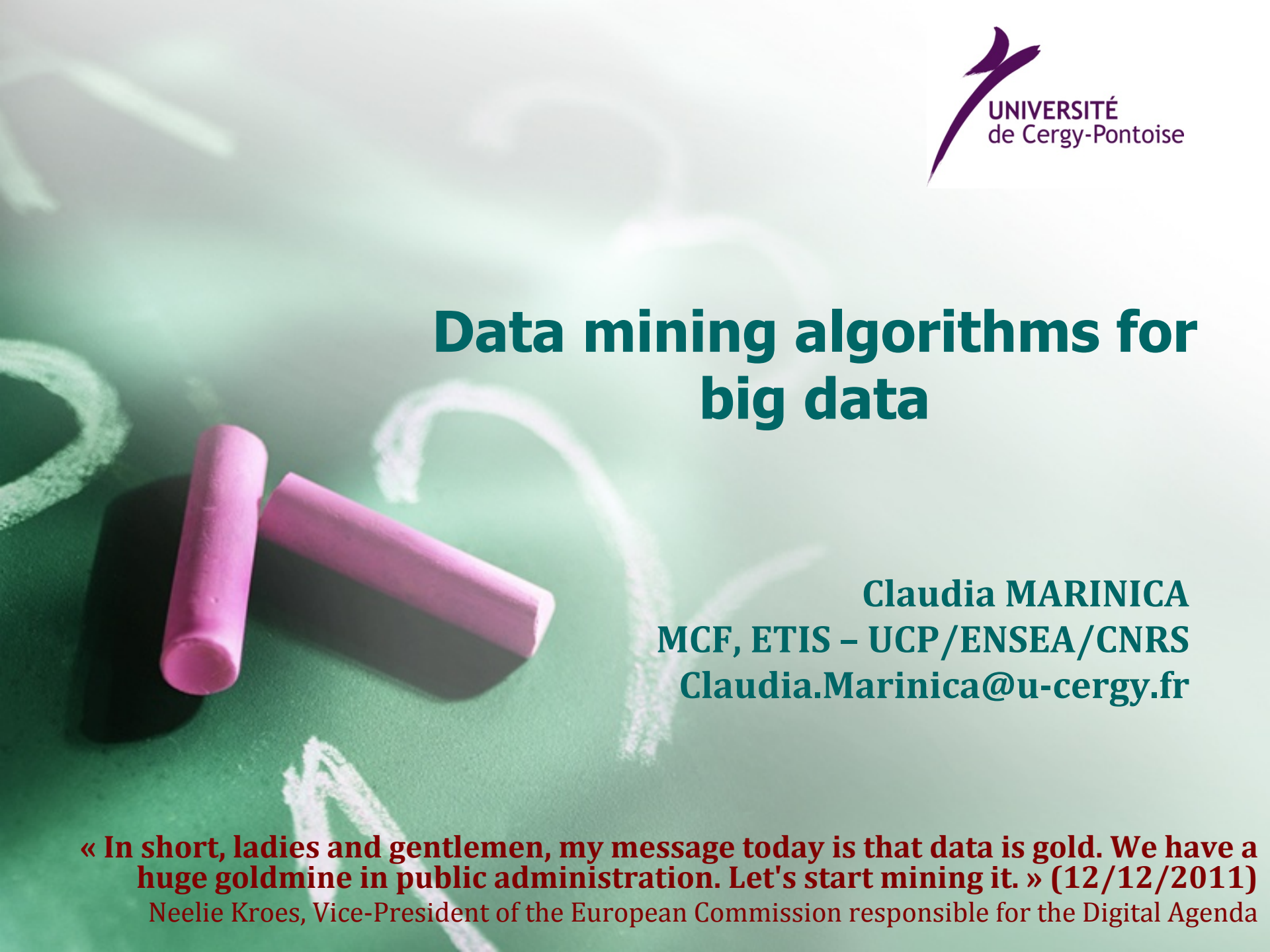


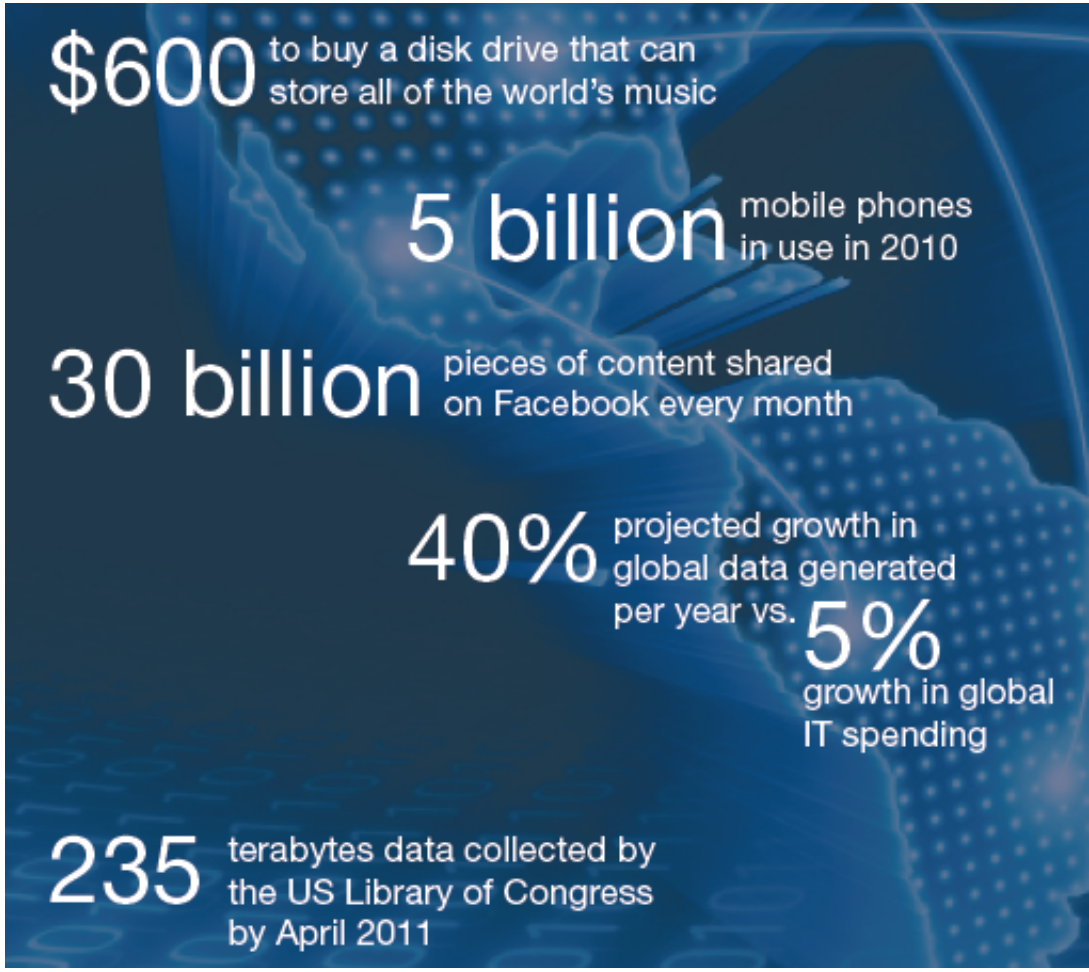
Data mining algorithms for big data

Claudia MARINICA
MCF, ETIS – UCP/ENSEA/CNRS
Claudia.Marinica@u-cergy.fr

« In short, ladies and gentlemen, my message today is that data is gold. We have a huge goldmine in public administration. Let's start mining it. » (12/12/2011)
Neelie Kroes, Vice-President of the European Commission responsible for the Digital Agenda

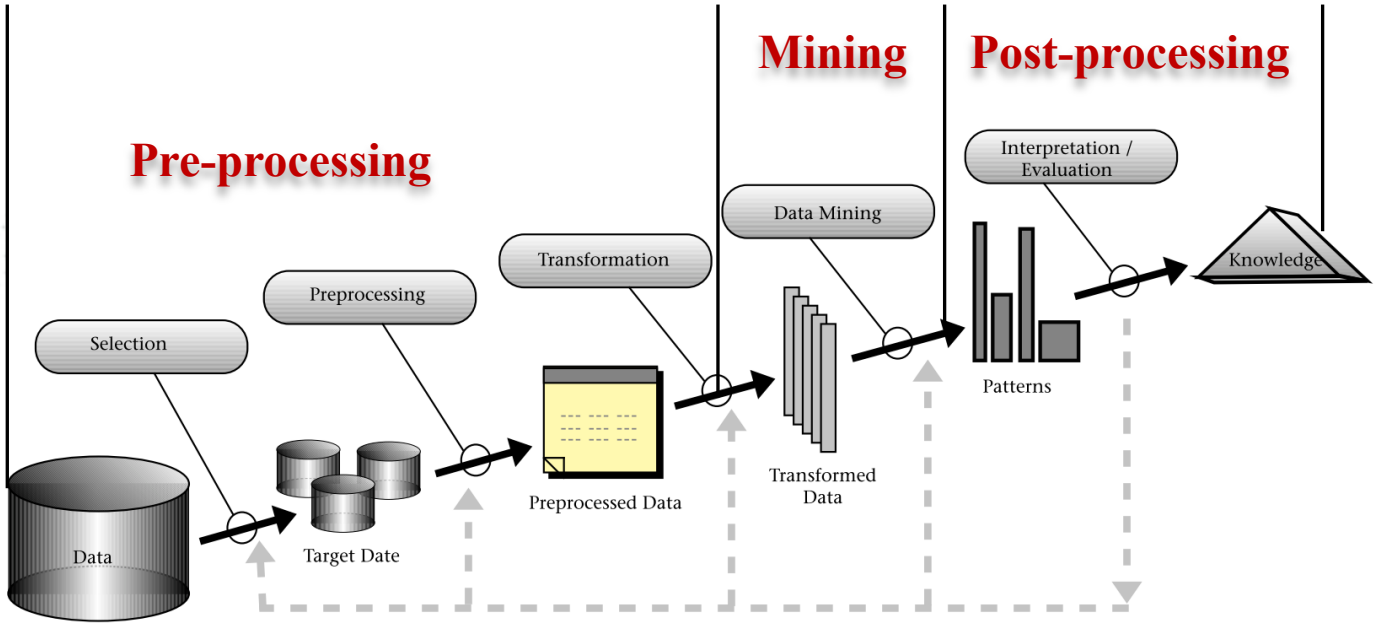
Why Knowledge Discovery from Databases (KDD)?

- Data available
- Limits of humans
- Several needs:
  - Industrial,
  - Medical,
  - Marketing,
  - …



KDD

« ... is the extraction of implicit, previously unknown, and potentially useful information from data . »



[Fayyad et al., 1996]

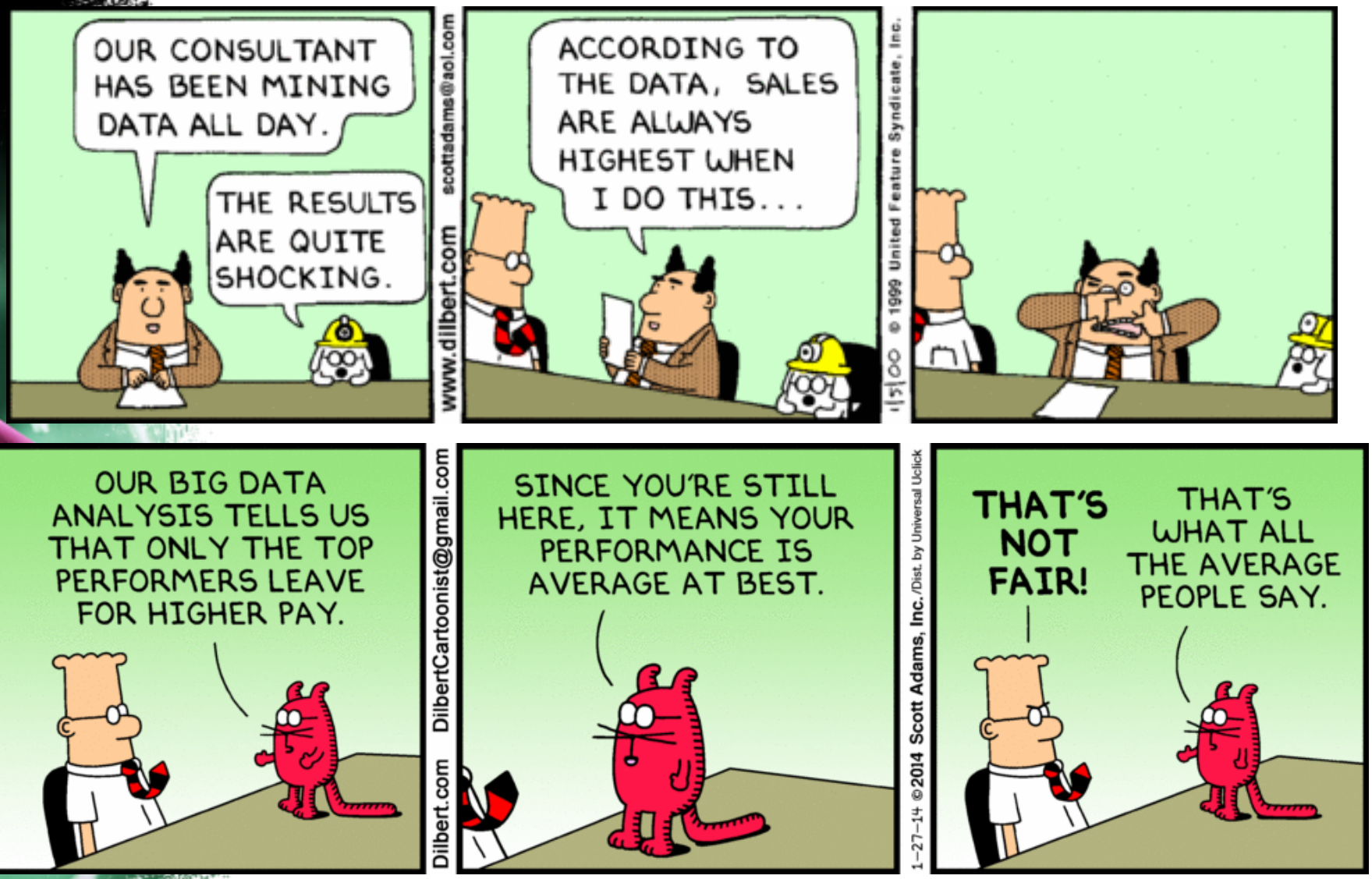
Valid: hold on new data with some certainty

Useful: should be possible to act on the item

Unexpected: non-obvious to the system

Understandable: humans should be able to interpret the pattern

KDD


OUR CONSULTANT HAS BEEN MINING DATA ALL DAY.
THE RESULTS ARE QUITE SHOCKING.
www.dilbert.com scottadams@aol.com
ACCORDING TO THE DATA, SALES ARE ALWAYS HIGHEST WHEN I DO THIS...
1|5|00 © 1999 United Feature Syndicate, Inc.
OUR BIG DATA ANALYSIS TELLS US THAT ONLY THE TOP PERFORMERS LEAVE FOR HIGHER PAY.
Dilbert.com DilbertCartoonist@gmail.com
SINCE YOU'RE STILL HERE, IT MEANS YOUR PERFORMANCE IS AVERAGE AT BEST.
THAT'S NOT FAIR!
THAT'S WHAT ALL THE AVERAGE PEOPLE SAY.
1-27-14 ©2014 Scott Adams, Inc./Dist. by Universal Uclick

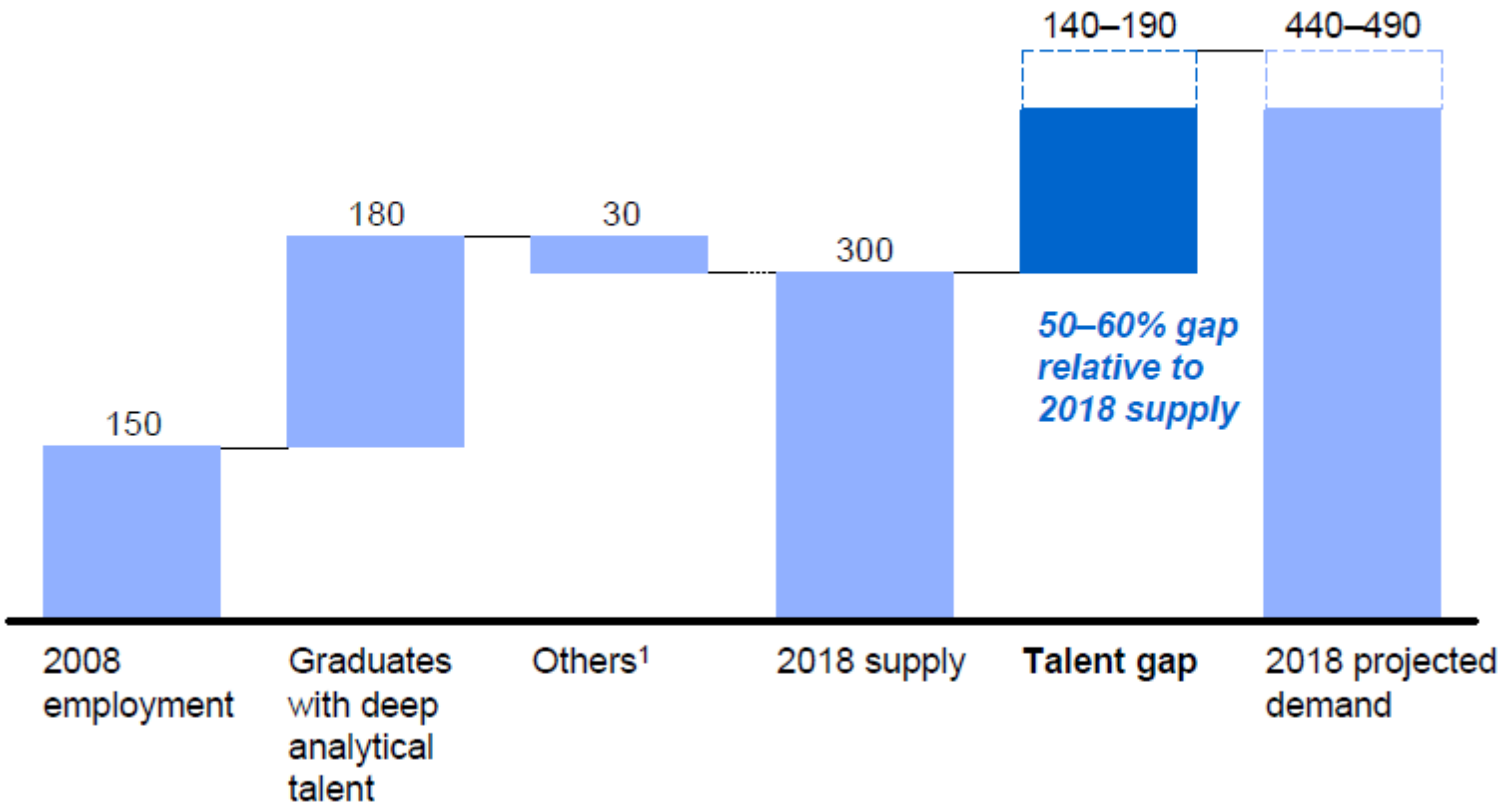
KDD
Goal: examples of applications

- Medical diagnosis
- Customers' profiling, mailing, bank loan decision, ...
- Hand writing recognition
- Finance, stock market previsions
- Customer Relationship Management (CRM) : find new and keep the old customers !
  - Fraud detection,
  - Detection of not reliable customers, …

KDD
Good news: increasing demand

Demand for deep analytical talent in the United States could be 50 to 60 percent greater than its projected supply by 2018

Supply and demand of deep analytical talent by 2018
Thousand people



1 Other supply drivers include attrition (-), immigration (+), and reemploying previously unemployed deep analytical talent (+).

SOURCE: US Bureau of Labor Statistics; US Census; Dun & Bradstreet; company interviews; McKinsey Global Institute analysis

KDD: Pre-processing

- Data integration from different sources (D. Vodislav's lecture!)
  - Attributes' name conversion (CNo -> CustomerNumber)
  - Domain knowledge use to detect redundant information
- Verify the data coherence:
  - Application-based constraints
  - Incoherence's' resolution
- «Completion»
  - Missing values

Data pre-processing is the tasks that takes a lot of time in the KDD process!

KDD: Pre-processing

- Numerical attributes discretization
  - Independently from the Data Mining task
    - E.g.: equally intervals
  - Related to the Data Mining task
    - E.g.: intervals which maximize the information gain
  - Generate additional attributes:
    - Aggregate a set of attributes
      - E.g. : from calls
      - Number of minutes per day, week, local call

KDD: Data Mining

- Definition [Fayad et al. 96]

  Data Mining is the application of afficient algorithms in order to identify the patterns in the data

- Data Mining methods:
  - Clustering
  - Classification
  - Frequent pattern mining
  - Linear regression
  - Outlier detection
  - Etc.

KDD: Data Mining

- Descriptive methods
  - Find human-interpretable patterns that describe the data
  - Example: Clustering
- Predictive methods
  - Use some variables to predict unknown or future values of other variables
  - Example: Recommender systems

KDD: Post-processing

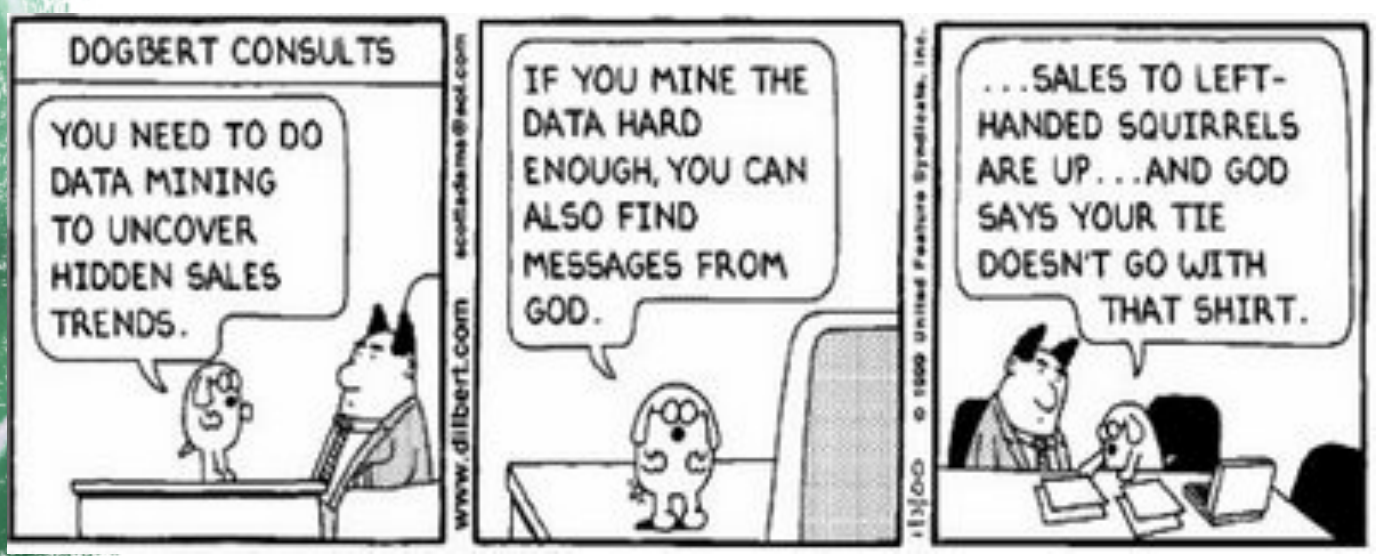
- Present the discovered patterns using a good visualization approach
- Evaluation of pattern by the expert
- If the evaluation is bad, launch a new mining task by changing:
  - The parameters
  - The mining methods
  - The data
- If the evaluation is positive:
  - Integrate the discovered knowledge in a knowledge base
  - Use this knowledge in future KDD process

Mining or not?

- Is NOT a Data Mining task…
  - Search for a phone number in a list
  - Make a Google search
- Is a Data Mining task:
  - Analyse the results of the queries that you did via Google

Meaningfulness of Analytic Answers

- A risk with "Data mining" is that an analyst can "discover" patterns that are meaningless
- Statisticians call it Bonferroni's principle:
  - Roughly, if you look in more places for interesting patterns than your amount of data will support, you are bound to find crap



Meaningfulness of Analytic Answers

Example:

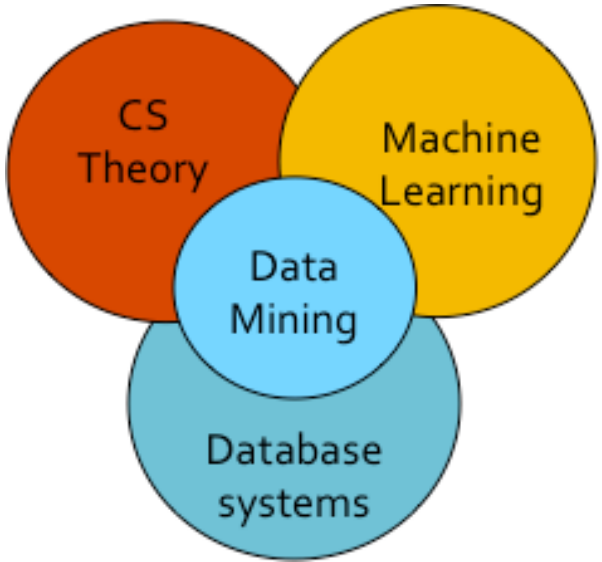
- We want to find (unrelated) people who at least twice have stayed at the same hotel on the same day
- 109 people being tracked
- 1,000 days
- Each person stays in a hotel 1% of time (1 day out of 100)
- Hotels hold 100 people (so 105 hotels)
- If everyone behaves randomly (i.e., no terrorists) will the data mining detect anything suspicious?

- Expected number of "suspicious" pairs of people:
  - If everyone behaves randomly (i.e., no terrorists) will the data mining detect anything suspicious?
  - 250,000
  - … too many combinations to check – we need to have some additional evidence to find "suspicious" pairs of people in some more efficient way

Data mining and other areas

- Data mining overlaps with:
  - **Databases**: Large-scale data, simple queries
  - **Machine learning**: Small data, Complex models
  - **Computer Science Theory**: (Randomized) Algorithms

- Different cultures:
  - To a DB person, data mining is an extreme form of **analytic processing** – queries that examine large amounts of data
    - Result is the query answer
  - To a ML person, data-mining is the **inference of models**
    - Result is the parameters of the model
  - DataMining does both!



Data mining algorithms

- Frequent pattern mining

Frequent pattern mining

Supermarket shelf management – Market-basket model:

- **Goal**: Identify items that are bought together by sufficiently many customers
- **Approach**: Process the sales data collected with barcode scanners to find dependencies among items
- A classic rule:
  - **If on Friday night a man buys diapers, then he is likely to buy beer too!**
  - Don't be surprised if you find beers next to diapers..

Frequent pattern mining

Market-basket model:

- A large set of items:
  - The products sold in
- **Approach**: Process the sales data collected with barcode scanners to find dependencies among items
- A classic rule:
  - **If on Friday night a man buys diapers, then he is likely to buy beer too!**
  - Don’t be surprised if you find beers next to diapers..

The Market-Basket Model

- A large set of **items**
  - *e.g., things sold in a supermarket*
- A large set of **baskets**
- Each basket is a small subset of items
  - *e.g., the things one customer buys on one day*
- Want to discover association rules
  - *People who bought {x,y,z} tend to buy {v,w}*
    - Amazon!

Input:

TID	*Items*
1	Bread, Coke, Milk
2	Beer, Bread
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Coke, Diaper, Milk

Output:

Rules Discovered:

{Milk} --> {Coke}
{Diaper, Milk} --> {Beer}

FI – Applications (1)

- **Items** = products; **Baskets** = sets of products someone bought in one trip to the store
- **Real market baskets**: Chain stores keep TBs of data about what customers buy together
  - *Tells how typical customers navigate stores, lets them position tempting items*
  - *Suggests tie-in "tricks", e.g., run sale on diapers and raise the price of beer*
  - *Need the rule to occur frequently, or no $$'s*
- **Amazon's people who bought X also bought Y**

FI – Application (2)

- Baskets = sentences; Items = documents containing those sentences
  - *Items that appear together too often could represent plagiarism*
  - *Notice items do not have to be "in" baskets*

- Baskets = patients; Items = drugs & side-effects
  - *Has been used to detect combinations of drugs that result in particular side-effects*
  - *But requires extension: Absence of an item needs to be observed as well as presence*

Frequent Itemsets

- Simplest question: Find sets of items that appear together "frequently" in baskets

- **Support for itemset I**: Number of baskets containing all items in I
  - *(Often expressed as a fraction of the total number of baskets)*

TID	*Items*
1	Bread, Coke, Milk
2	Beer, Bread
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Coke, Diaper, Milk

Support of {Beer, Bread} = 2

- Given a support threshold ***s***, then sets of items that appear in at least s baskets are called frequent itemsets

Frequent Itemsets: Example

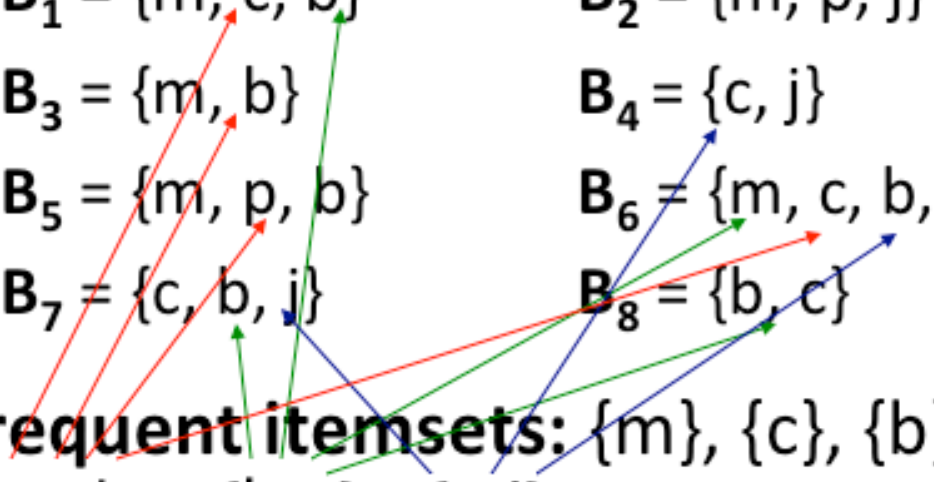
- **Items** = {milk, coke, pepsi, beer, juice}
- **Support threshold** = 3 baskets

B_1 = {m, c, b} B_2 = {m, p, j}

B_3 = {m, b} B_4 = {c, j}

B_5 = {m, p, b} B_6 = {m, c, b, j}

B_7 = {c, b, j} B_8 = {b, c}



- **Frequent itemsets:** {m}, {c}, {b}, {j}, {m,b} , {b,c} , {c,j}.

Frequent Itemsets: Computational Model

- Typically, data is kept in flat files rather than in a database system:
  - *Stored on disk*
  - *Stored basket-by-basket*
  - *Baskets are small but we have many baskets and many items*
    - Expand baskets into pairs, triples, etc. as you read baskets
    - Use k nested loops to generate all sets of size k

Note: We want to find frequent itemsets. To find them, we have to count them. To count them, we have to generate them.

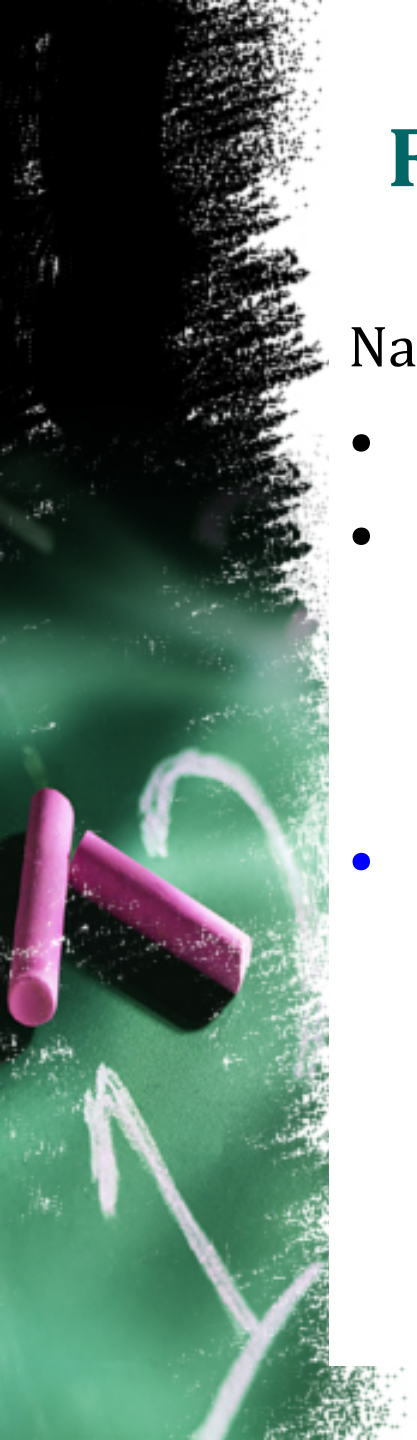
Frequent Itemsets: Computational Model

Main-Memory Bottleneck (imagine for Big Data!!!)

- For many frequent-itemset algorithms, main-memory is the critical resource
- As we read baskets, we need to count something, e.g., occurrences of pairs of items
- The number of different things we can count is **limited** by main memory
- Swapping counts in/out is a disaster

FI: Computational Model

- The hardest problem often turns out to be finding the frequent **pairs of items** {i1, i2}
  - *Why? Freq. pairs are common, freq. triples are rare*
    - Why? Probability of being frequent drops exponentially with size; number of sets grows more slowly with size
- Let's first concentrate on pairs, then extend to larger sets
- The approach:
  - *We always need to generate all the itemsets*
  - *But we would only like to count (keep track) of those itemsets that in the end turn out to be frequent*



FI: Computational Model

Naïve Algorithm

- Naïve approach to finding frequent pairs
- Read file once, counting in main memory the occurrences of each pair:
  - *From **each basket** of n items, generate its n(n-1)/2 pairs by two nested loops*
- **Fails** if (#items)2 exceeds main memory
  - *Remember: #items can be 100K (Wal-Mart) or 10B (Web pages)*
    - Suppose 105 items, counts are 4-byte integers
    - Number of pairs of items: 105(105-1)/2 = 5*109
    - Therefore, 2*1010 (**20 gigabytes**) of memory needed

FI: Computational Model

Naïve Algorithm

- Two approaches:
- Approach 1: Count all pairs using a matrix
- Approach 2: Keep a table of triples [i, j, c] = "the count of the pair of items {i, j} is c."
  - *If integers and item ids are 4 bytes, we need approximately 12 bytes for pairs with count > 0*
  - *Plus some additional overhead for the hashtable*
- Note:
  - *Approach 1 only requires 4 bytes per pair*
  - *Approach 2 uses 12 bytes per pair (but only for pairs with count > 0)*

FI: Computational Model

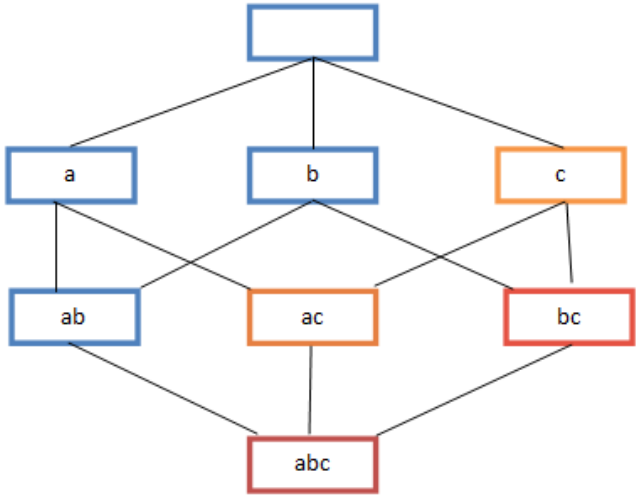
Naïve Algorithm

- Two approaches:
- Apr
- Ap
  of
  - ➢
  - ➢
- No
  - ➢
  - ➢ *Approach 2 uses 12 bytes per pair (but only for pairs with count > 0)*

Problem is if we have too many items so the pairs do not fit into memory.

Can we do better?

FI: Apriori Algorithm (1)



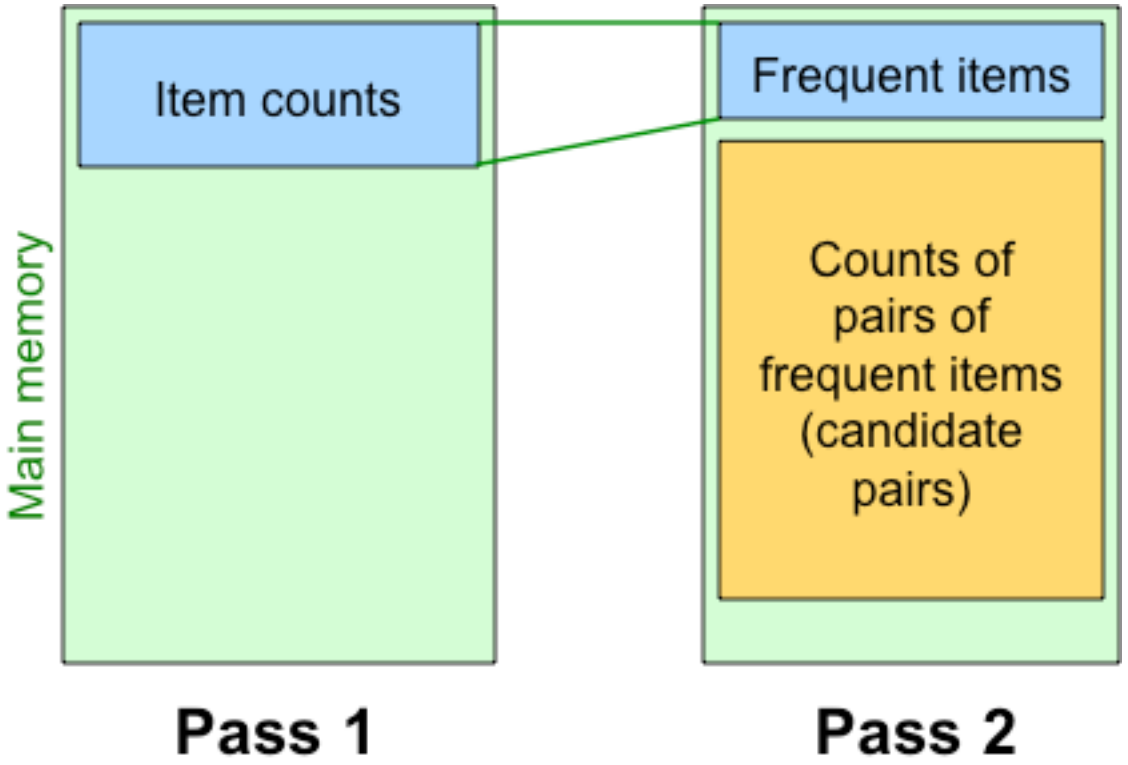
- A two-pass approach called APriori **limits the need for main memory**
- Key idea: monotonicity
  - *If a set of items I appears at least s times, so does every subset J of I*
- Contrapositive for pairs:
  If item *i* does not appear in *s* baskets, then no pair including *i* can appear in *s* baskets
- So, how does APriori find freq. pairs?

FI: Apriori Algorithm (2)

- **Pass 1:** Read baskets and count in main memory the occurrences of each **individual item**
  - Requires only memory proportional to #items
- **Items that appear times are the frequent items**
- **Pass 2:** Read baskets again and count in main memory only those pairs where both elements are frequent (from Pass 1)
  - *Requires memory proportional to square of **frequent** items only (for counts)*
  - *Plus a list of the frequent items (so you know what must be counted)*

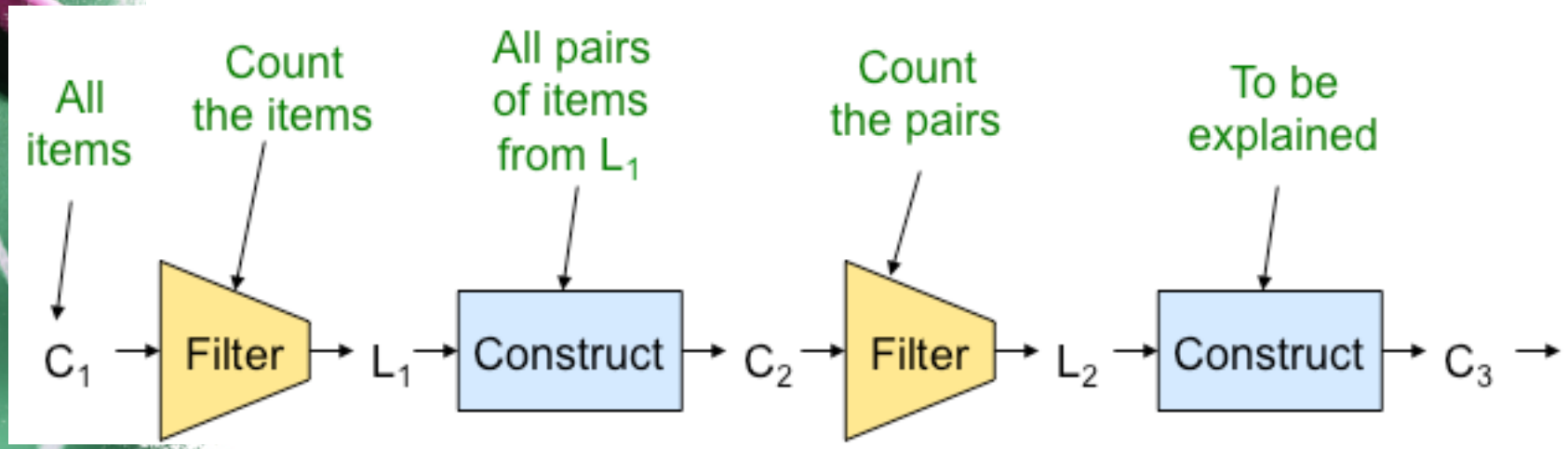
FI: Apriori Algorithm (1)

- Main-Memory



FI: Apriori Algorithm (1)

- For each k, we construct **two sets of k-tuples** (sets of size k):
  - ***Ck*** *= candidate k-tuples = those that might be frequent sets (support > s) based on information from the pass for k–1*
  - ***Lk*** *= the set of truly frequent k-tuples*



FI: Apriori Algorithm (1)

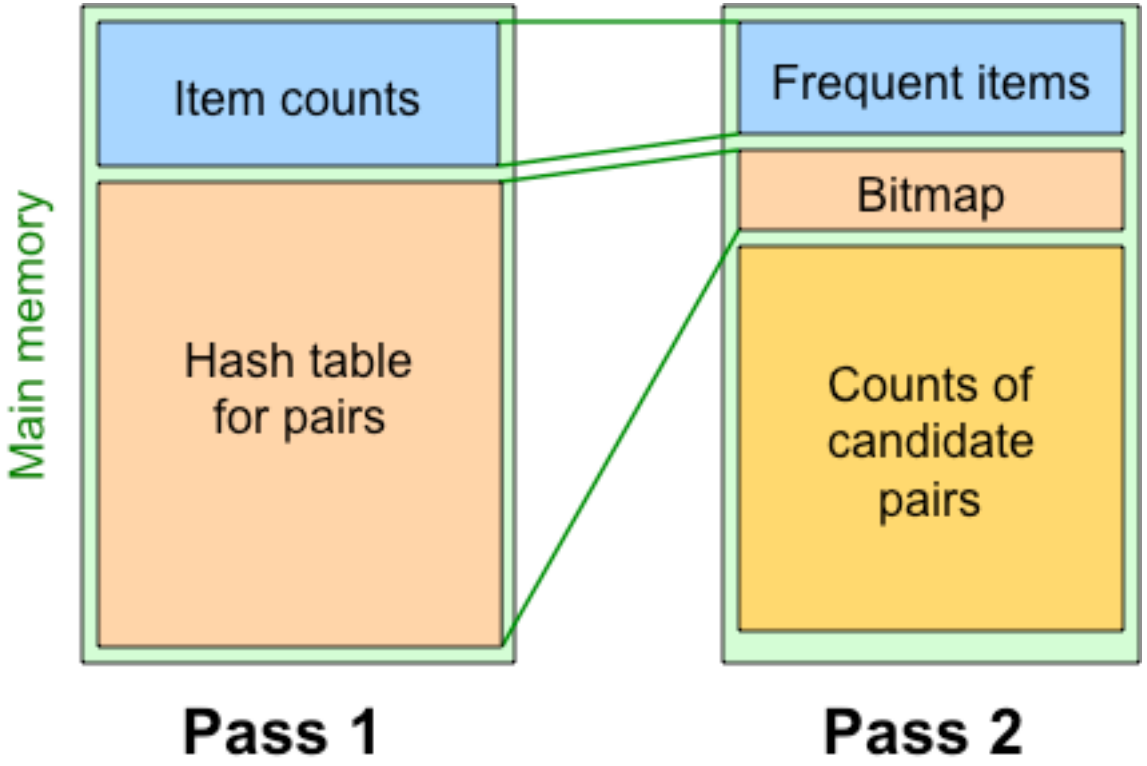
- BigData?
  - *One pass **for each** k (itemset size)*
  - *Needs room in main memory to **count each candidate** k–tuple*
  - *For typical market-basket data and reasonable support (e.g., 1%), **k = 2 requires the most memory***

FI: PCY (Park-Chen-Yu) Algorithm

- Observation:
  In pass 1 of APriori, most memory is idle
  - ➢ *We store only individual item counts*
  - ➢ *Can we use the idle memory to reduce memory required in pass 2?*

- Pass 1 of PCY: In addition to item counts, maintain a hash table with as many buckets as fit in memory
  - ➢ *Keep a count for each bucket into which pairs of items are hashed*
    - For each bucket just keep the count, not the actual pairs that hash to the bucket!

FI: PCY (Park-Chen-Yu) Algorithm



FI: in <=2 passes?

- A-Priori, PCY, etc., take **k passes** to find frequent itemsets of size k
- Can we use **fewer passes**?
- Use 2 or fewer passes for all sizes, but may miss some frequent itemsets
  - *Random sampling*
  - *SON (Savasere, Omiecinski, and Navathe)*
  - *Toivonen*

FI: in <=2 passes?

Sampling:

- Take a random sample of the market baskets

- Run Apriori or one of its improvements in main memory
  - *So we don't pay for disk I/O each time we increase the size of itemsets*
  - *Reduce support threshold proportionally to match the sample size*

FI: in <=2 passes?

Sampling:

- Optionally, verify that the candidate pairs are truly frequent in the entire data set by a second pass (avoid false positives)

- But you don't catch sets frequent in the whole but not in the sample
  - *Smaller threshold, e.g., s/125, helps catch more truly frequent itemsets*
  - *But requires more space*

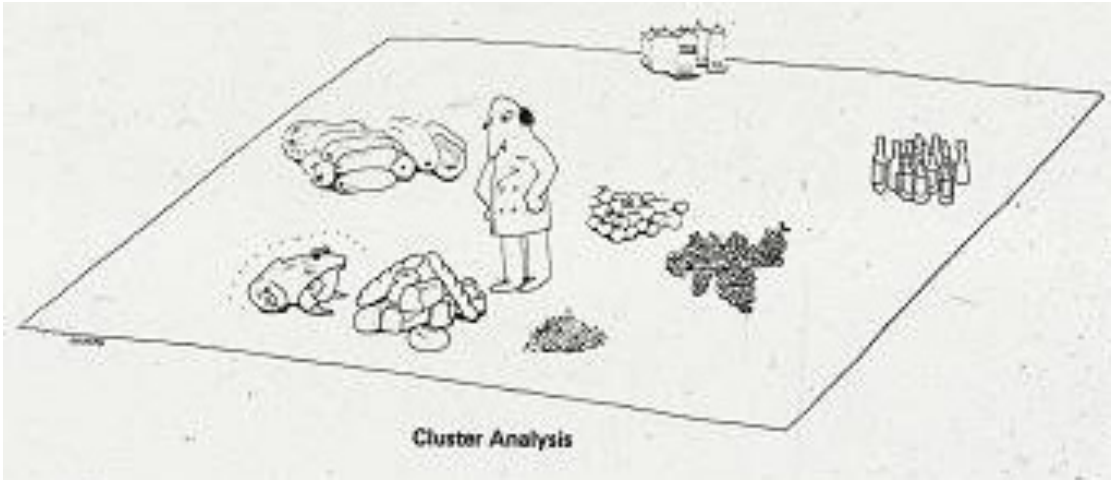
FI: in <=2 passes?

In MapReduce platform:

- Phase 1: Find candidate itemsets
  - *Map?*
  - *Reduce?*
- Phase 2: Find true frequent itemsets
  - *Map?*
  - *Reduce?*

Data mining algorithms

- Clustering



Clustering: the problem

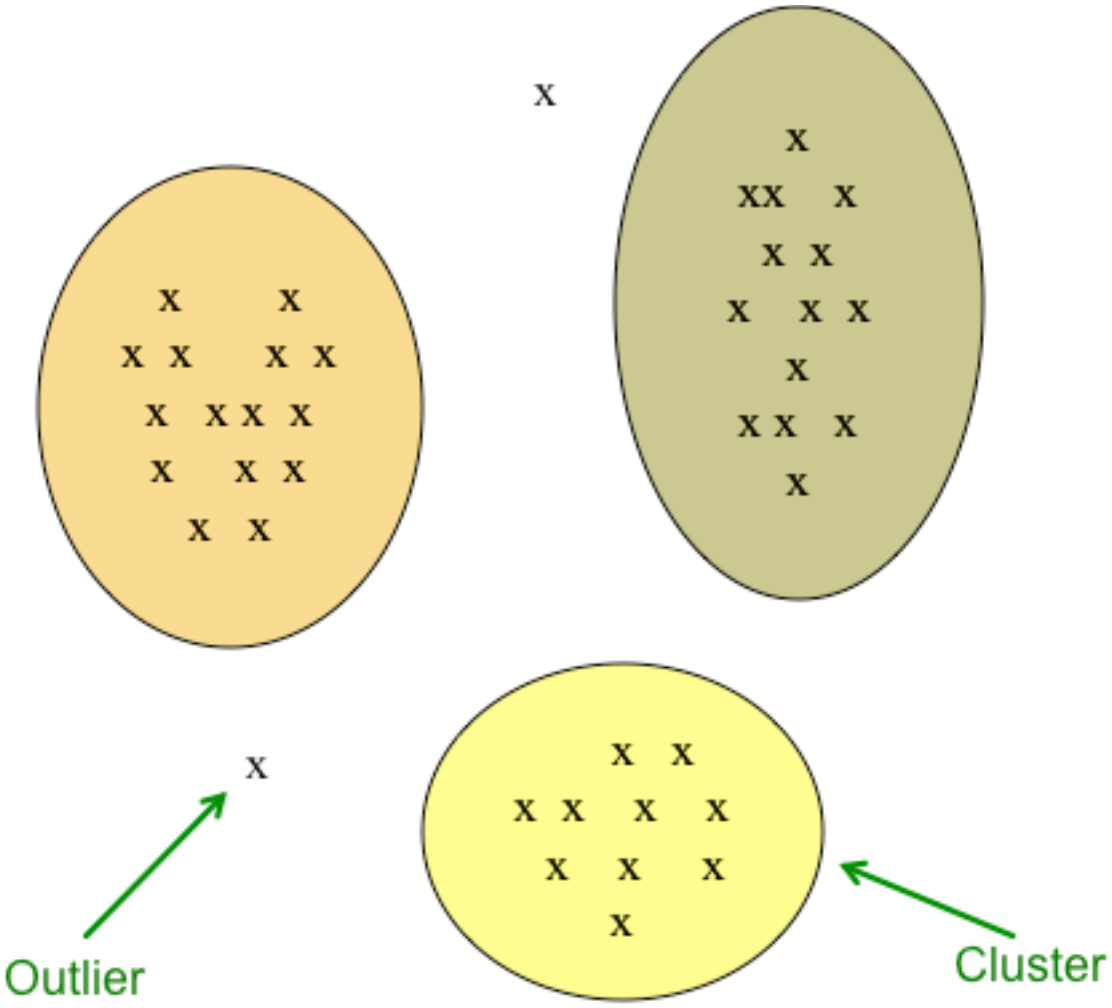
- Given a cloud of data points we want to understand its structure

Clustering: the problem

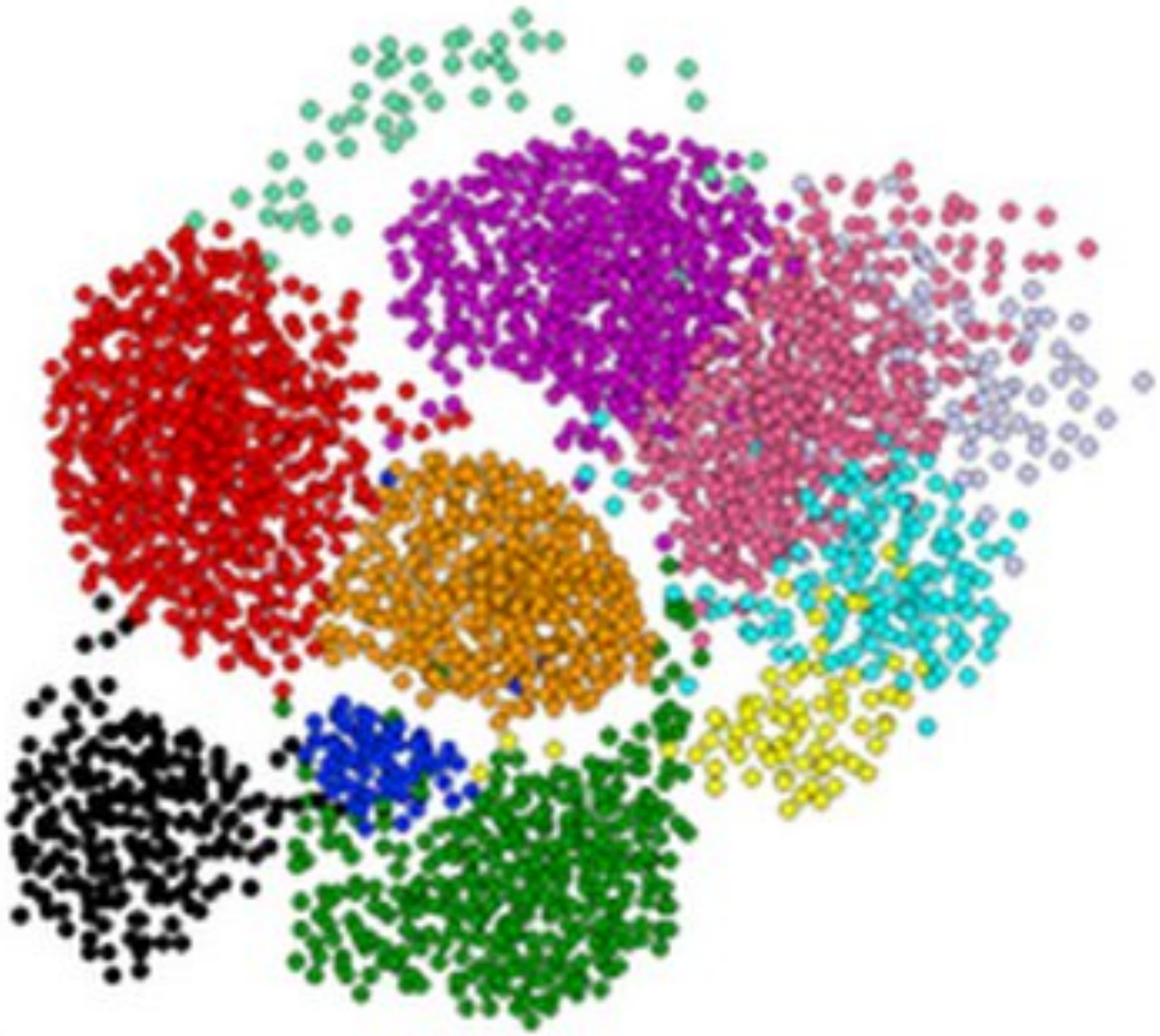
- Given a **set of points**, with a notion of **distance** between points, group the points into some number of **clusters**, so that
  - *Members of a cluster are close/similar to each other*
  - *Members of different clusters are dissimilar*

- Usually:
  - *Points are in a **high-dimensional space***
  - *Similarity is defined using a **distance measure***
    - Euclidean, Cosine, Jaccard, edit distance, …

Clustering: outliers & clusters



Clustering: a hard problem!



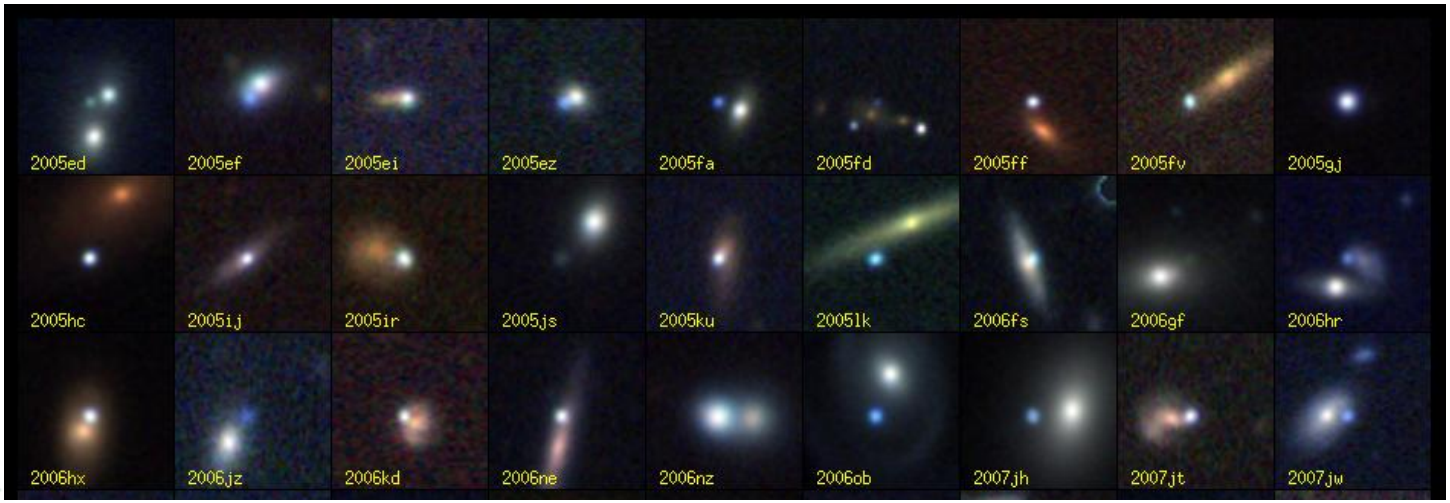
Clustering: a hard problem! Why?

- Clustering in two dimensions looks easy
- Clustering small amounts of data looks easy
- And in most cases, looks are not deceiving

- Many applications involve not 2, but 10 or 10,000 dimensions
- High-dimensional spaces look different: Almost all pairs of points are at about the same distance

Clustering e.g.1: how to cluster Galaxies

- A catalog of 2 billion “sky objects” represents objects by their radiation in 7 dimensions (frequency bands)
- Problem: Cluster into similar objects, e.g., galaxies, nearby stars, quasars, etc.
- Sloan Digital Sky Survey



Clustering e.g.2: how to cluster Music CDs

- Intuitively: Music divides into categories, and customers prefer a few categories
- But what are categories really?

- Represent a CD by a set of customers who bought it

- Similar CDs have similar sets of customers, and vice-versa

Clustering PB2: how to cluster Music CDs

Space of all CDs:

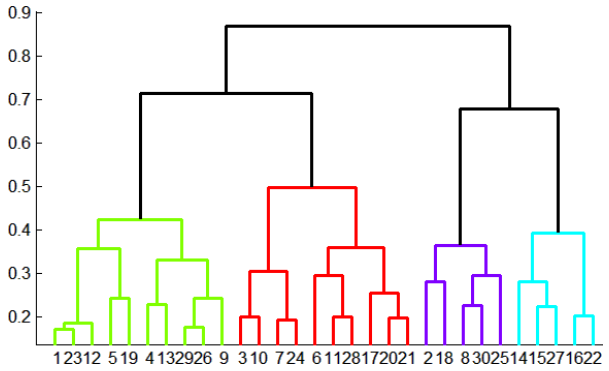
- Think of a space with one dim. for each customer
  - *Values in a dimension may be 0 or 1 only*
  - *A CD is a point in this space (x1, x2,…, xk), where xi = 1 iff the i th customer bought the CD*
- For Amazon, the dimension is **tens of millions**
- **Task**: Find clusters of similar CDs

Clustering: similarity measure

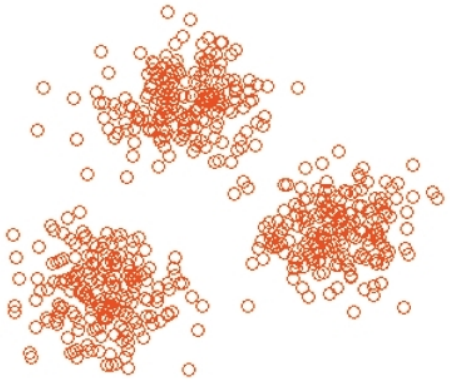
- Sets as vectors: Measure similarity by the cosine distance
- Sets as sets: Measure similarity by the Jaccard distance
- Sets as points: Measure similarity by Euclidean distance

Clustering: methods

- Hierarchical:
  - ➢ *Agglomerative (bottom up):*
    - Initially, each point is a cluster
    - Repeatedly combine the two "nearest" clusters into one
  - ➢ *Divisive (top down):*
    - Start with one cluster and recursively split it



- Point assignment:
  - ➢ *Maintain a set of clusters*
  - ➢ *Points belong to "nearest" cluster*



Clustering: k-means Algorithm(s)

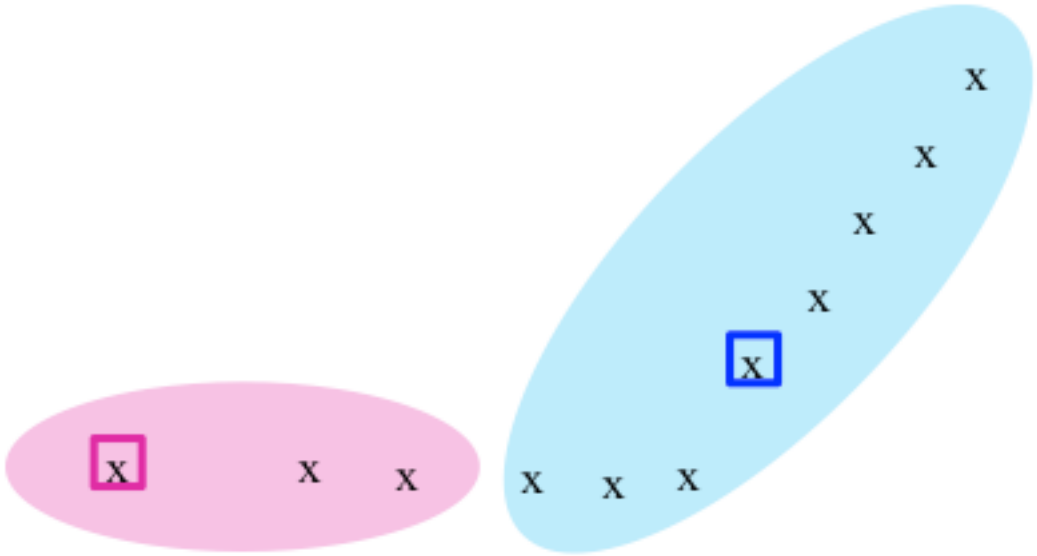
- Assumes Euclidean space/distance
- Start by picking k, the number of clusters
- Initialize clusters by picking one point per cluster
- Example: Pick one point at random, then k-1 other points, each as far away as possible from the previous points

Clustering: k-means Algorithms

Populating Clusters

- 1) For each point, place it in the cluster whose current centroid it is nearest
- 2) After all points are assigned, update the locations of centroids of the k clusters
- 3) Reassign all points to their closest centroid
  - *Sometimes moves points between clusters*
- Repeat 2 and 3 until convergence
  - *Convergence: Points don't move between clusters and centroids stabilize*

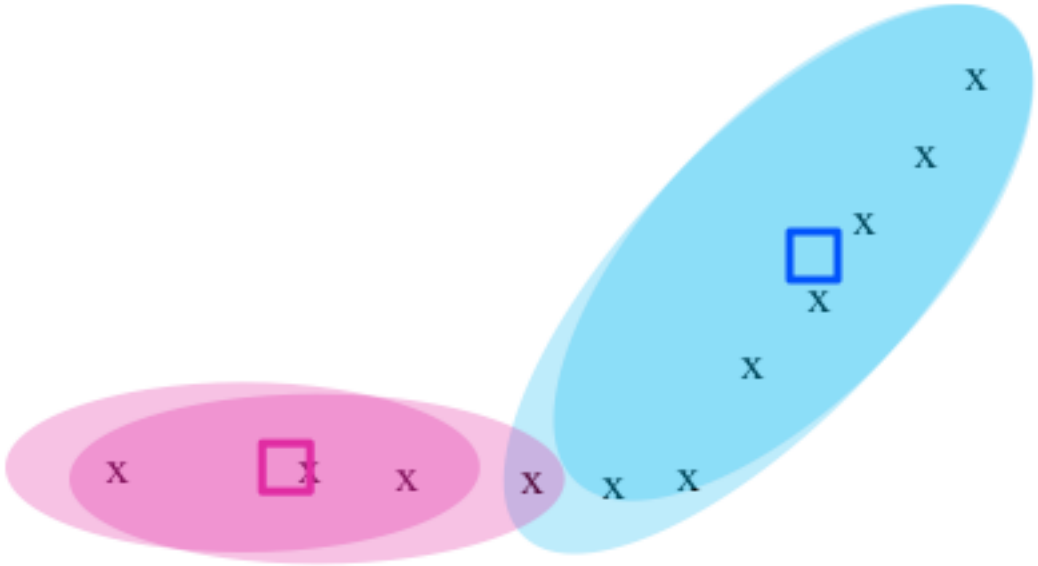
Clustering: k-means Algorithms – Example



x … data point
□ … centroid

Clusters after round 1

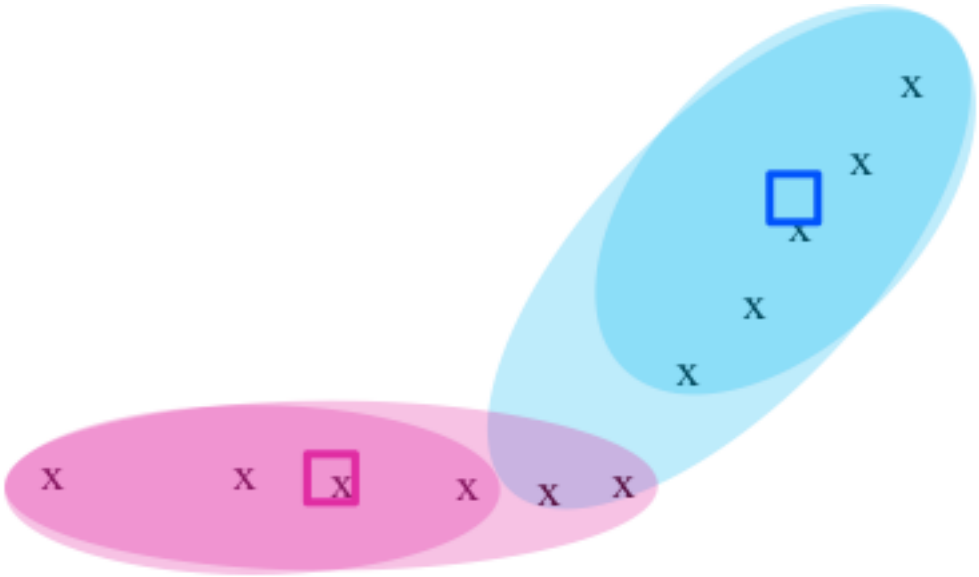
Clustering: k-means Algorithms – Example



x … data point
□ … centroid

Clusters after round 2

Clustering: k-means Algorithms – Example



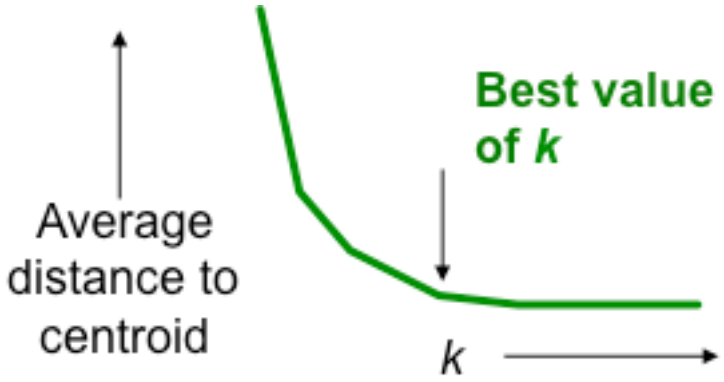
x … data point
□ … centroid

Clusters at the end

Clustering: k-means Algorithms

How to select k?

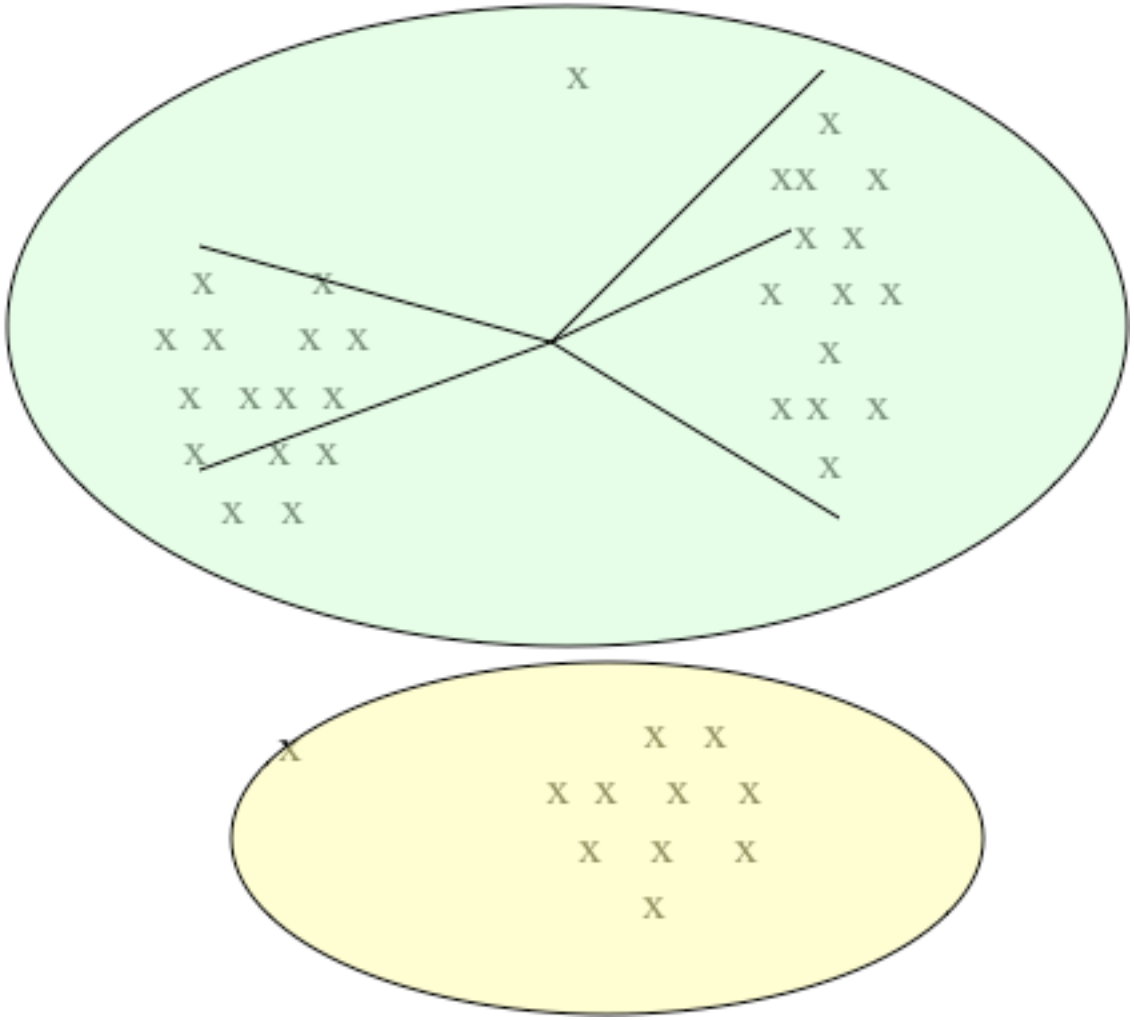
- Try different k, looking at the change in the average distance to centroid as k increases
- Average falls rapidly until right k, then changes little



Clustering: k-means Algorithms

How to select k?

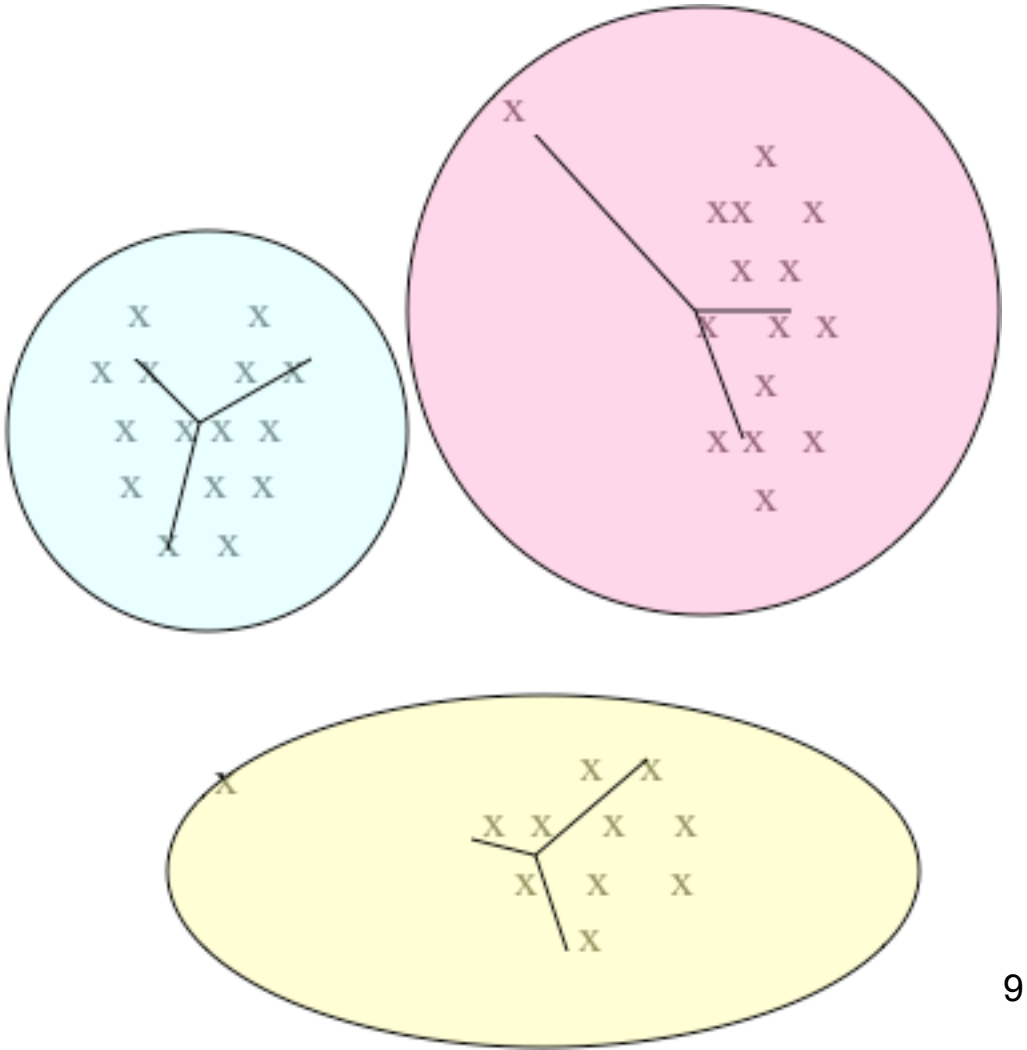
Too few;
many long
distances
to centroid.



Clustering: k-means Algorithms

How to select k?

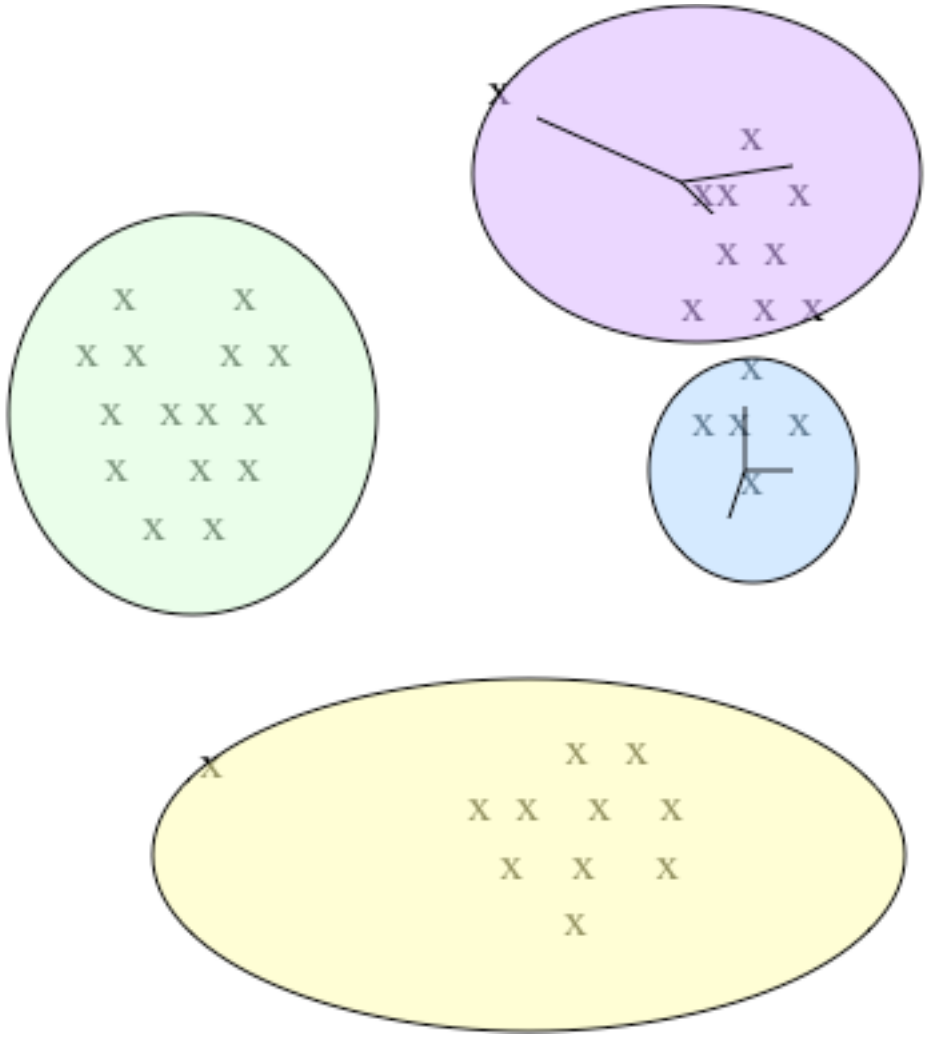
Just right;
distances
rather short.



Clustering: k-means Algorithms

How to select k?

Too many;
little improvement
in average
distance.



Clustering: what type of algorithms for bigdata?

Challenge: how to work on high volume and high dimensionality?

- Single-machine clustering
  - *Sample based techniques*
  - *Dimension reduction techniques*

- Multiple-machine clustering
  - *Parallel clustering*
  - *MapReduce based clustering*

Clustering: k-means Algorithms for bigdata

BFR Algorithm

- is a variant of k-means designed to handle very large (disk-resident) data sets

- Assumes that clusters are normally distributed around a centroid in a Euclidean space
  - *Standard deviations in different dimensions may vary*
- Efficient way to summarize clusters (want memory required O(clusters) and not O(data))

Data mining algorithms for bigdata

Other challenges:

- Mining Sparse Data
- Mining Uncertain Data
- Mining Incomplete Data
- Mining Complex and Dynamic Data

References

These slides were taken/inspired by:

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