaption
Slowly changing conditions
Re-evaluating peacewise changes
Outer factors (laws, stock market)



Gradual Drift:
contract commitment
Keeping business operational
Fallback opportunity
Multi-interest company

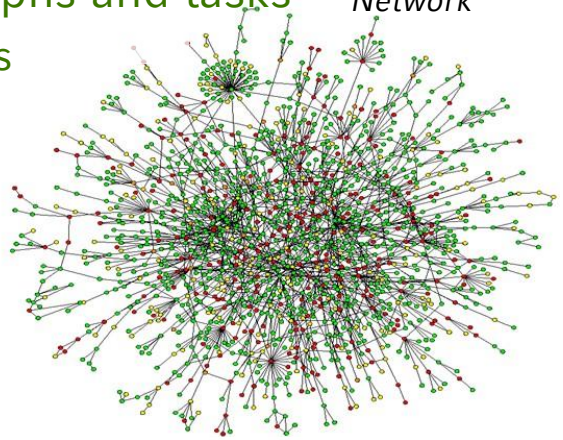
Graph Mining

- Graphs, graphs everywhere!
  - Chemical data analysis, proteins
  - Biological pathways/networks
  - Program control flow, traffic flow, work flow analysis
  - XML, Web, social network analysis

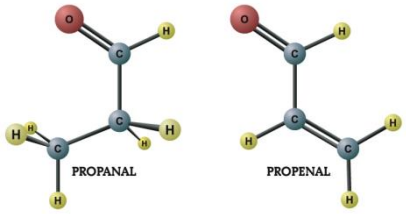
Social Network Graph (facebook, Dez 2010)

- Graphs form a complex and expressive data type
  - Trees, lattices, sequences, and items are degenerated graphs
  - Different applications result in different kinds of graphs and tasks
    - Diversity of graphs and tasks → diversity of challenges
  - Complexity of algorithms: many problems are of high complexity (NP-complete or even P-SPACE!)

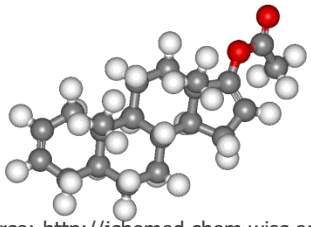
Yeast Protein Interaction Network



from H. Jeong et al Nature 411, 41 (2001)



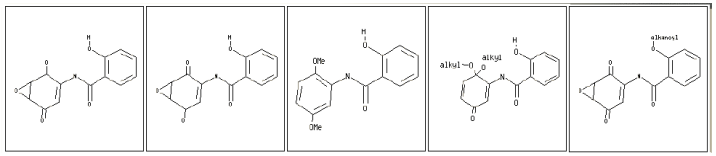
source: http://www.nrao.edu/pr/2004/GBTMolecules/

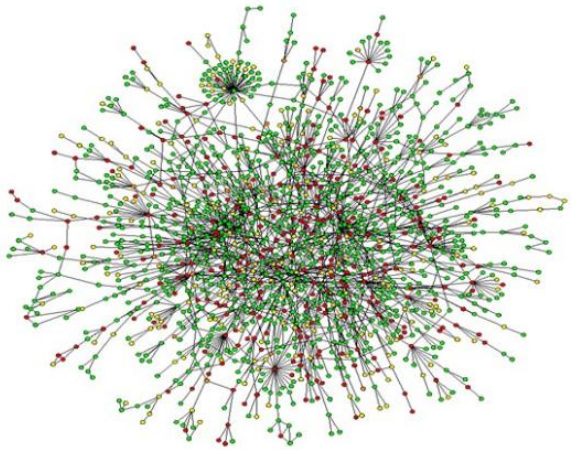


source: http://jchemed.chem.wisc.edu/

Graph Databases vs. Network Data

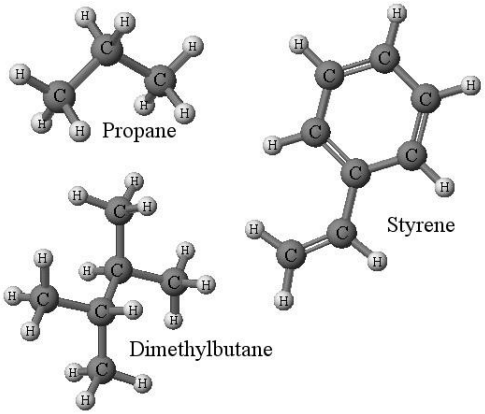
- Different applications result in different kinds of graphs and tasks
  - E.g. chemical graphs: relatively small, repeating vertex labels
  - E.g. large scale domains (web, computer networks, social networks): very big, vertex labels are distinct
- Diversity of graphs and tasks → diversity of challenges
- Graph mining can be divided into two fundamental settings:
  - Mining in a **set of graphs**, e.g.:
    - Finding similar graphs
    - Determining all frequent subgraphs
    - Classification of graphs
  - Mining in **one single large graph**, e.g.:
    - How does the network ´behave´?
    - Determine striking patterns, e.g. homogeneous and connected components

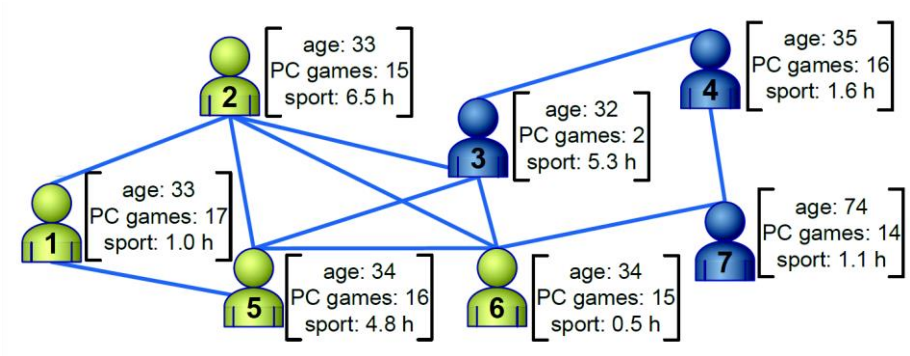


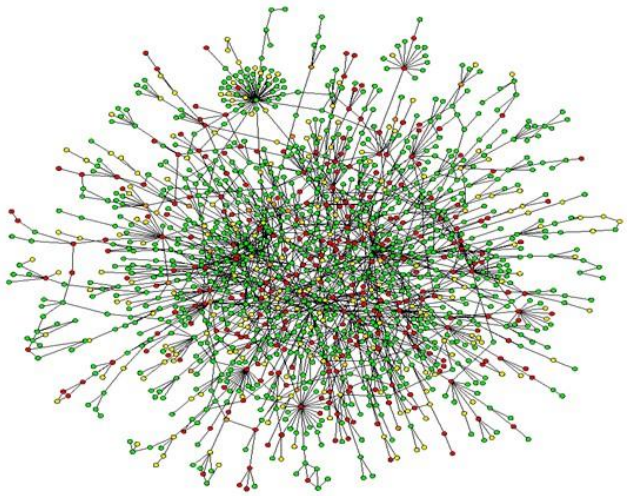


Graph Data: Mining Tasks

- Some interesting problems investigated in graph mining:
  - How to measure similarity between graphs?
  - How to find frequent patterns in a graph database?
  - How does a real graph look like?
  - How to identify groups in social networks?
  - How to integrate additional information into graph mining techniques?



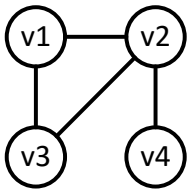




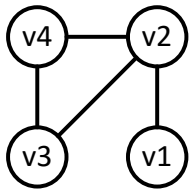
Graph Similarity

- Similarity between objects basic requirement for mining and exploration
  - Retrieval, Clustering, Classification, …
    - Many techniques rely on similarity/distance measures
- Traditional vector data: several distance functions introduced
  - Euclidean Distance, Cosine Distance, Mahalanobis Distance, …
- Similarity between graphs more complex
  - Arbitrary permutation of nodes still results in same graph

  → Computing, e.g., Frobenius norm („entrywise“ Euclidean Distance) between two adjacency matrices not meaningful

g_1 

$$M_1 = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

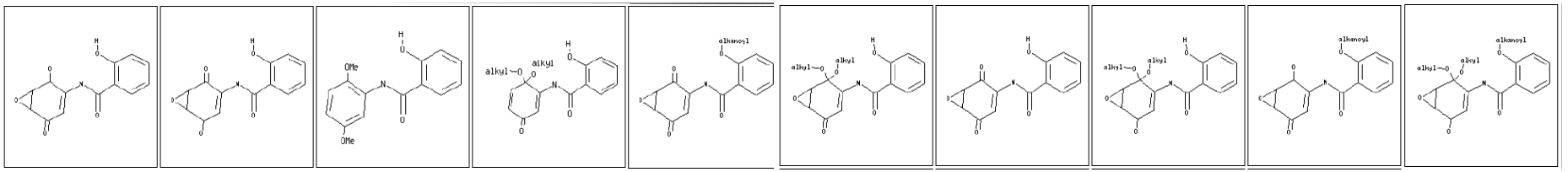
g_2 

$$M_2 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

$g_1 \cong g_2$
but
$\|M_1 - M_2\|_F = 2$

Frequent Subgraph Mining

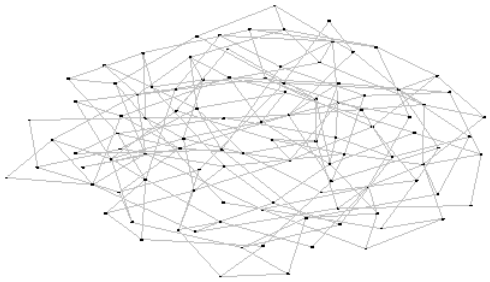
- Input: Collection of undirected labeled graphs
- Aim: Determine all connected graphs that occur as subgraph in at least a given percentage (*support*) or number (*frequency*) of all graphs in *DB*
- Analogy to "traditional" frequent itemset mining:
  - Each graph of the graph database represents a transaction
  - Each subgraph represents an itemset
- Applications:
  - As preprocessing: characterizing graph sets, discriminating different groups of graphs, classifying graphs, clustering graphs, building graph indices, facilitating similarity search
  - Bioinformatics, computer vision, video indexing, chemical informatics

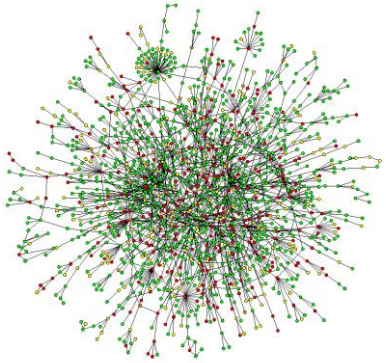


E.g. frequent molecular fragments (e.g. in drug discovery)

How does a „real“ network look like?

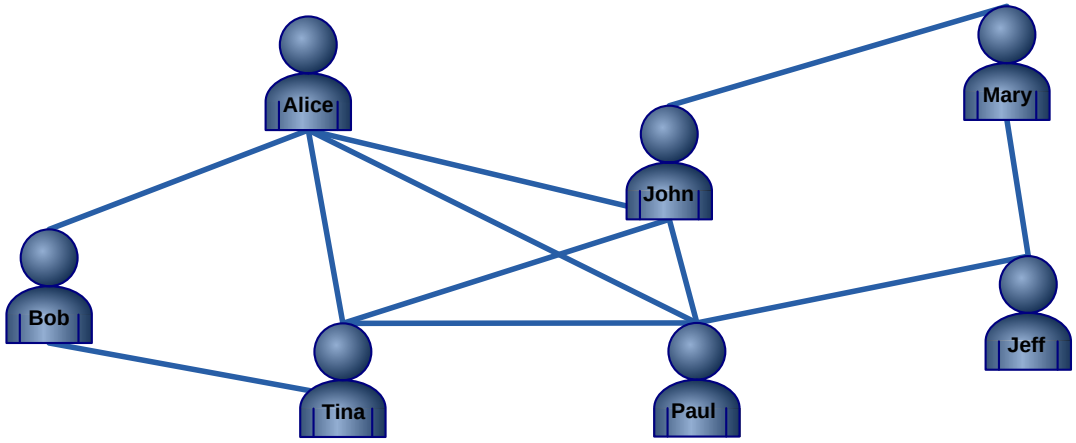
- For sure: dependent on application, different graphs possible
- But: Many real world networks follow certain rules; some characteristics show up regularly
- Important applications:
  - Detection of abnormal/interesting patterns
    - Specify what is ‘normal’
  - Development of graph generators
    - Run experiments on synthetic but realistic graphs
  - Simulation studies
    - E.g. test next-generation internet protocol on graph „similar“ to what Internet will look like a few years into the future
  - Realism of samples
    - Efficient testing on smaller samples of whole data
    - Samples must still reflect original characteristic





Clustering in Network Data

- Input: A graph
- Aim: Find clusters of *vertices* in the graph
- Related to "traditional" clustering (e.g. *k*-Means):
  - In traditional clustering, we cluster objects based on *attribute data*
  - Here, we cluster objects based on *graph data* (information about relationships between the objects)
- Example: Given a social network, find groups of people that are densely connected by "friendship" edges

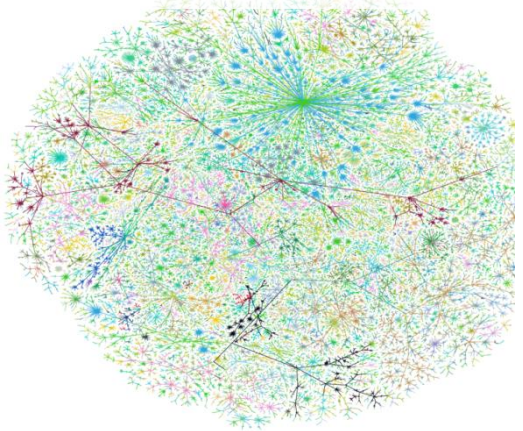


Clustering in Network Data

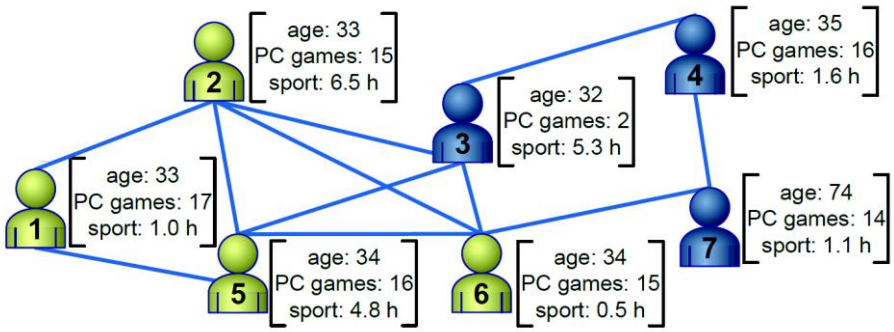
- Many different applications:
  - Friendship graph: find circles of friends
  - Protein-/Gene-Interaction Network: find groups of highly interacting proteins/genes
  - Internet graph: Find groups of websites with similar topics

Internet snapshot

source: the internet mapping project



- Extension: Integrated Clustering
  - Combined data sources: attribute information (for individual objects) **and** graph information, e.g.
    - Social networks: users' interests
    - Gene networks: expression values
    - Citation networks: conference/paper information



  - Find clusters with homogenous attribute values **and** cohesive subgraph structure