Analysis of Massive Data Sets

http://www.fer.hr/predmet/avsp

Prof. dr. sc. Siniša Srbljić

Doc. dr. sc. Dejan Škvorc

Doc. dr. sc. Ante Đerek

Faculty of Electrical Engineering and Computing
Consumer Computing Laboratory

Clustering

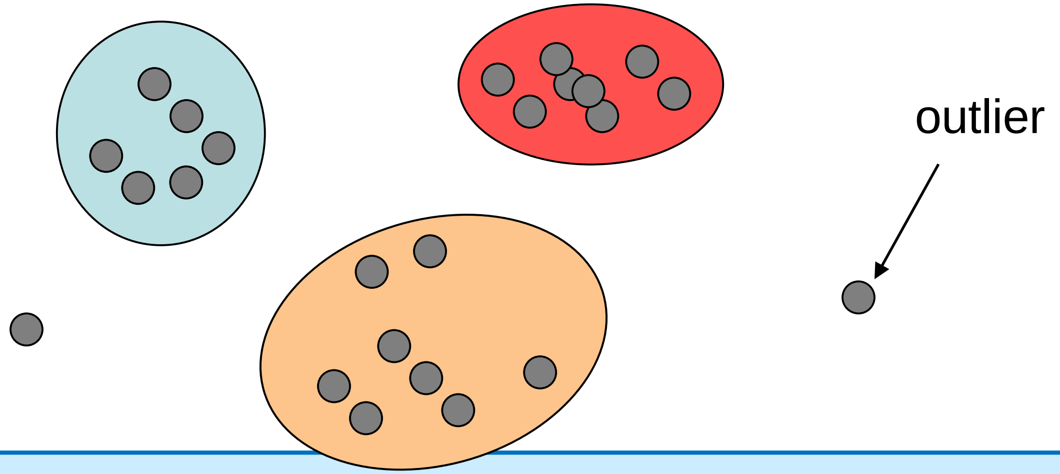
Goran Delač, PhD

Outline

- **Problem of clustering**
  - Cluster representation
  - Traditional approaches
- **BFR**
- **CURE**

The problem of clustering

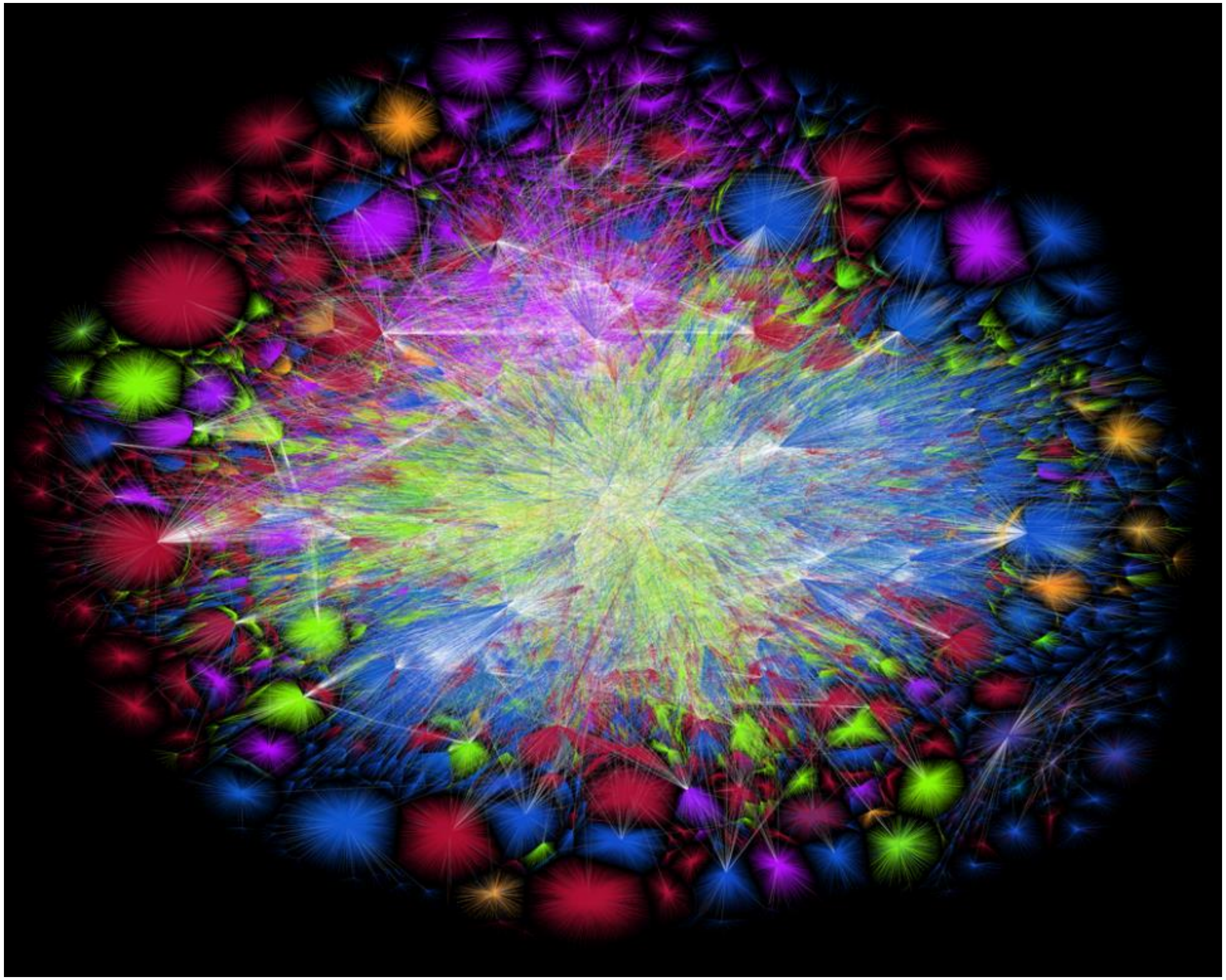
- **Why clustering?**
  - To get a better understanding of the data set
  - Given a set of data points, we would like to understand their **structure** – to see which points are **most similar** when compared to other points in the data set



The problem of clustering

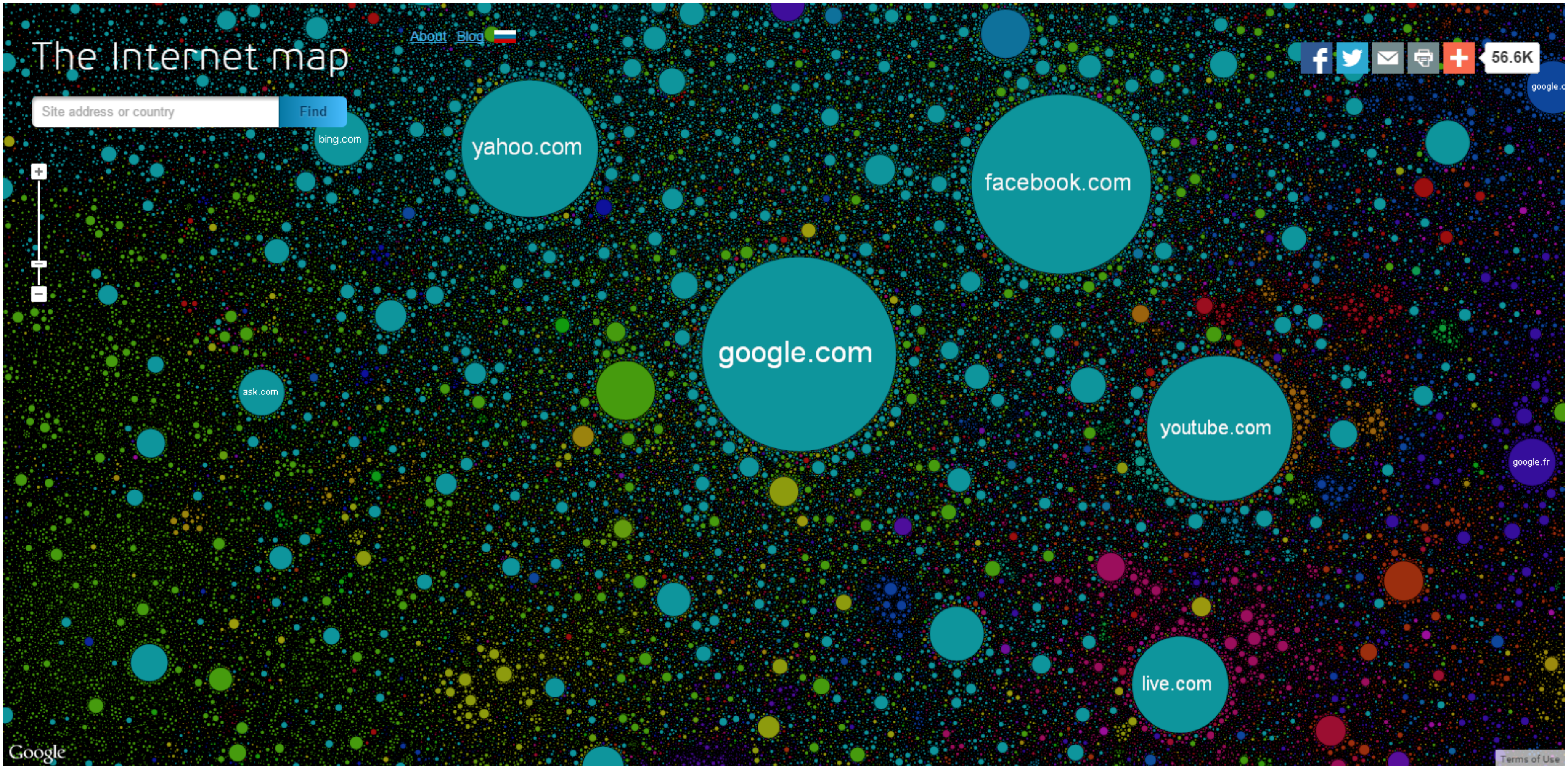
- **What is clustering?**
  - A set of points is **grouped** into clusters with a **notion of distance** between the points
    - Goal: group together points that are close to each other (**similar!**)
    - Points in different clusters are distant / not similar
  - Distance
    - Euclidian, Cosine distance, Manhattan distance, Jaccard similarity, Edit distance, Hamming distance, … [2]

The problem of clustering



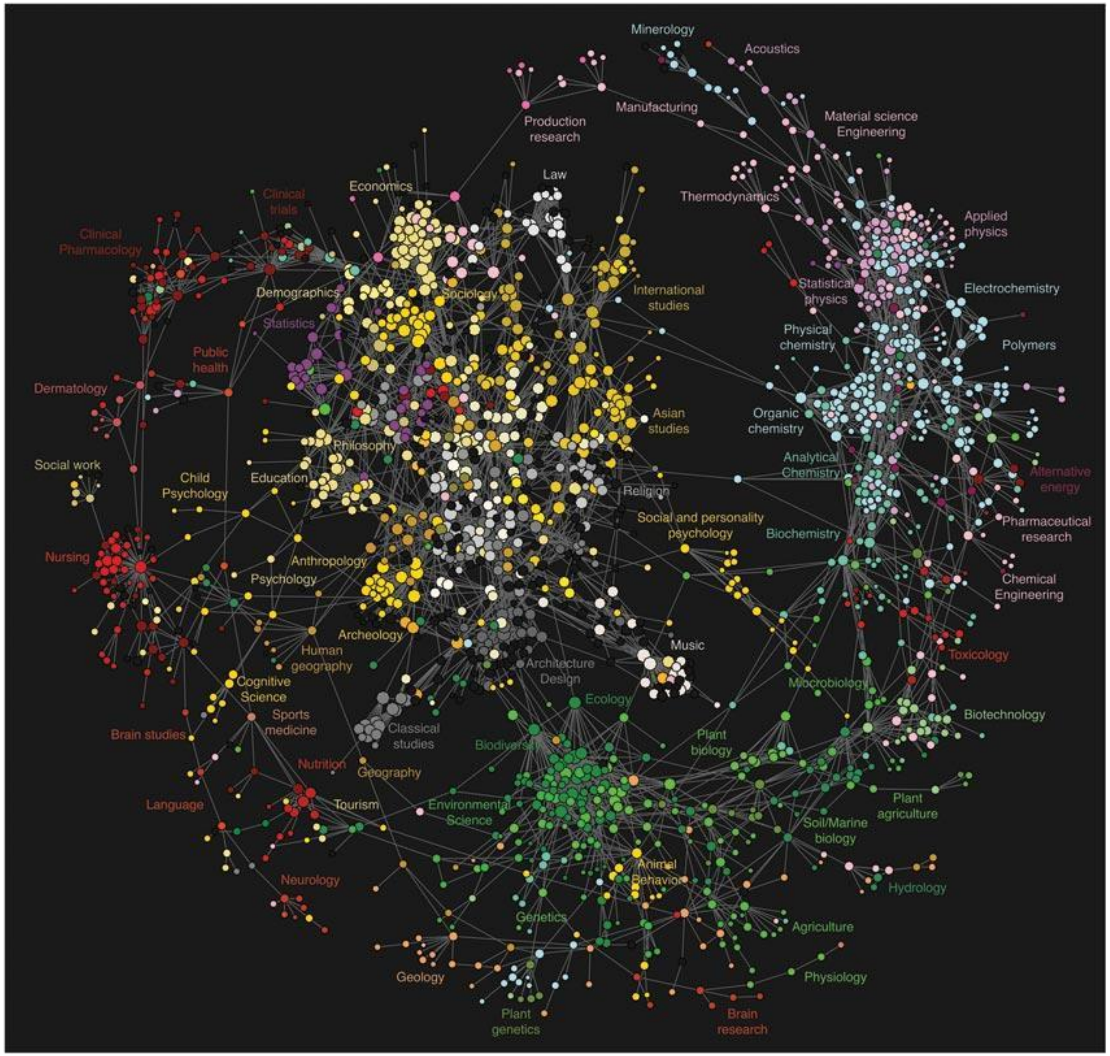
The Opte project, http://www.opte.org/the-internet/

The problem of clustering



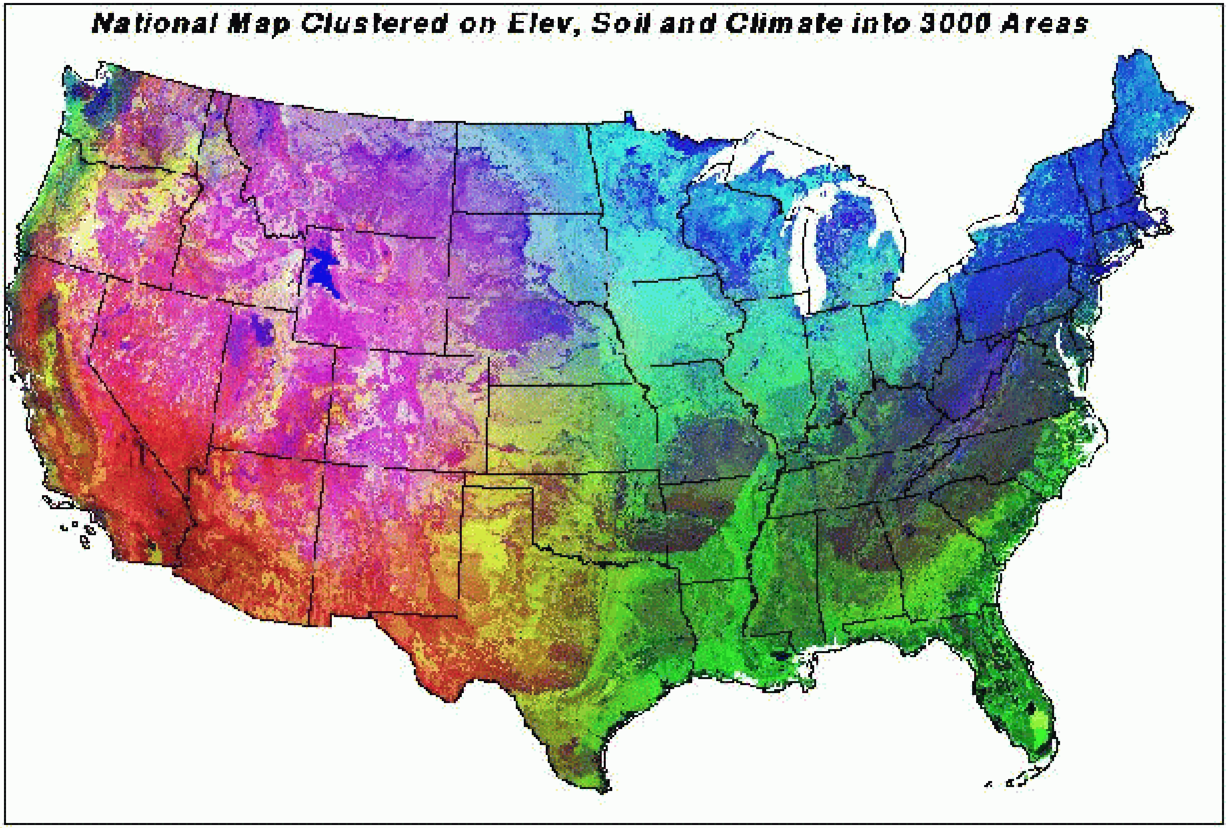
The Internet Map, http://internet-map.net/

The problem of clustering



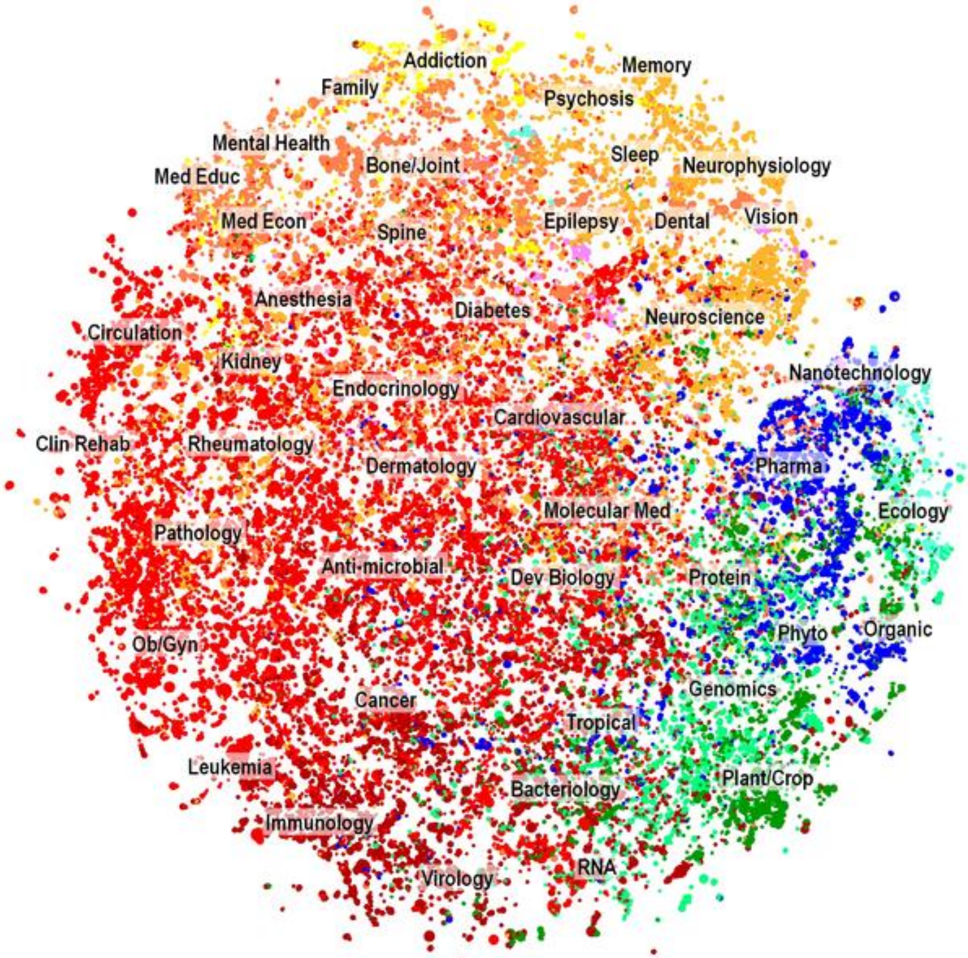
Web usage data outline map of knowledge, http://www.nature.com/news/2009/090309/full/458135a.html

The problem of clustering



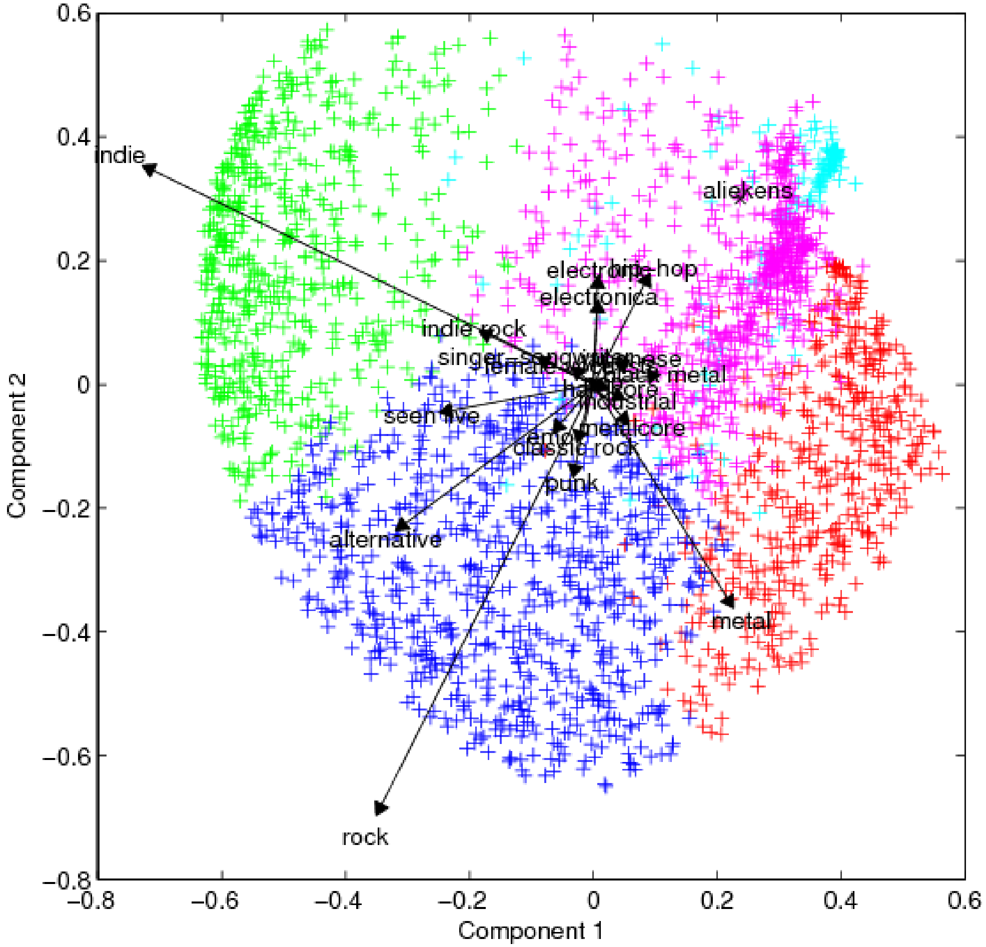
National Map of Vegetation Ecoregions Produced Empirically Using Multivariate Spatial Clustering, http://www.geobabble.org/~hnw/esri98/

The problem of clustering



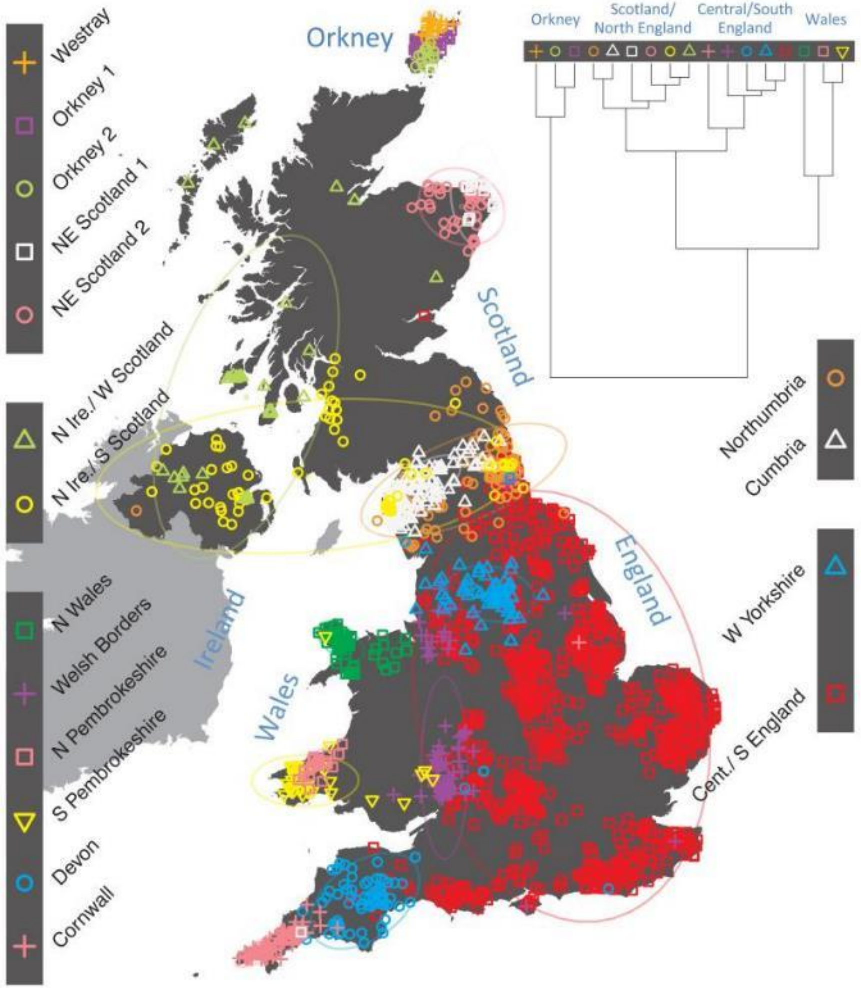
Clustering Map of Biomedical Articles (text similarity),
http://www.arbesman.net/blog/2011/03/24/clustering-map-of-biomedical-articles/

The problem of clustering



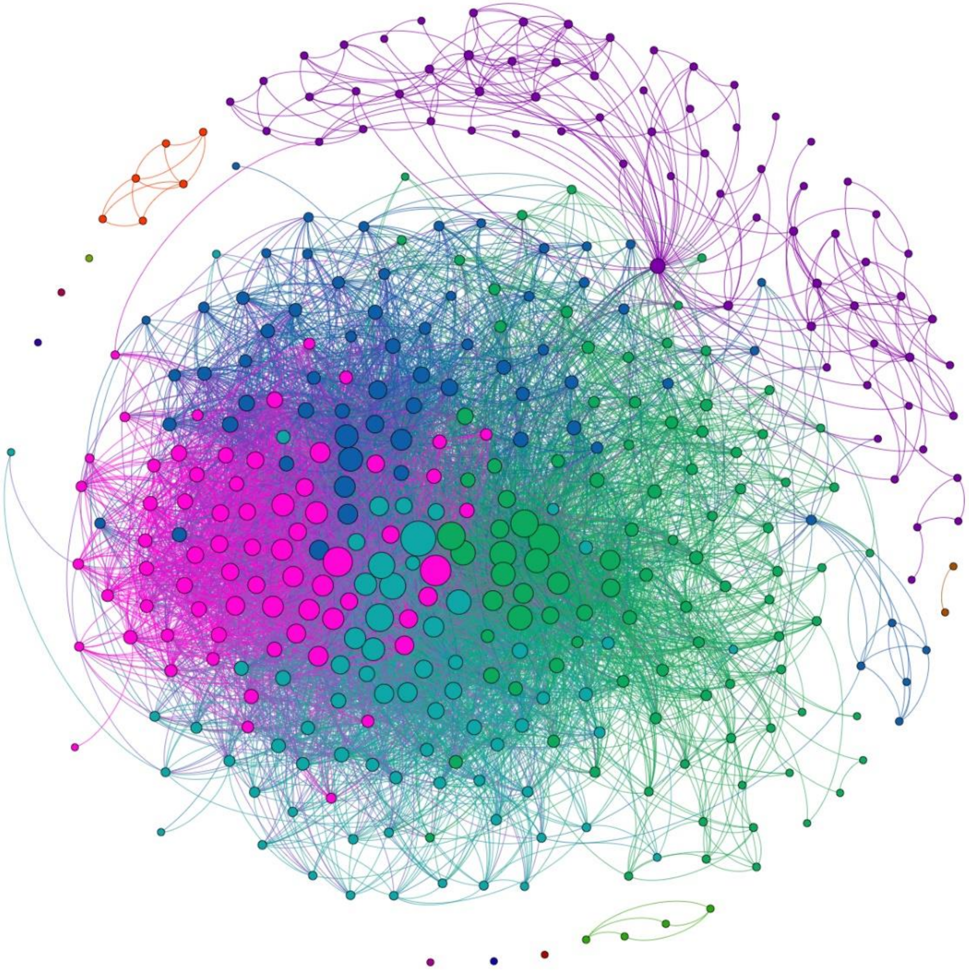
Music profiles (Last.fm),
http://anthony.liekens.net/index.php/Computers/DataMining

The problem of clustering



Clustering of the 2,039 UK individuals into 17 clusters based only on genetic data, http://www.ncbi.nlm.nih.gov/pmc/articles/PMC4632200/

The problem of clustering



Social network graphs,
http://social-dynamics.org/tag/clustering-algorithm/

The problem of clustering

- **Hard problem!**
  - High-dimensional data
  - A large amount of data points (possibly does not fit into main memory)

- **Curse of high dimensionality**
  - In a high-dimensional space almost all pairs of data points are equally distant from each other

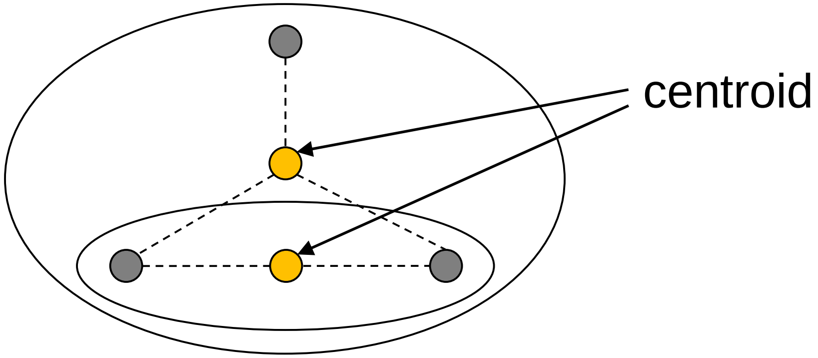
Cluster representation

- **How to represent multiple points?**
  - In **Euclidian** space, clusters can be represented by **centroids**

$$c_i = \frac{\sum_{j=1}^{N} x_{i,j}}{N}$$

$1 \le i \le D$ dimension

$1 \le j \le N$ point index



Cluster representation

- **How to represent multiple points?**
  - In **Euclidian** space, clusters can be represented by **centroids**

Distance to cluster = distance to centroid

Cluster representation

- **Non-Euclidian spaces**
    - Not possible to compute centroid

1. **Clusteroid** – actual representative point is chosen (a point from the data set)
    - Treat clusteroid as a cluster with an appropriate notion of proximity (Jaccard distance, Edit distance, …)

How to pick a clusteroid?

- Point that is ***"closest"*** to all other points in cluster
    - Smallest maximum distance to other points
    - Smallest average distance to other points
    - Smallest sum of squares of distances to other points

Cluster representation

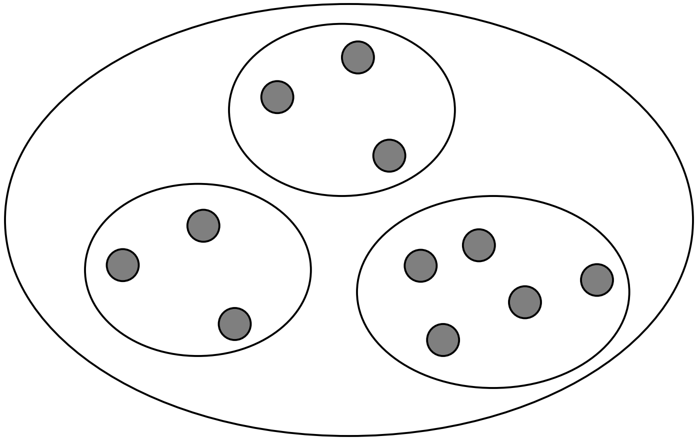
2. Cluster Cohesion

- **Diameter** of a merged cluster
  - Maximum distance between two points
- **Average distance**
  - Between all the points in the cluster
- **Density-based clustering**
  - Number of points in a measure of volume
  - Number of points divided by diameter or avg. distance

Hierarchical clustering

- Agglomerative (bottom up)
  - In the beginning each point is its own cluster
  - Iteratively merge closest clusters
- Divisive (top down)
  - Start with a single cluster
  - Split it recursively

$O(n^3)$



Careful implementation

$O(n^2 \log(n))$

Point assignment methods

- **Maintain a set clusters**
  - Known number of clusters
- **Points are assigned to their nearest cluster**
- **K-means clustering**
- **Better suited for large data sets**

K-means clustering

- **Euclidian space**

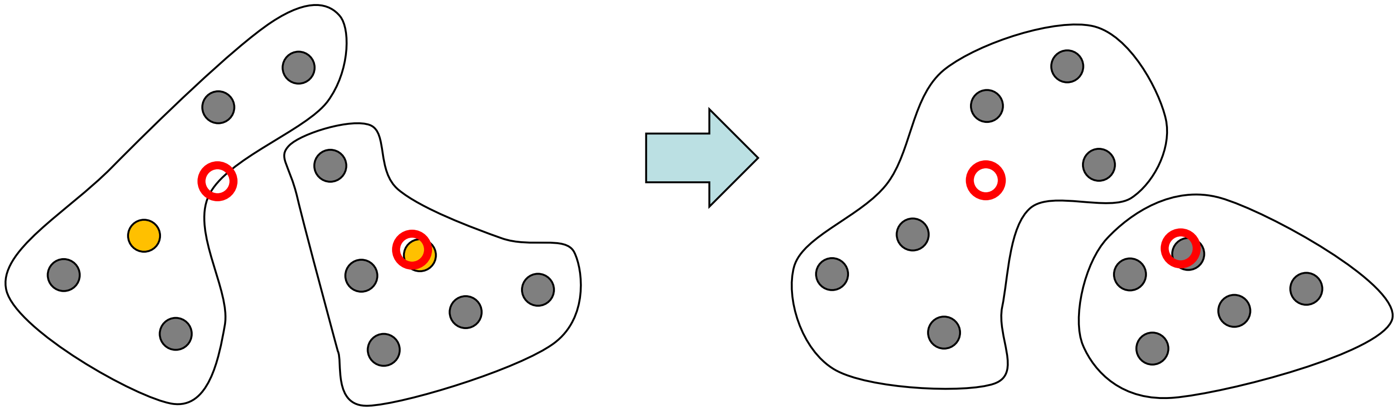
1. Initialization
   - Choose **k** points to be the initial centroids
   - Assign points to their nearest centroid

2. Recompute centroids' locations

3. Reassign points to new centroids
   - Points can move across cluster boundaries

K-means clustering

- Steps 2-3 are repeated until convergence
  - Points do not change clusters
  - OR some other stopping condition is met



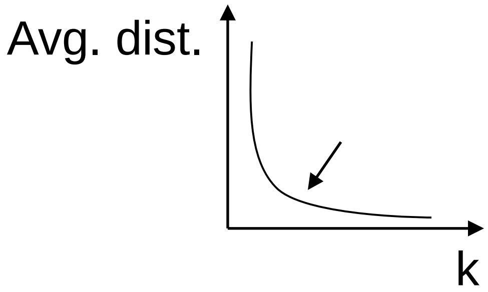
K-means clustering

- How to choose k?

- How to choose k points from the data set?

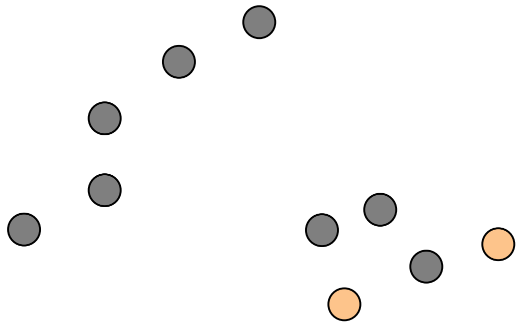
K-means clustering

- **Choosing k**
  - Intuitively – if we know the number of clusters in advance or we want to cluster the data in a specific number of clusters
  - Trying different values and observing cluster quality
    - E.g. increasing k will decrease average distance to the centroid within a cluster



K-means clustering

- **Choosing k points**
  - At random? Not a good idea



K-means clustering

- **Choosing k points**
  - Sampling
    - Get a representative sample of the data set
    - Cluster it using hierarchical clustering to get **k** clusters
    - Choose **k** points closest to the centroids
  - Disperse points
    - Pick first point at random
    - Choose the next point to be the most distant one to the selected points (the one with the greatest minimal distance)
    - Repeat the process until k points are selected

K-means clustering

- **Computational complexity**

$$O(rkN)$$

k – number of centroids
N – number of points
r – number of rounds

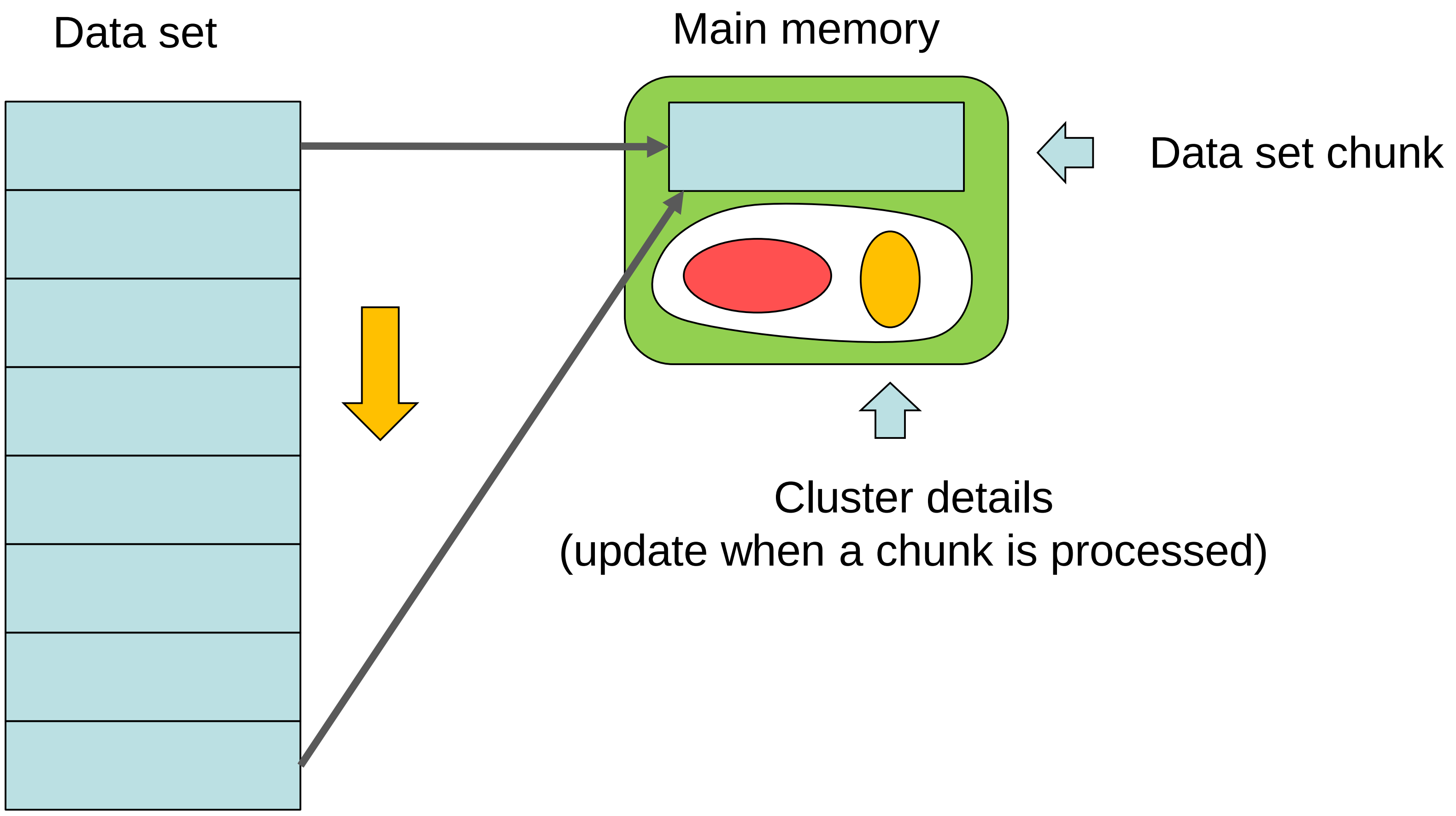
  - Number of rounds (r) can get large
  - N can be really large in massive data sets

  - Can we get meaningful clustering results in just one pass?

BFR

- **Bradley, Fayyad, Reina**
- **Based on k-means clustering approach**
- **Idea**
  - Avoid storing information on all points as they do not fit into main memory
  - Keep track of summarized cluster information

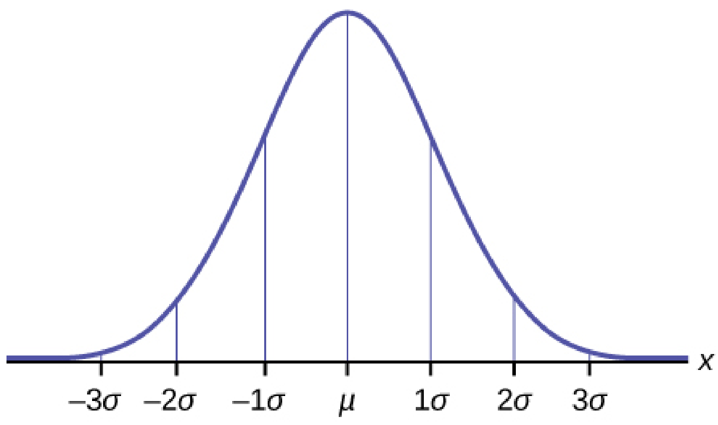
BFR

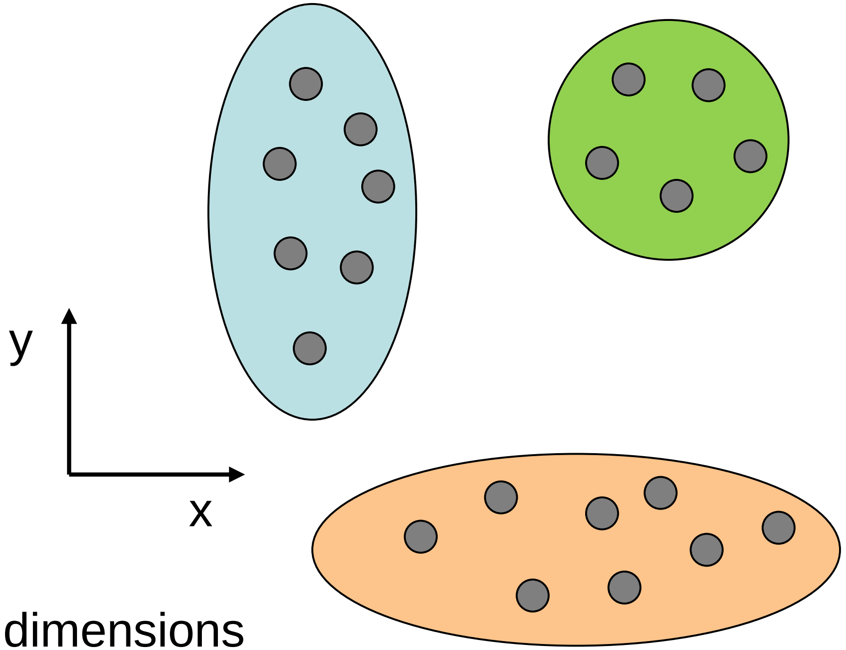


O(clusters) instead of **O(data)**

BFR

- **Assumptions**
  - Cluster members are normally distributed around the centroid
  - Euclidian space

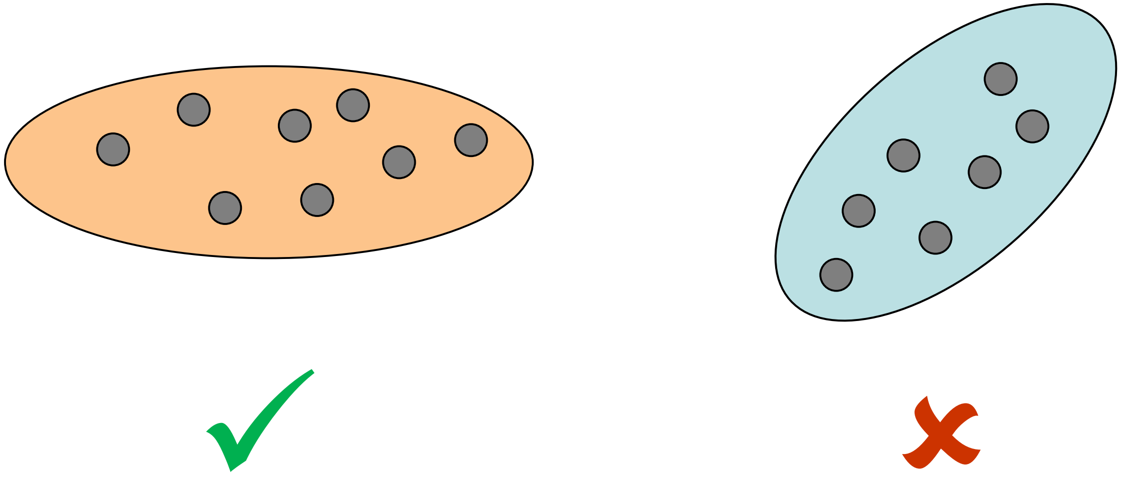




Standard deviations can vary for different dimensions

BFR

- **Assumptions**
  - Clusters are axis-aligned ellipses



BFR

□ Algorithm

1. Choose initial centroids
   - Some appropriate method, like in k-means clustering

2. Process data set
   - Read data set chunk at a time
   - Maintain cluster information

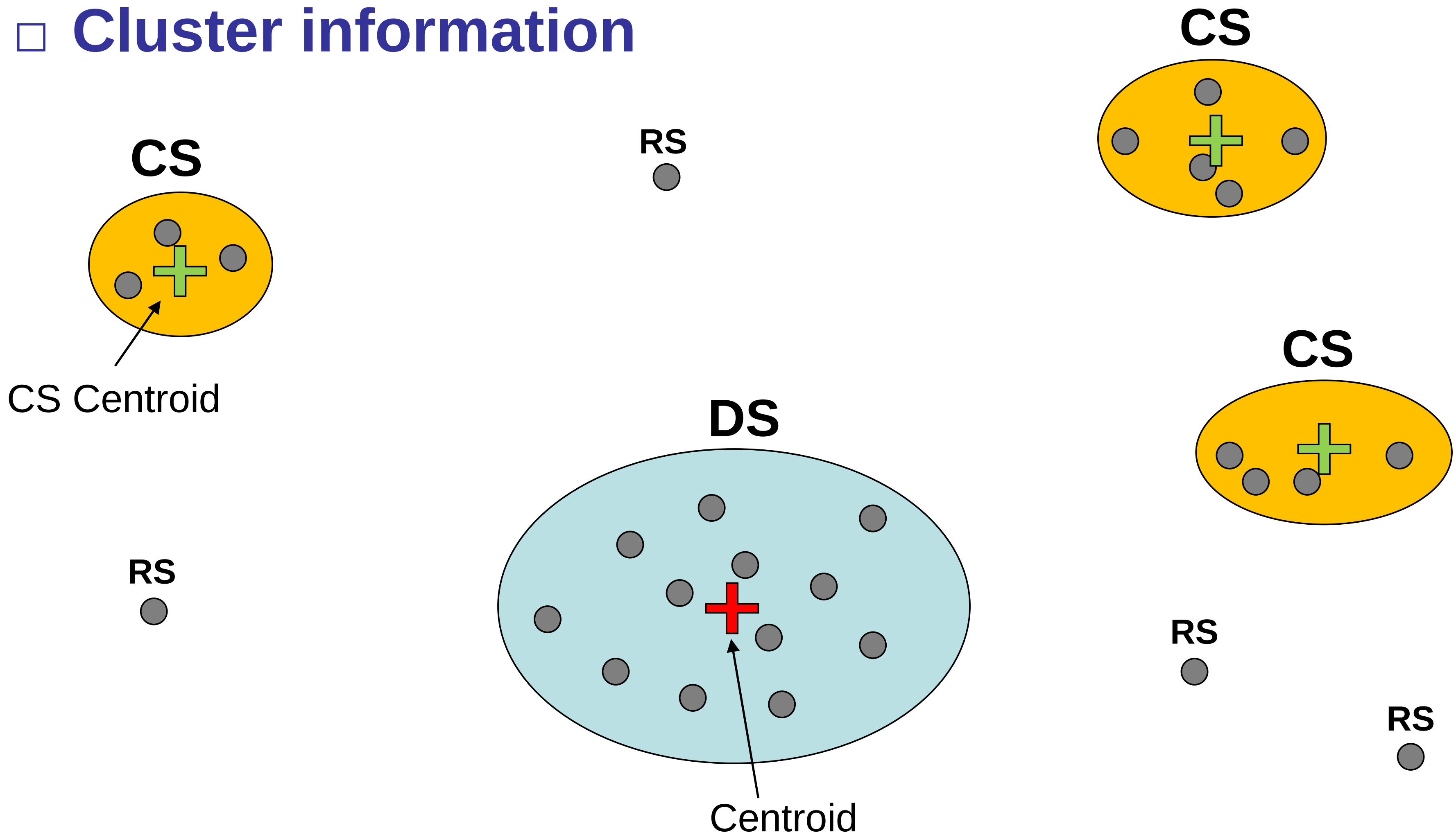
3. Finalize cluster information

BFR

- **Cluster information**
  - Discard set (**DS**)
    - Points that are close enough to the centroid to be summarized and then discarded
  - Compression set (**CS**)
    - Groups of points that are close together but not close to any cluster centroid
    - Points are summarized in a compression set and discarded (the are not assigned to any cluster!)
  - Retained set (**RS**)
    - Isolated points – kept in memory

BFR

- Cluster information



- **Cluster information (DS + CS)**
  - The number of points (**N**)
    - Points that exist in a cluster
  - Vector SUM
    - Each element is a sum of point coordinates across a dimension

$$v_i = \sum_{j=1}^{N} p_{i,j}$$

  - Vector SUMSQ
    - Each element is a sum of squares of point coordinates across a dimension

$$v_i = \sum_{j=1}^{N} p_{i,j}^2$$

BFR

- **Cluster information (DS + CS)**
  - 2d + 1 values are stored to represent each cluster / compression set
    - d is the number of dimensions
  - Centroid
    - Can be calculated as $\mathbf{c_i = SUM_i / N}$ **(not stored!)**
    - $\mathbf{SUM_i}$ sum of coordinates across $\mathbf{i}^{\mathbf{th}}$ dimension
  - Variance
    - $\mathbf{v_i = (SUMSQ_i / N) - (SUM / N)^2}$ **(not stored!)**
    - Standard deviation is the square root of $\mathbf{v_i}$

BFR

Cluster information (DS + CS) – Benefits

- (ds + cs) * (2d + 1) + rs*d values are stored at any time
  - *ds* number of clusters **- fixed**
  - *cs* number of compression sets
  - *rs* size of retains set
  - As *cs* increases, *rs* decreases

- Large dataset
  - $(ds + cs) * (2d + 1) + rs * d \ll N * d$

BFR

□ Cluster information (DS + CS) – Benefits

- It is easier to compute new centroid from SUM_i
  - Updates are simple additions to existing vector

- Much easier to compute new variance using SUM_i and $SUMSQ_i$

- Very efficient when combining sets
  - Simple computation of a new centroid and variance

BFR

Dataset chunk processing

1. Load chunk from memory

2. Find points that are ***close enough*** to a cluster centroid
   - Update cluster data
   - Discard points

BFR

□ Dataset chunk processing

3. Cluster the remaining points along with the points in the retain set **(RS)**
   - Points are added to compression sets **(CS)**
   - Any main-memory cluster algorithm can be applied
   - Define criterion when to add points to clusters and when to merge clusters
   - CS data is updated and points are discarded!

4. Check if some compression sets **(CS)** can be merged

BFR

- **Finalized cluster information**
  - CS and RS sets can be treated as outliers and discarded
  - CS and RS are merged with a cluster whose centroid is **close enough**

BFR

- **When is a point close enough to be included into a cluster?**
- **When to merge compression sets?**

BFR

Mahalanobis distance (MD)

- If **MD** is **blow** a certain **threshold** - likelihood that a point **belongs** to a certain cluster is **high**

- Normalized Euclidian distance form centroid

$$MD(x,c) = \sqrt{\sum_{i=1}^{d} \left(\frac{x_i - c_i}{\sigma_i}\right)^2}$$

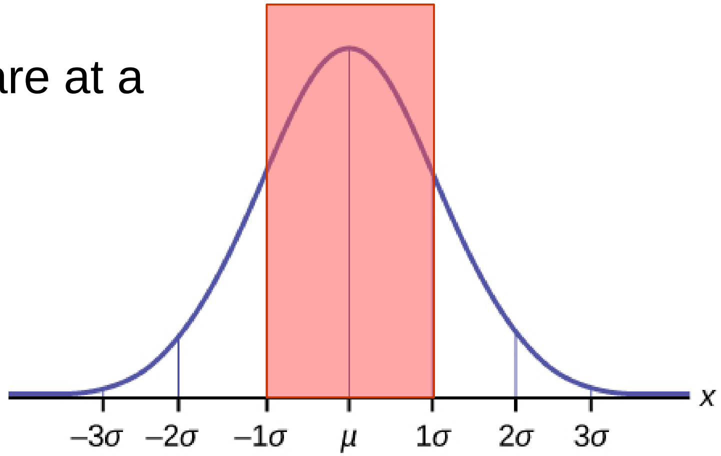
  - x point
  - c centroid of the cluster
  - d number of dimensions
  - σ_i standard deviations of points in the cluster

BFR

- **Mahalanobis distance (MD)**
  - Points are **normally distributed** around the centroid!
  - If $x_i - c_i = \sigma_i$

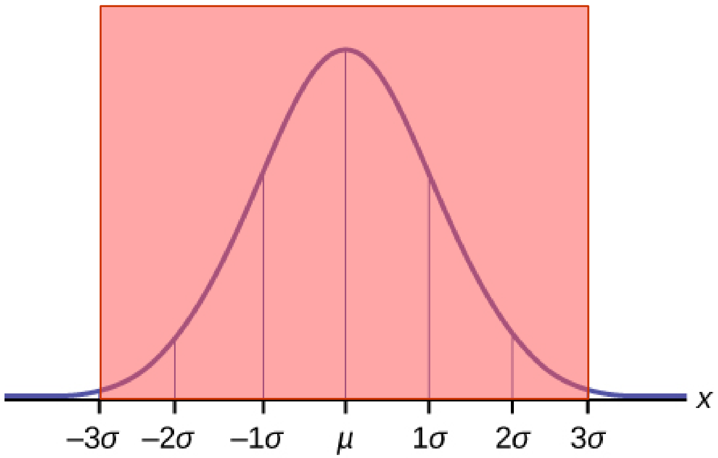
$$MD(x,c) = \sqrt{\sum_{i=1}^{d}\left(\frac{x_i - c_i}{\sigma_i}\right)^2} = \sqrt{\sum_{i=1}^{d}\left(\frac{\sigma_i}{\sigma_i}\right)^2} = \sqrt{d}$$

68% points in the cluster are at a distance of $\sqrt{d}$



- **Mahalanobis distance (MD)**

> 99% points in the cluster are at a distance of $3\sqrt{d}$



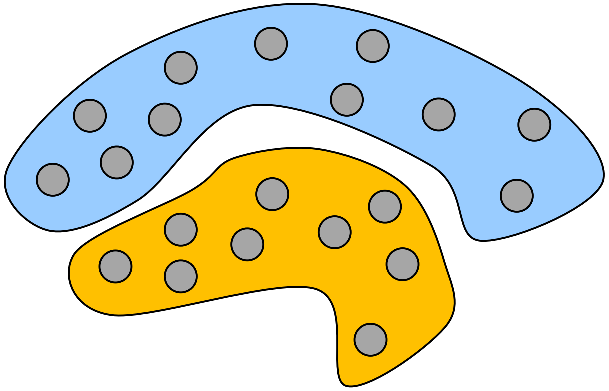
Appropriate threshold can be chosen, e.g. $3\sqrt{d}$

BFR

- **When to merge compression sets?**
  - Calculate combined variance of merged two CS
    - SUM, SUMSQ make this operation easy
  - If variance is below a certain threshold, merge the compression sets
  - Many alternative approaches
    - Treat dimensions differently, as they might differ in importance
    - Cluster density

CURE

- **Clustering using representatives**
  - Can handle massive datasets
  - Euclidian space
  - No assumptions on the cluster shape!



CURE

- **1. pass**

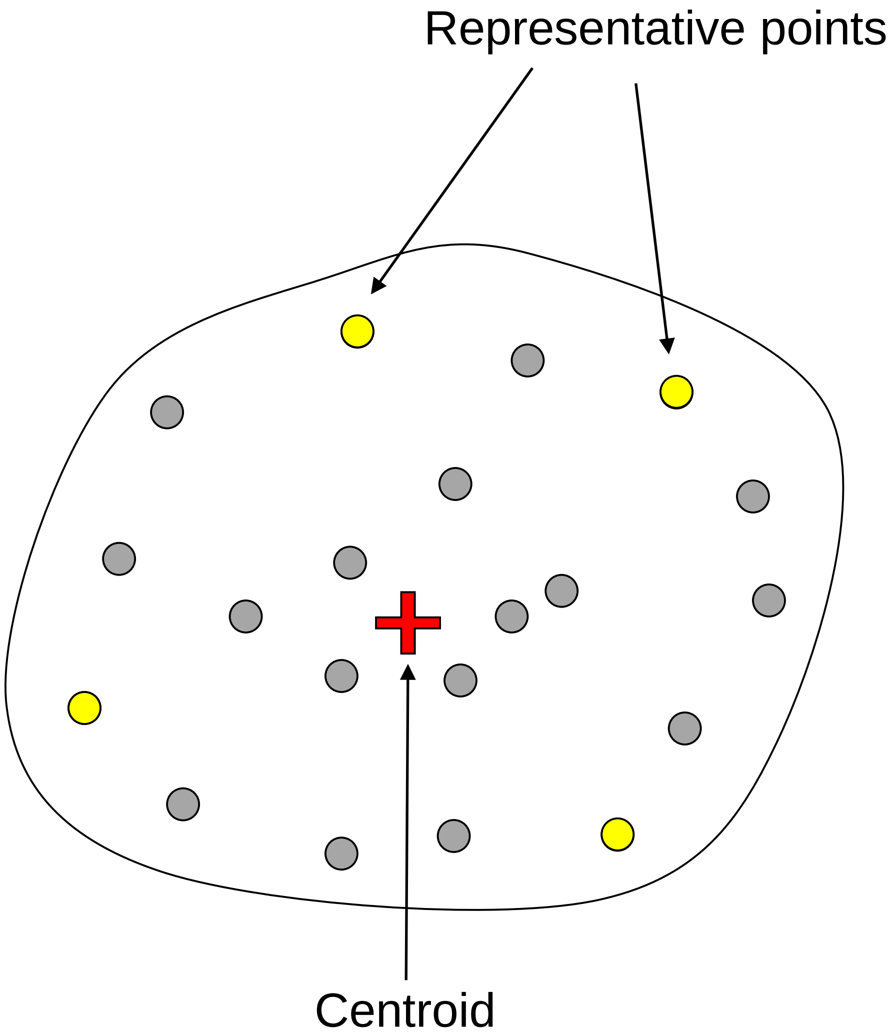
1. Take a data set sample that fits into main memory

2. Cluster the sample data set
   - Any algorithm that can handle oddly shaped clusters
   - Usually hierarchical clustering

3. Choose **n** representative points
   - **Points should be as distant from each other as possible (see previously descried methods)**
   - Usually a small set of points is chosen

CURE

- **1. pass**



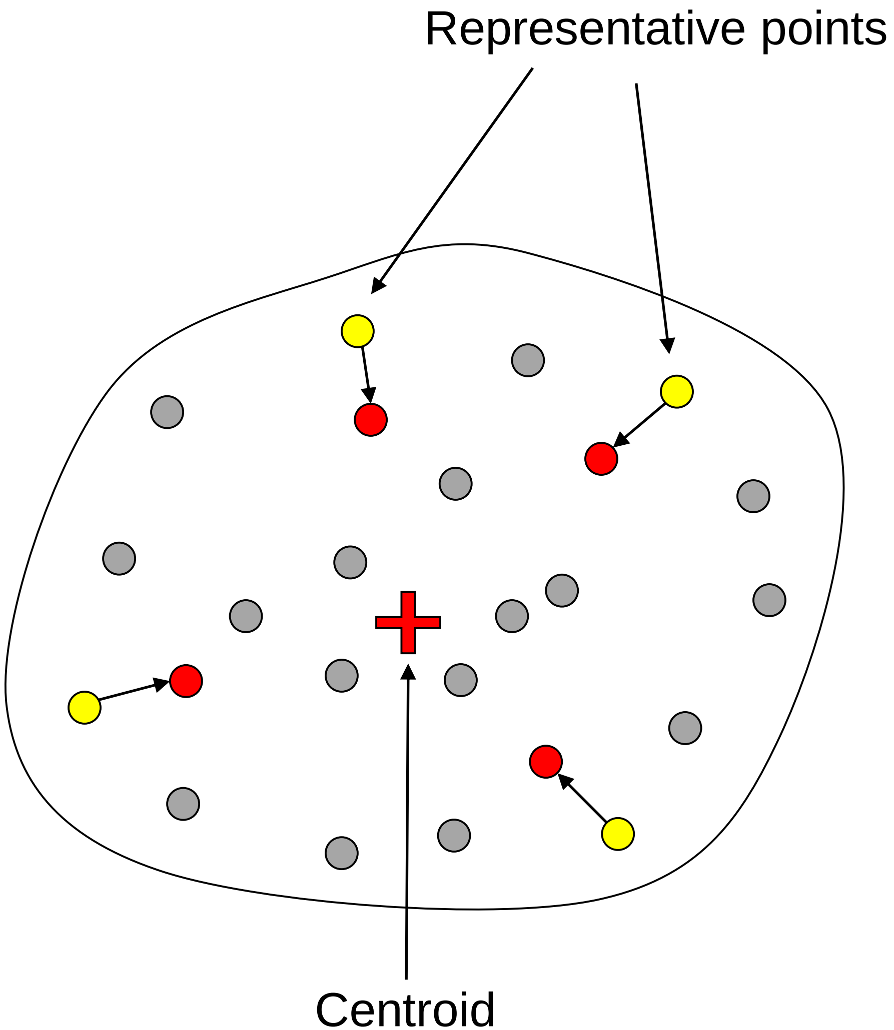
CURE

- **1. pass**

4. Move the representative points some portion of a distance towards the centroid
   - e.g. 20%
   - The data set is not actually changed, these are simply synthetic points used by the algorithm!

CURE

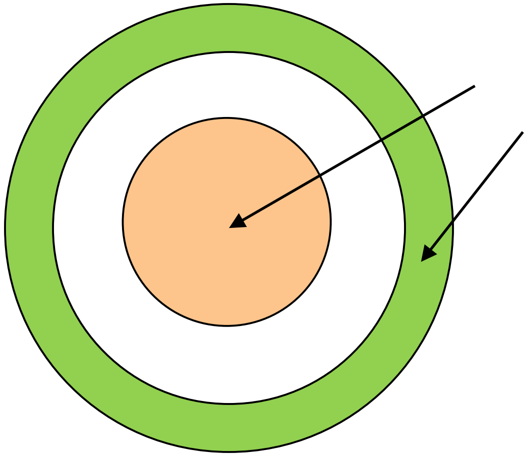
- **1. pass**



CURE

- **1. pass**

5. Check if clusters can be merged
   - Merge clusters if they have **representative points** that are sufficiently close
   - Note that distance to centroids is not taken into account!



Both clusters have the same centroid!

CURE

- **2. pass**
  - Go through all points **p** in the original data set
  - Assign **p** to a cluster **c** if the point is closest to one of its representative points
    - All other representative points across all clusters are more distant

Literature

1. J. Leskovec, A. Rajaraman, and J. D. Ullman, "Mining of Massive Datasets", 2014, Chapter 7 : “Clustering" (link)
2. Distance and Similarity measures: https://reference.wolfram.com/language/guide/DistanceAndSimilarityMeasures.html