Data analysis and stochastic modeling

Lecture 3 – Cluster analysis

Guillaume Gravier

guillaume.gravier@irisa.fr

with a lot of help from Dr. HOI Chu's course

https://svn.mosuma.net/r4000/doc/course/ci6227/public/lectures/lecture07cluster.pdf

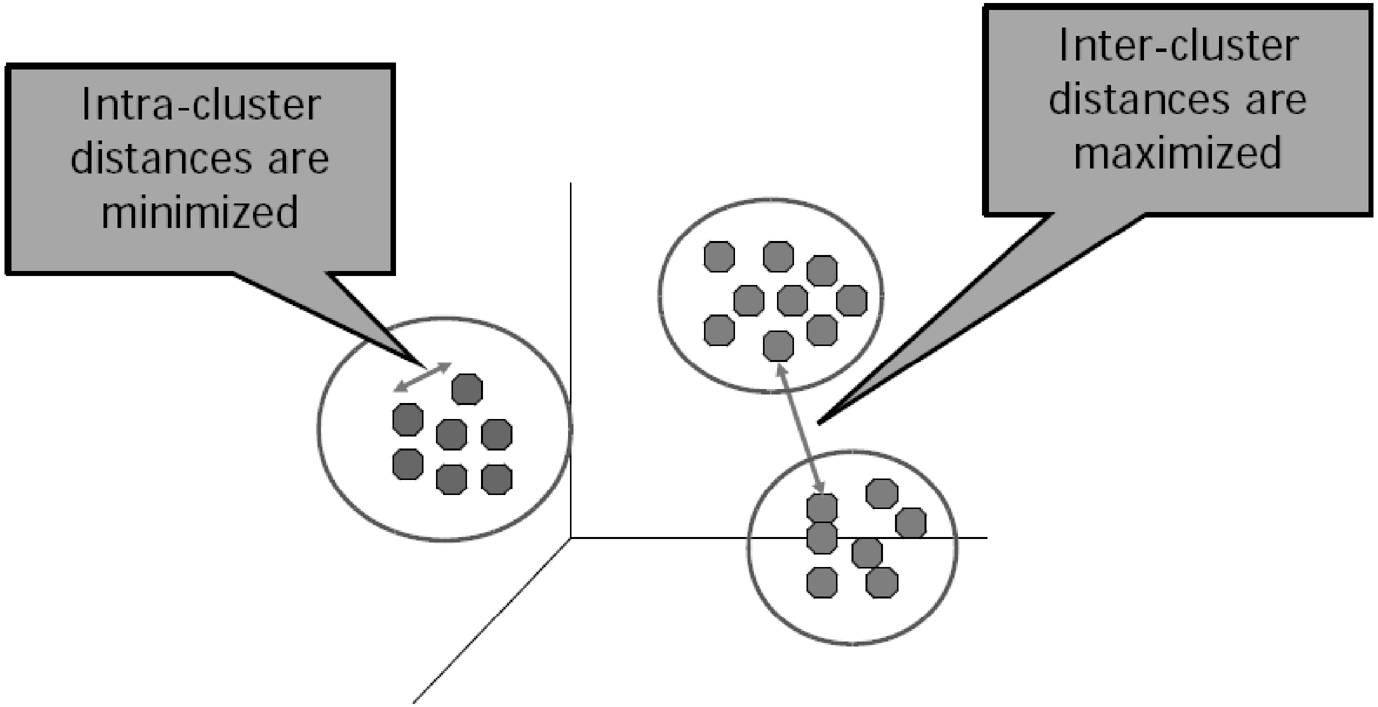
What are we gonna talk about today?

- What's cluster analysis?
- Partitioning clustering
  - $\longrightarrow$ k-means and the likes
- Hierarchical clustering
  - $\longrightarrow$ bottom-up clustering, linkage methods
- A quick survey of other methods
  - $\longrightarrow$ density methods, spectral clustering, etc.
- Case study

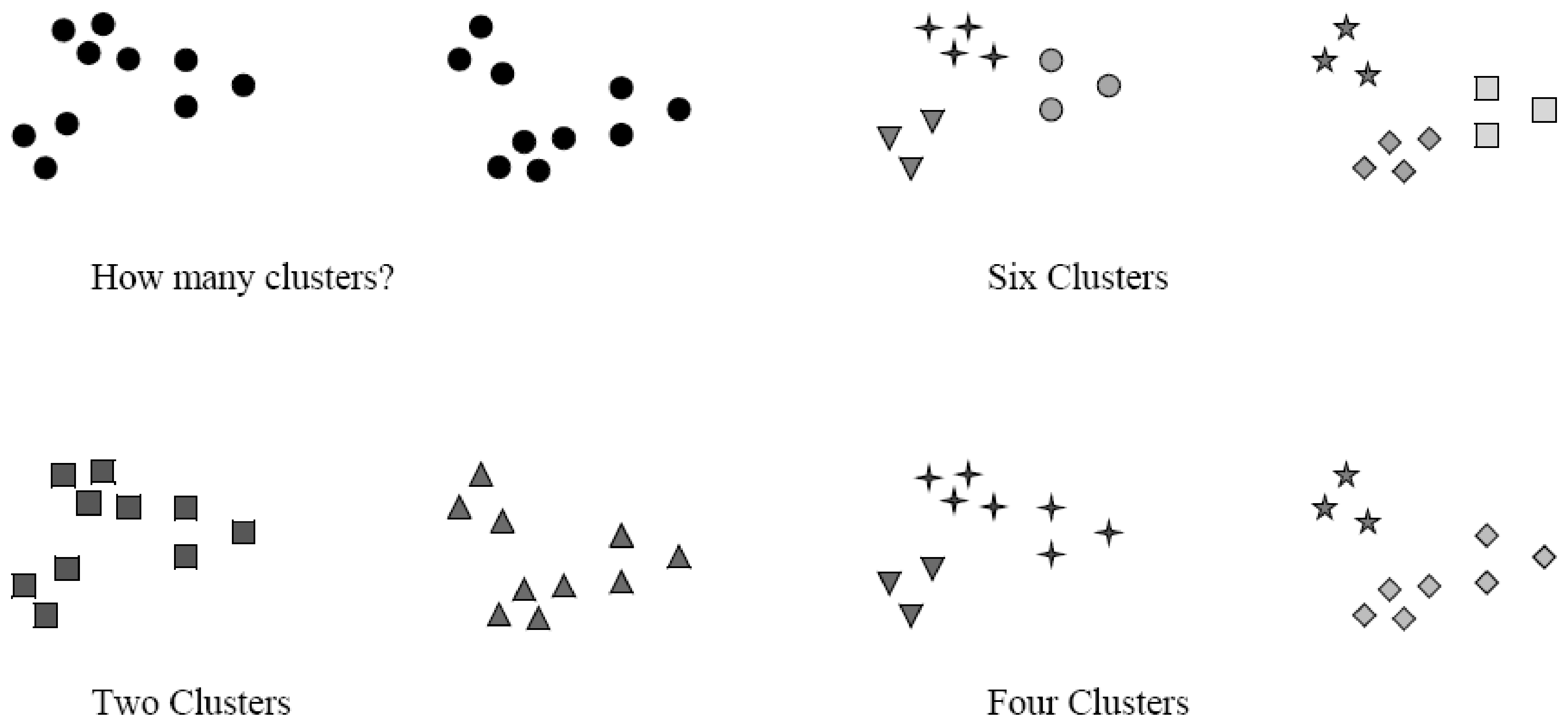
In short: an overview of the art of grouping data according to their similarity.

What is clustering?

Clustering consists in grouping data together in "classes" where objects in a class are similar and objects from different classes are different.



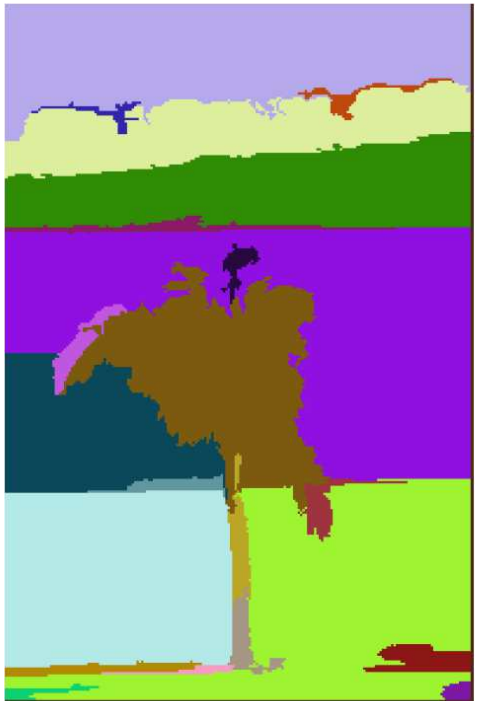
An ambiguous notion



What's clustering good for?

- **(exploratory) data analysis and understanding** (data mining)
  - Biology: taxonomy of living things
  - Information retrieval: grouping similar documents
  - Land use: identifying areas with similar properties
  - Marketing: discover distinct groups of customers
  - City planning, earth quake analysis, etc.

- **pre-processing tool**
  - quantization, coding and compression
  - classification, segmentation, etc.
  - k nearest neighbor search



Quality

Good clustering

= low within-class similarity

= high across-class similarity (eventually)

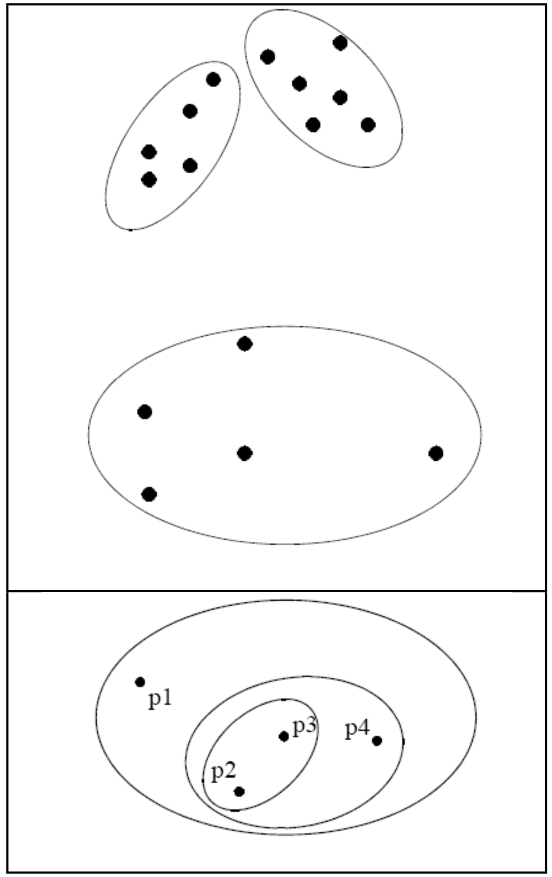
but other factors are to be considered:

- scalability
- dynamic behavior
- ability to deal with noise and outliers
- (in)sensitive to the order of input
- etc.

Two main philosophies

Clustering $\Rightarrow$ a set of clusters

- partitioning data: divide the entire data set
  - non-overlapping clusters
  - each object belongs to one a only one cluster
- aggregating data: group similar data
  - nested clusters
  - hierarchical structure



But other philosophies do exist: density-based, grid-based, model-based, constraints-based, etc.

About pairwise distances

similar objects $\Rightarrow$ notion of distance

- Any measure of the similarity $d(x_i, x_j)$ between two objects can be used to solve the problem
- The adequate measure highly depends on the nature of the data and on the nature of the problem
  - distance measures
    - Euclidian, Manhattan, Mahalanobis, χ^2, etc.
  - similarity (no triangular inequality)
    - cosine, template matching, edit distance, generalized likelihood, etc.
  - conceptual measures
    - whatever one can think of...

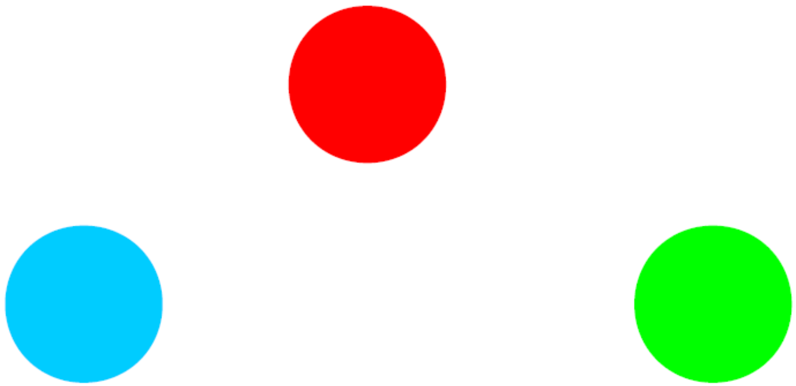
Some typology elements

- **exclusive or not?**
  - points might belong to several clusters (with or without weights)
- **fuzzy or not?**
  - in fuzzy algorithms, a point belong to all clusters with a weight $\in [0, 1]$
- **partial or not?**
  - only part of the data is clustered
- **homogeneous or not?**
  - clusters of very different shape

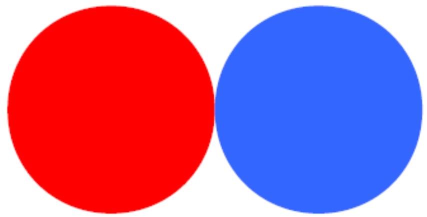
Some typology elements (cont'd)

- **well separated clusters**
  - every point in the cluster is closer to every point in the cluster than to any point outside the cluster
- **center clusters**
  - every point in the cluster is closer to the center of the cluster than to the center of any other cluster
- **contiguous cluster**
  - every point in the cluster is closer to at least one point in the cluster than to any point in another cluster
- **density-based cluster**
  - dense region of points in a cluster separated from other clusters by low-density regions

Some typology elements (cont’d)

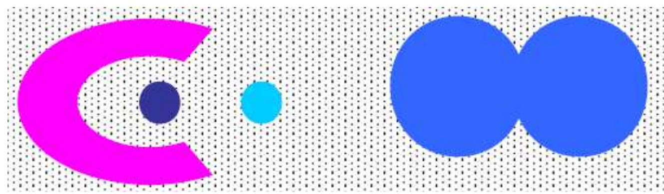


well separated



center

contiguous



dense

Some typology elements (cont'd)

To select the most appropriate solution one must look at the following elements

- Type of proximity or density measure
- Sparseness
  - Dictates type of similarity; Adds to efficiency
- Attribute type
  - Dictates type of similarity
- Type of Data
- Dimensionality
- Noise and Outliers
- Type of Distribution

The k-means algorithm

Idea: Divide some data x_i into K clusters represented by the mean value of their members c_k (centroids), so as to minimize the overall quantization error

$$e = \sum_i d(x_i, c_{f(i)})$$

Algorithm:

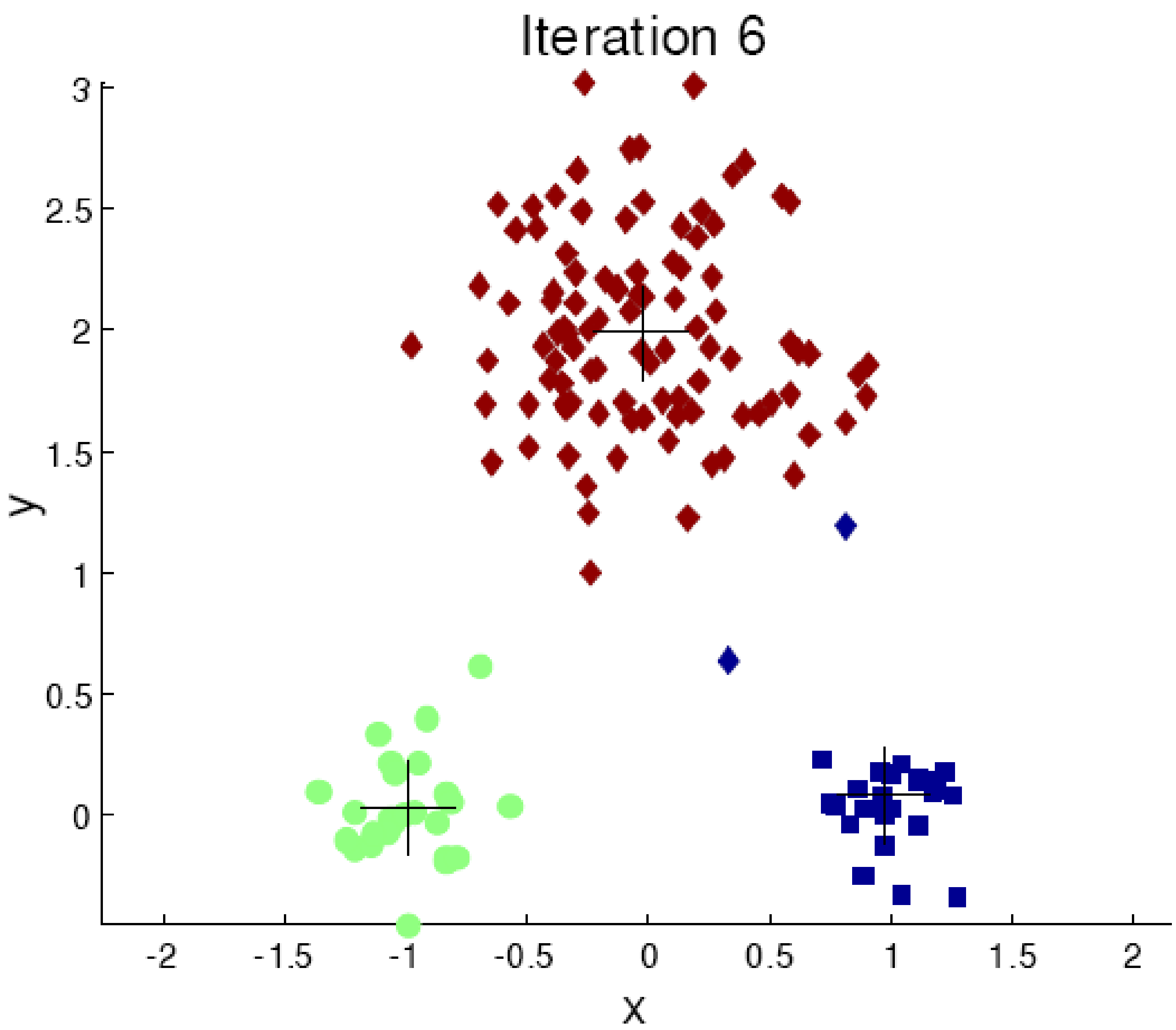
initialize K centroids c_k
while not converged **do**
  for $i = 1 \to N$ **do**
    assign x_i to the closest centroid ($f(i) \leftarrow \arg\min_k d(x_i, c_k)$)
  end for
  for $i = 1 \to K$ **do**
    update centroid c_k from all assigned points
  end for
end while

[J. B. McQueen. Some methods for classification and analysis of multivariate observations. Proc. Symposium on Math., Statistics, and Probability, pp. 281-297, 1967]

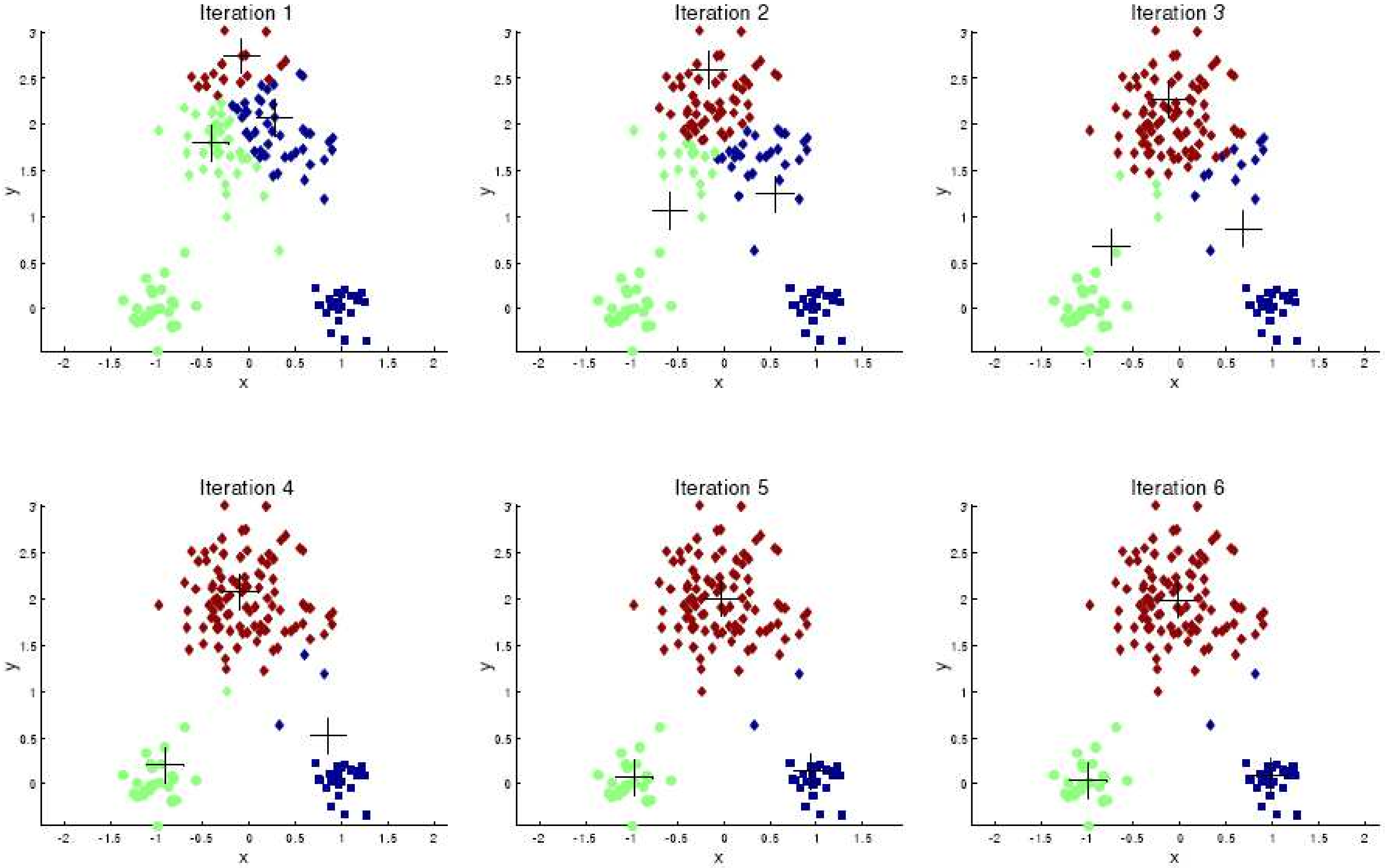
The art of k-means clustering

- the distance is the key
  - mean has to have a meaning ...
  - ... but we can use the median instead (the *k-medians* or *k*-medioids algorithm)
- convergence = nothing moves anymore!
  - convergence is guaranteed
  - often in 10 to 20 iterations
- complexity = O(iKNd)
  - i = #iterations, d=dimension(x_i)
- initialization is tricky!
  - random choice
  - multiple runs
  - hierarchical k-means (LBG)

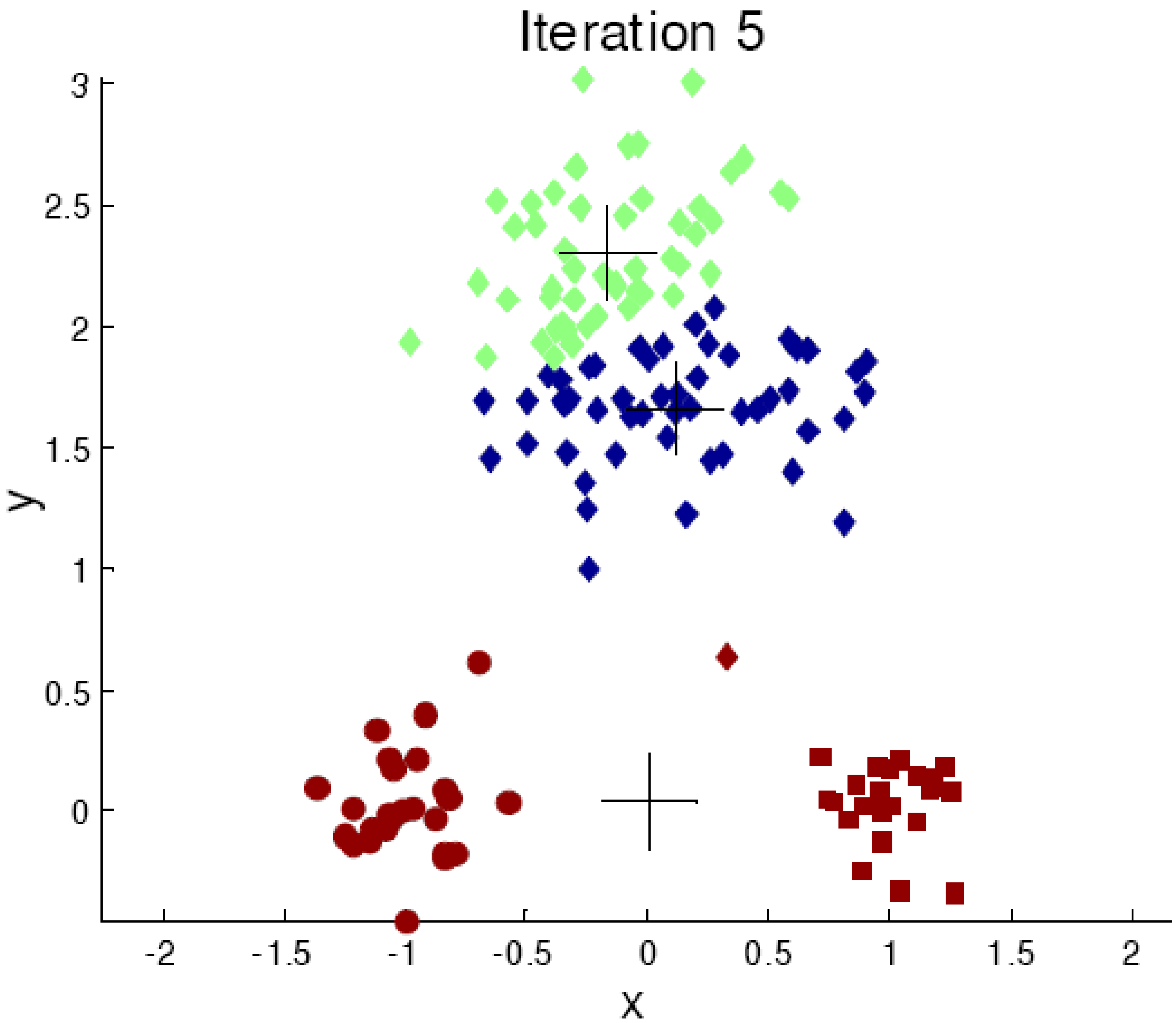
Initialization issues



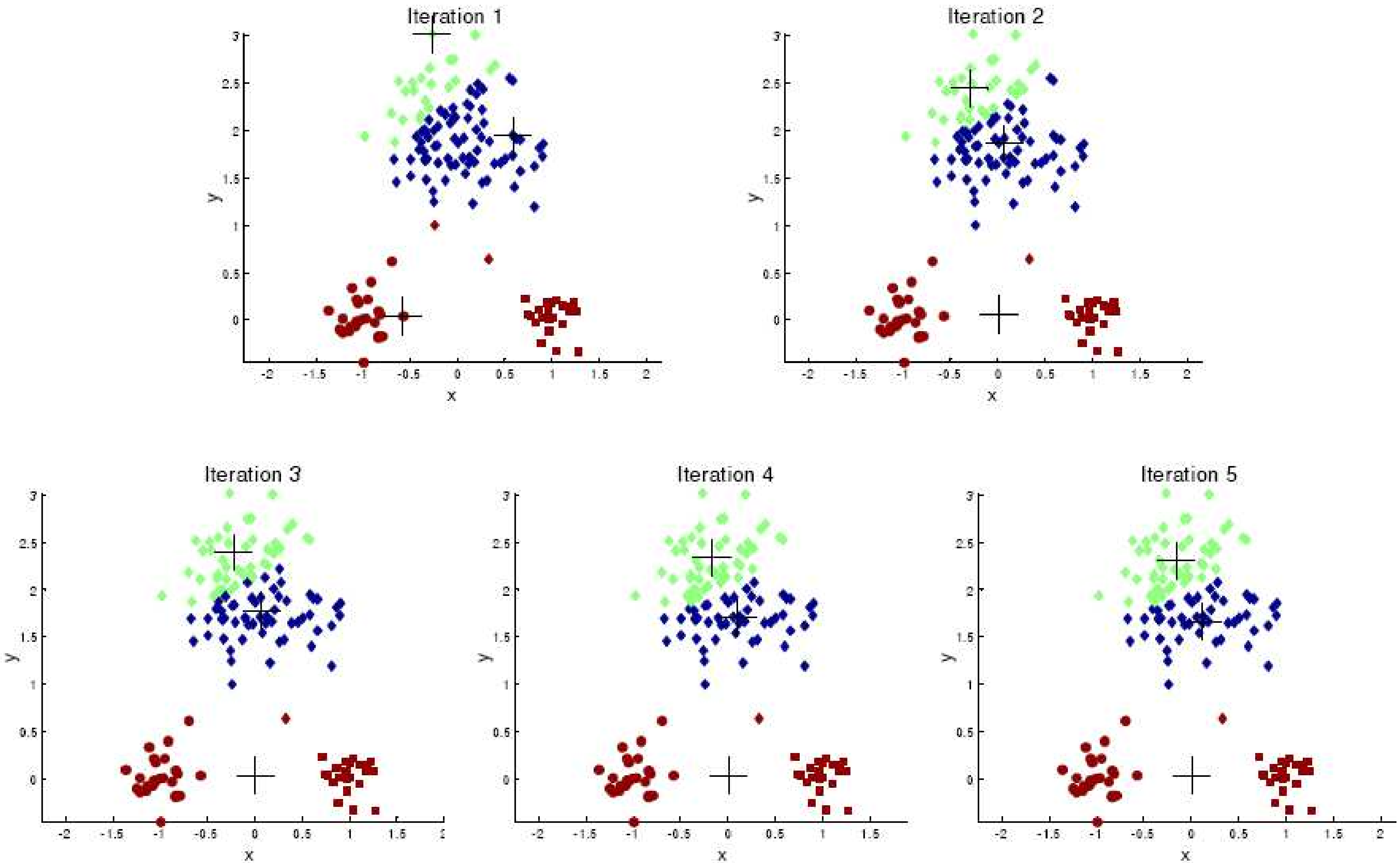
Initialization issues (cont'd)



Initialization issues (cont'd)



Initialization issues (cont'd)



Initialization issues (cont'd)

- **multiple runs**
  - but that's costly!
  - have to deal with "*dead*" clusters (e.g., replace them)
- **hierarchical k-means**
  - the original Linde-Buzo-Gray algorithm (aka bisecting k-means)

---

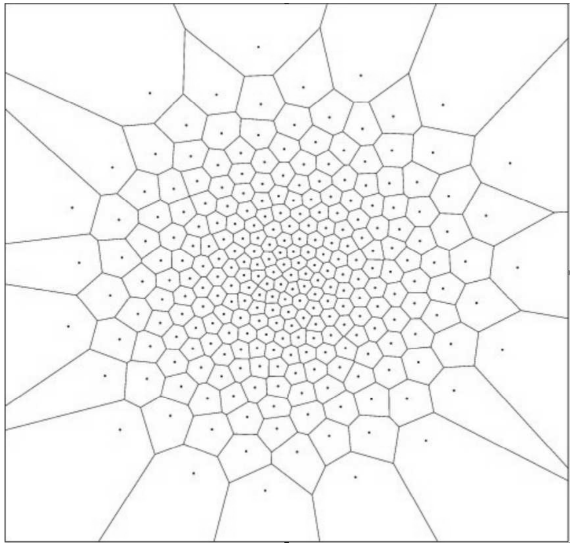
initialize centroids $c_1(1)$ to gravity center
$i \leftarrow 1$
while not enough clusters ($i < p$) **do**
    split each centroids $c_i(j)$ (along maximum variance line)
    run k-means
end while

  - and its many variants
    - split only the biggest cluster $\rightarrow$ arbitrary number of clusters instead of 2^p
    - points stay within their parent cluster $\rightarrow$ much faster

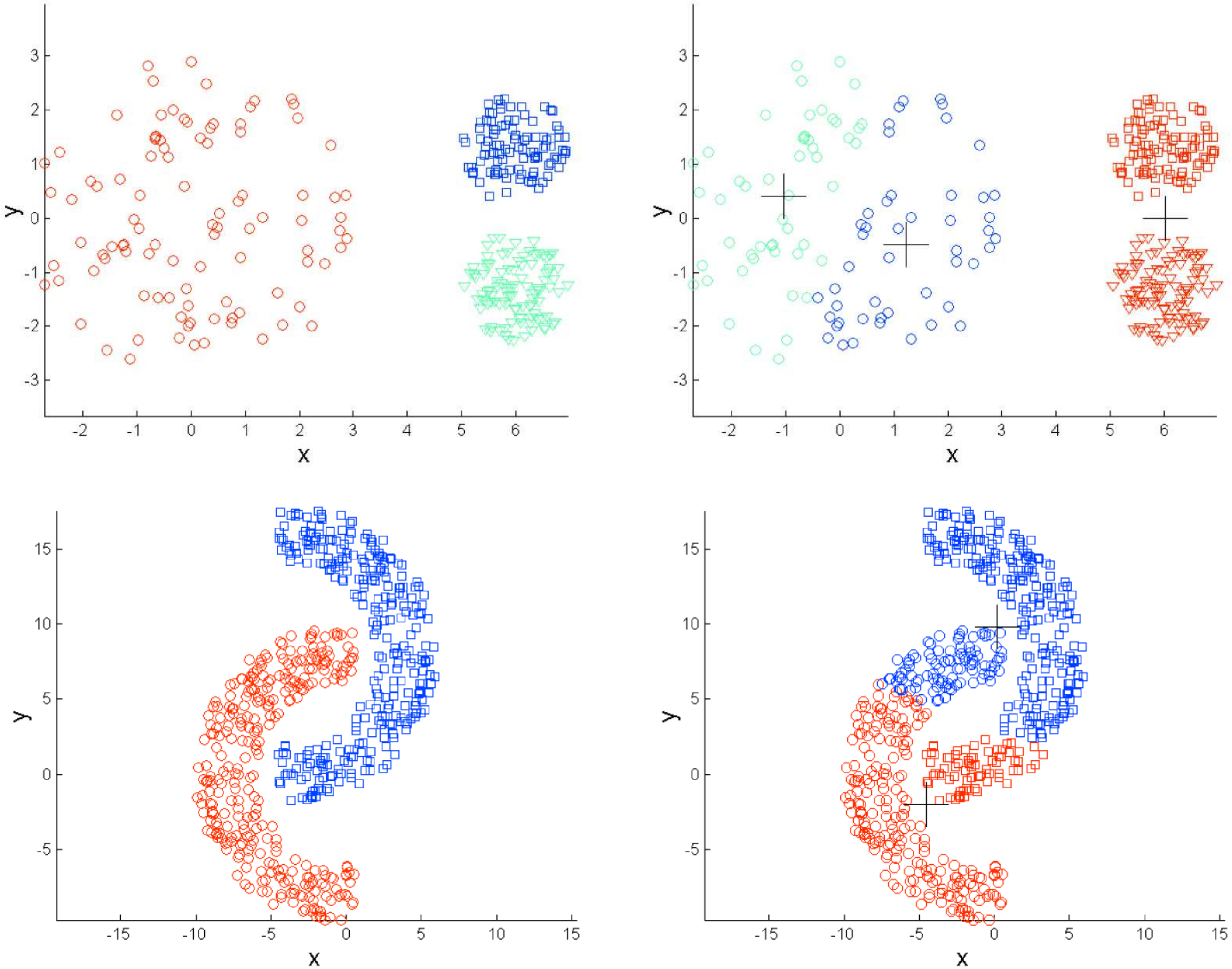
In any case, local optima!

Pros and Cons

- **Pros**
  - simple, clear and popular
  - decently efficient
  - guaranteed convergence
  - can accomodate any shape (with enough clusters)
- **Cons**
  - initialization and local optima
  - need to define the number of clusters
  - convex clusters of roughly the same size and density
  - tends to create unbalanced cells
  - highly sensitive to noise and outliers



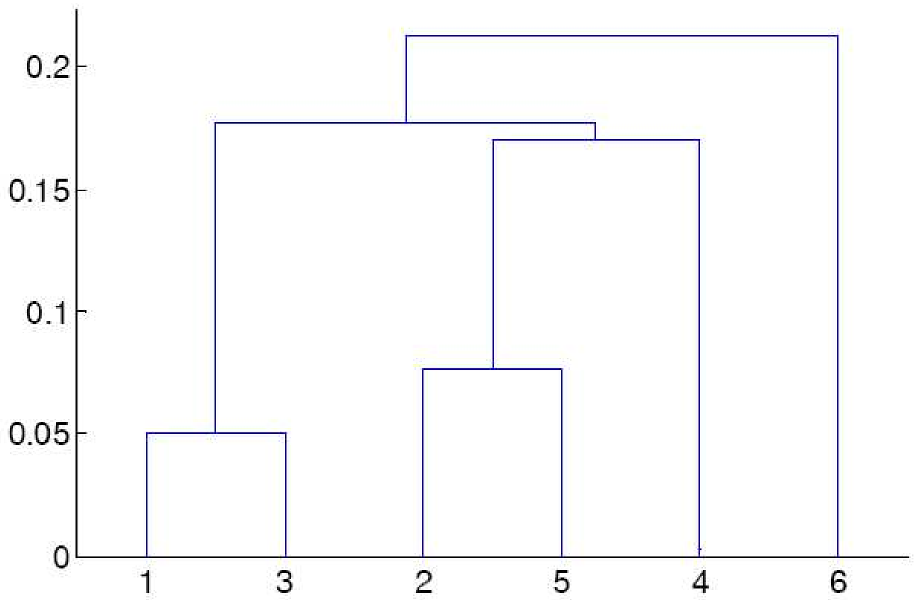
Pros and Cons illustrated

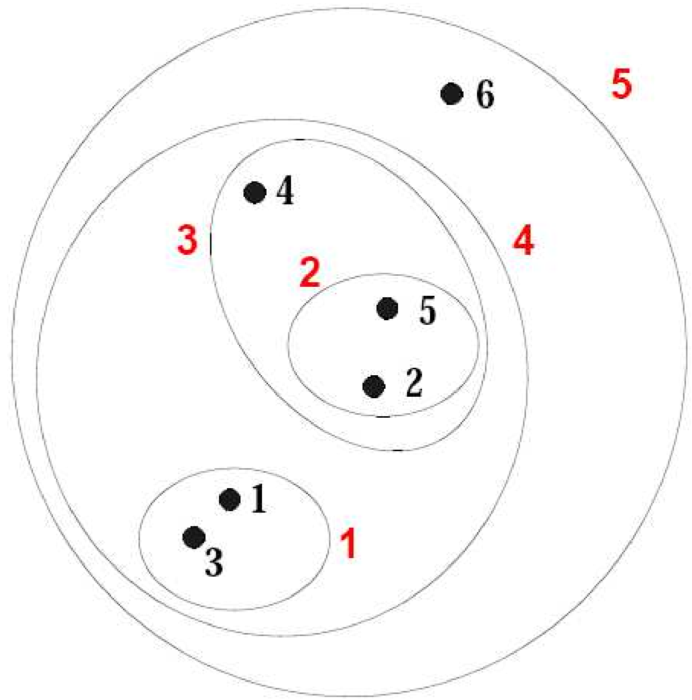


Hierarchical clustering

Idea: Progressively generate clusters by merging or divising data

- generate nested clusters
- can be visualized as a dendogram





Agglomerative vs. divisive

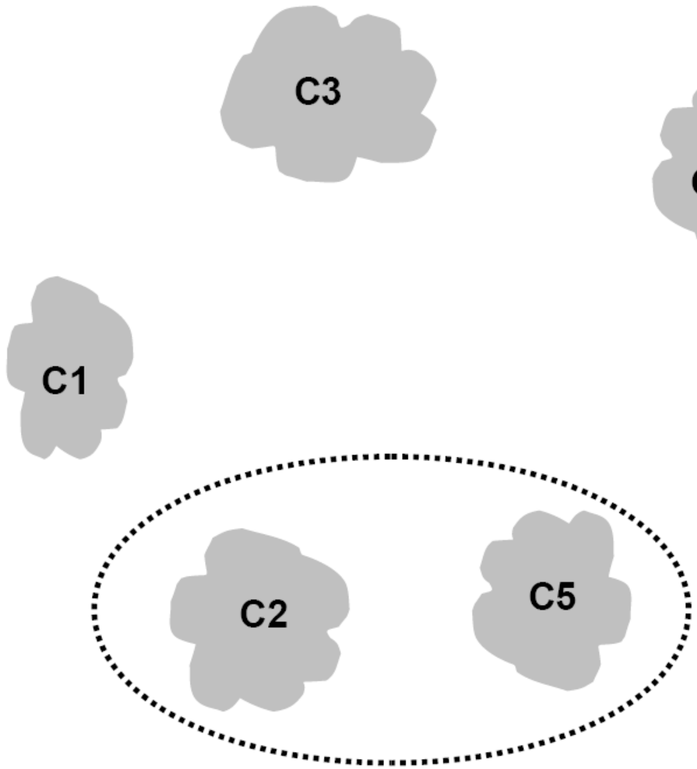
- **Agglomerative bottom-up clustering**
  - Bottom-up construction of the dendogram by progressively merging clusters

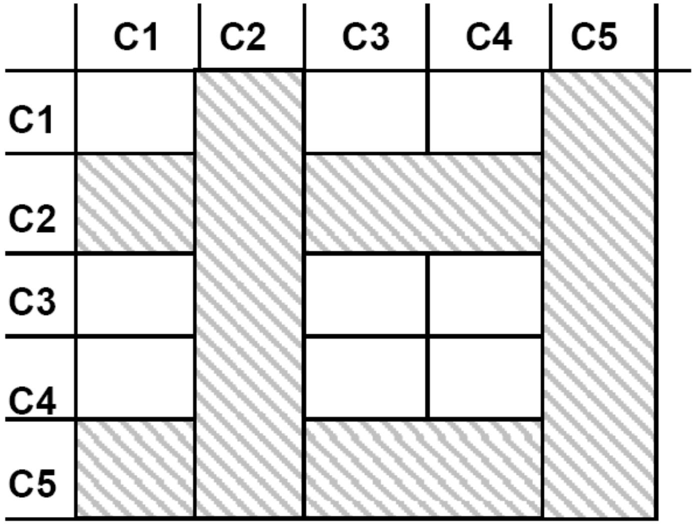
initialize N singleton clusters
nclusters $\leftarrow N$
while nclusters > 0 **do**
  merge the two closest clusters
  nclusters $\leftarrow$ nclusters $- 1$
end while

- **Divisive bottom-up clustering**
  - top-down construction of the dendogram
  - DIANA (Divisive ANAlysis)

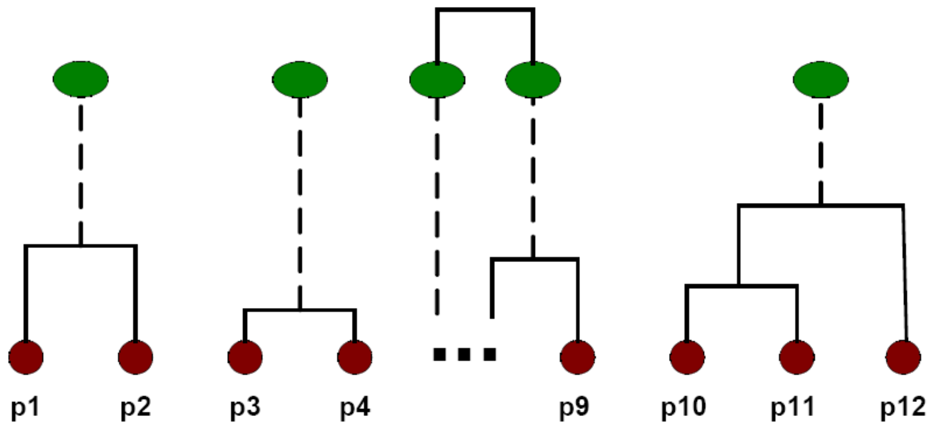
Proximity matrix and bottom-up clustering

update the proximity matrix.

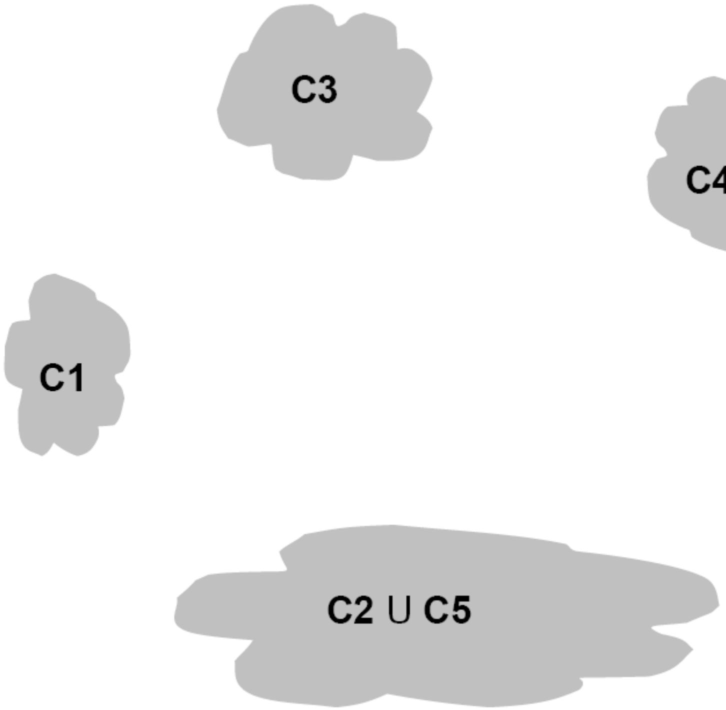




Proximity Matrix

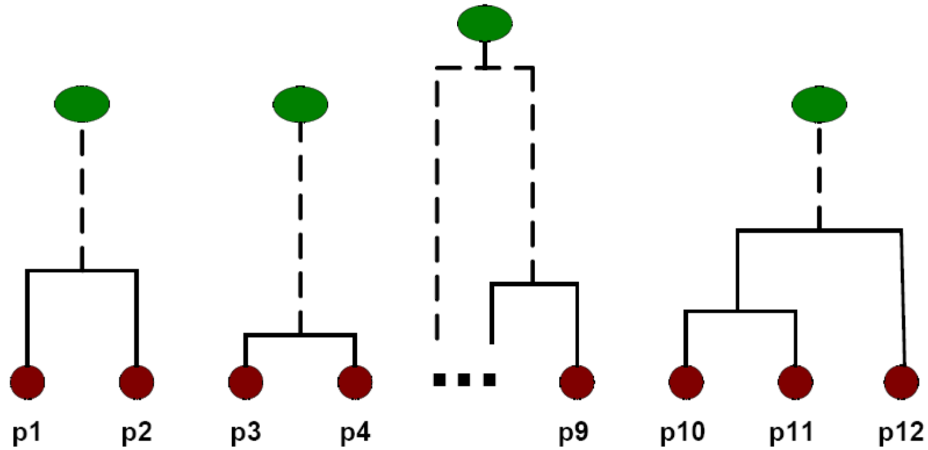


Proximity matrix and bottom-up clustering (cont'd)



	C1	C2 U C5	C3	C4
C1		?		
C2 U C5	?	?	?	?
C3		?		
C4		?		

Proximity Matrix



“Linkage” types

linkage = how to measure the distance between clusters

- single linkage: $D(A, B) = \min(d(x, y) \quad \forall (x, y) \in A \times B)$

  $\rightarrow$ for well-separated classes only

- total linkage: $D(A, B) = \max(d(x, y) \quad \forall (x, y) \in A \times B)$

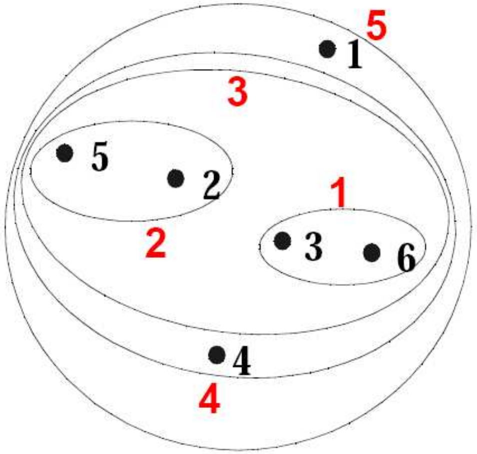
  $\rightarrow$ favors large clusters

- average linkage = average distance between elements in A and B

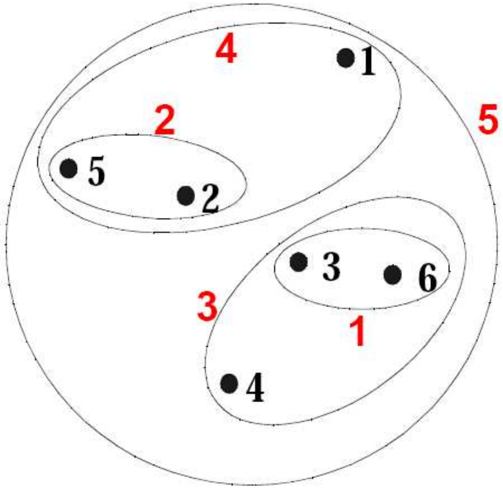
  $\rightarrow$ robust to noise and outliers but biased towards globular clusters

- Ward’s linkage = increase in variance for the cluster being merged
- and many others, including distance between mean/median or between (statistical) models of the data

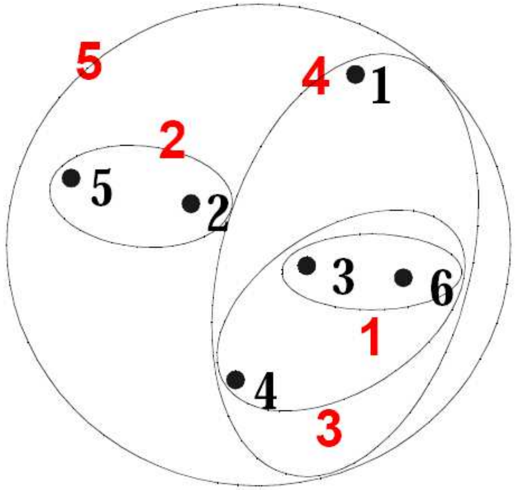
"Linkage" types (cont'd)



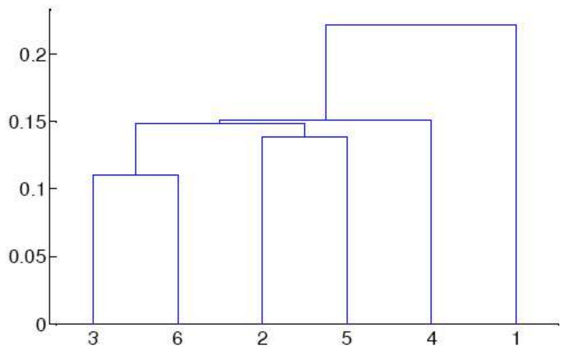
Sing Link (MIN)

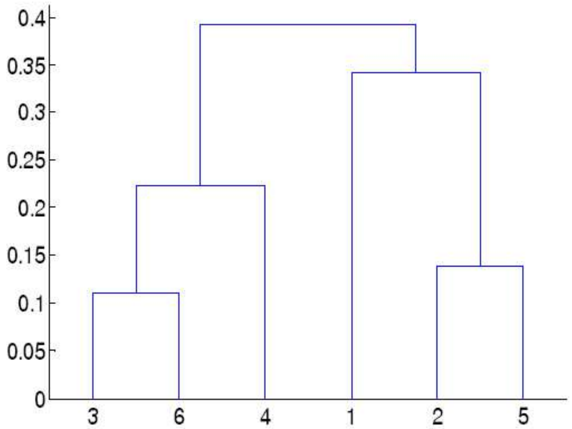


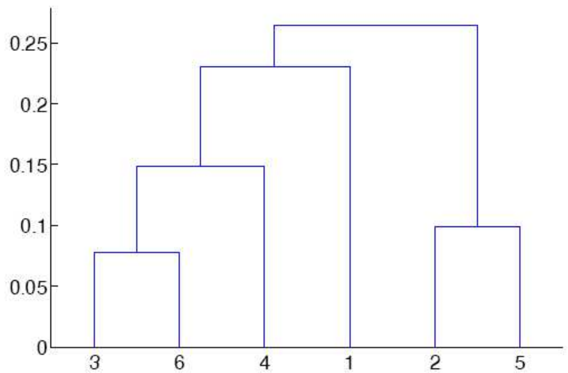
Complete Link (MAX)



Group Average





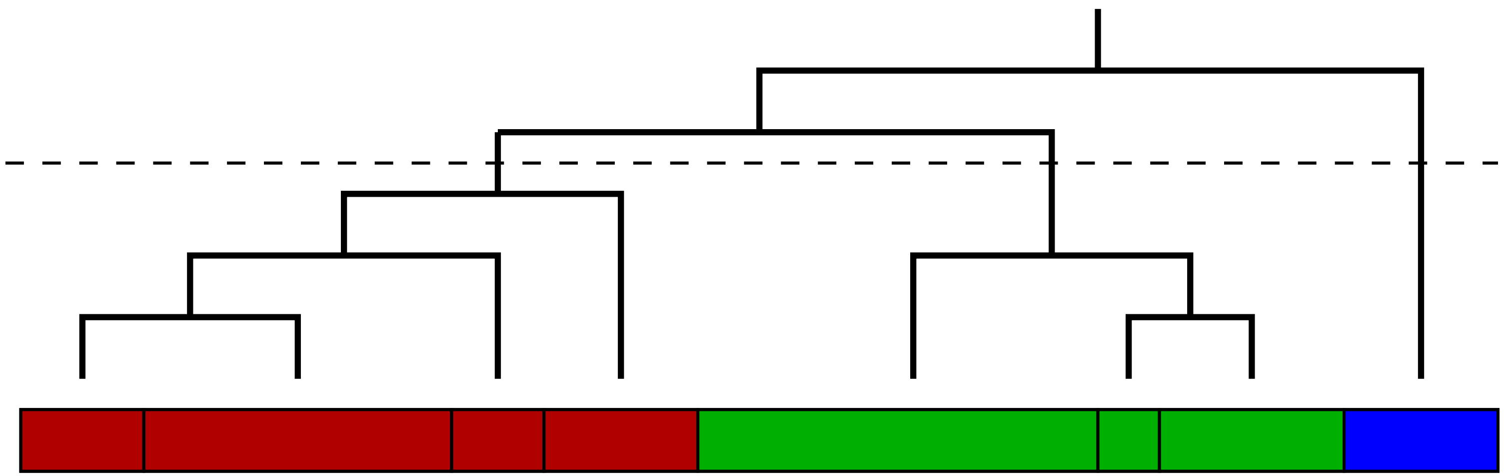


Pros and Cons

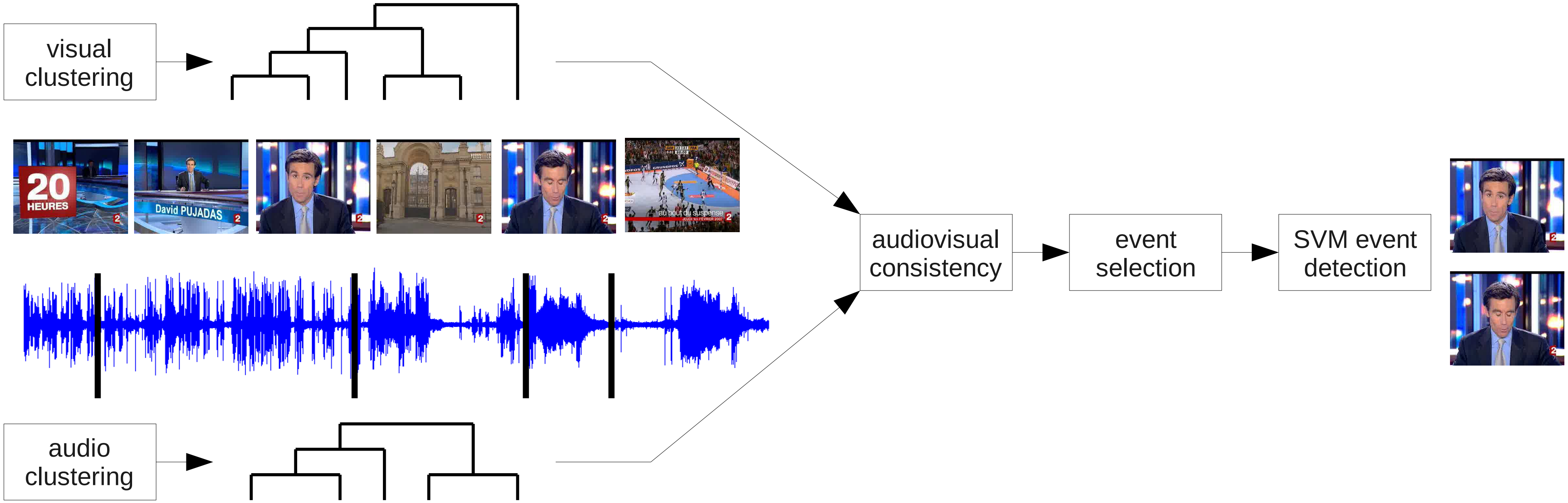
- **Pros**
  - better than k-means with non-metric distances
  - can combine metrics and cluster balancing criteria
  - possibility to define a posteriori the number of clusters (though not so easy in practice)
- **Cons**
  - quite slow and computationally demanding (O(N^3) or O($N^2 \log(N)$))
  - local optima may not be globally good
    - cannot undo what was done previously
    - require relocation methods
- cutting the dendogram is not as easy as it seems to be

Bottom-up clustering for temporal partitioning

- detect boundaries of segments $\longrightarrow$ see hypothesis testing
- group together segments with similar characteristics
  - model based representation of clusters (Gaussian densities and mixtures)
  - Kullback-Leibler divergence, generalized likelihood ratio
- find out where to cut the dendogram
  - model selection approaches: Bayesian information criterion

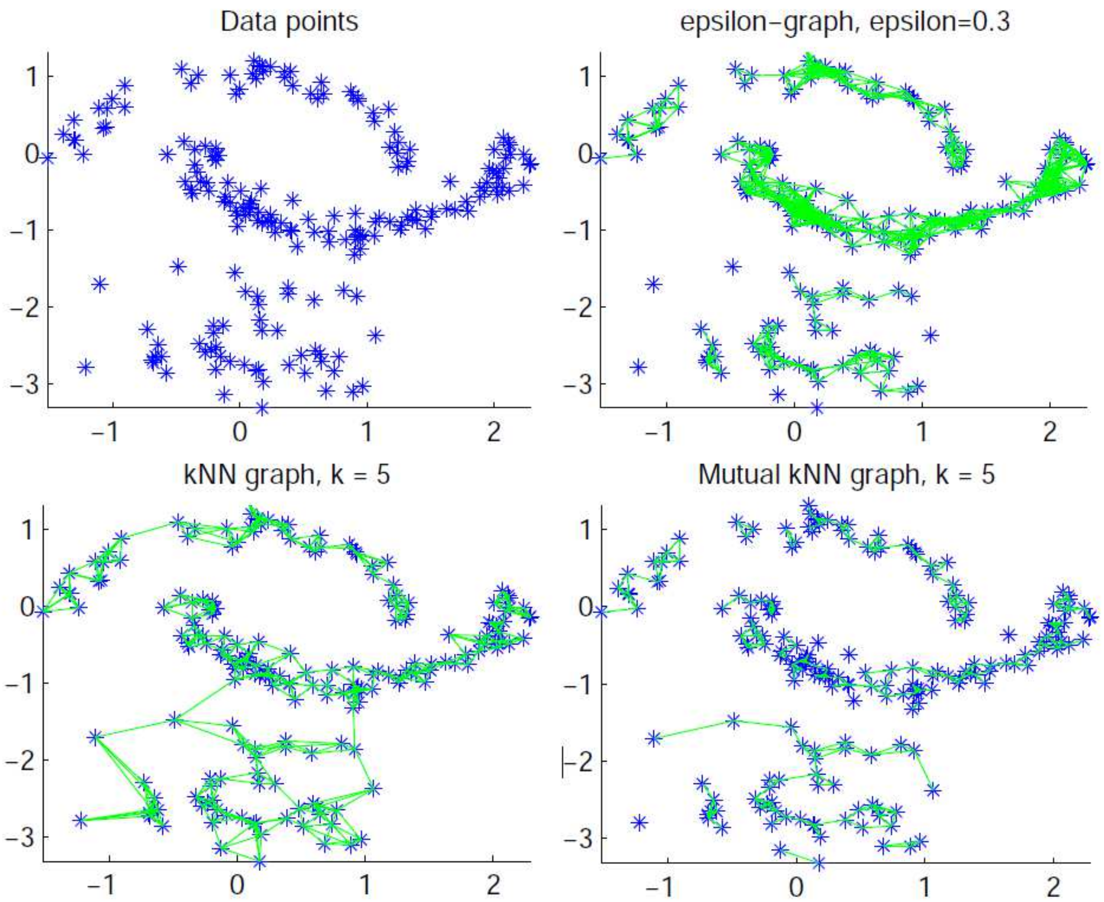


Mining TV sequences with bottom-up clustering



Similarity graphs for spectral clustering

- data point = node in a graph
- edges encode the similarity between two nodes with weights w_{ij}
  - ϵ-neighbor graphs
  - k-nearest neighbor graphs
  - fully connected graphs



[Ulrike von Luxburg. A tutorial on spectral clustering. Statistics and Computing, 17(4):395-416, 2007]

Graph Laplacian

The unnormalized graph Laplacian is the $n \times n$ matrix defined as

$$\mathbf{L} = \mathbf{D} - \mathbf{W}$$

with $\mathbf{D}$ a diagonal matrix with elements $d_i = \sum_j w_{ij}$.

Important properties of $\mathbf{L}$:

- $\mathbf{L}$ as n non-negative, real valued eigenvalues
- the smallest eigenvalue is 0 and corresponds to the unit eigen vector $\mathbb{1}$
- the multiplicity of the eigenvalue 0 is the number of connected components

$\Rightarrow$ **exploit the eigenvalues of the Laplacian of the similarity graph to perform clustering**

Normalized versions of the Laplacian are often used in practice.

The algorithmics of spectral clustering

Input: similarity matrix $\mathbf{S} \in \mathbb{R}^{n \times n}$, number of clusters k

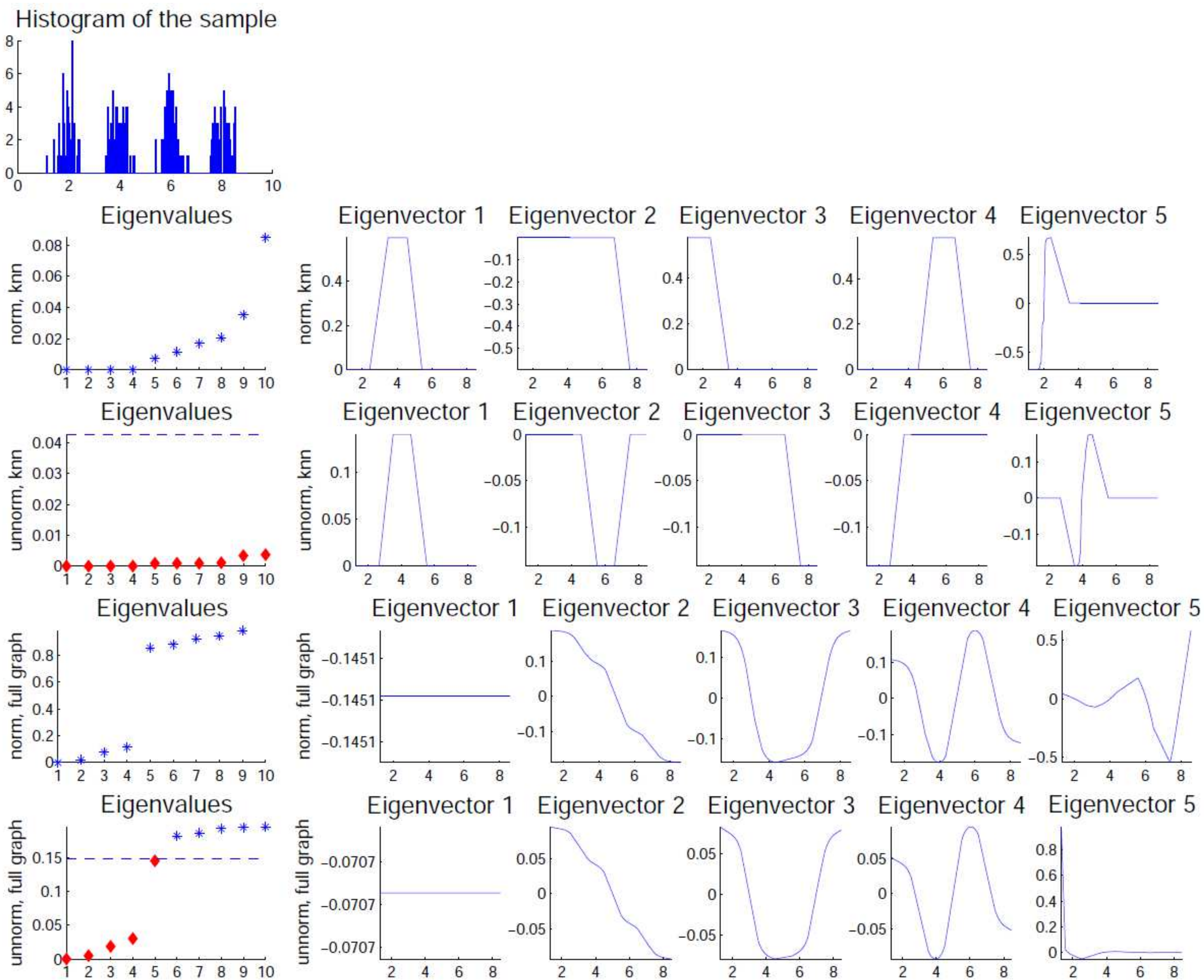
Algorithm:

construct a similarity graph from $\mathbf{S}$ with adjacency matrix $\mathbf{W}$
compute the Laplacian $\mathbf{L}$
compute the k eigenvectors $\mathbf{u}_1, \ldots, \mathbf{u}_k$ of $\mathbf{L}$ associated with the k lowest eigenvalues
define $\mathbf{U} \in \mathbb{R}^{n \times k}$ with $\mathbf{u}_1, \ldots, \mathbf{u}_k$ as columns
define $\mathbf{y}_i \in \mathbb{R}^k$ the i-th row of $\mathbf{U}$ for $i \in [1, n]$
run k-means clustering (or other) on the vector $\mathbf{y}_i$

Strong links with

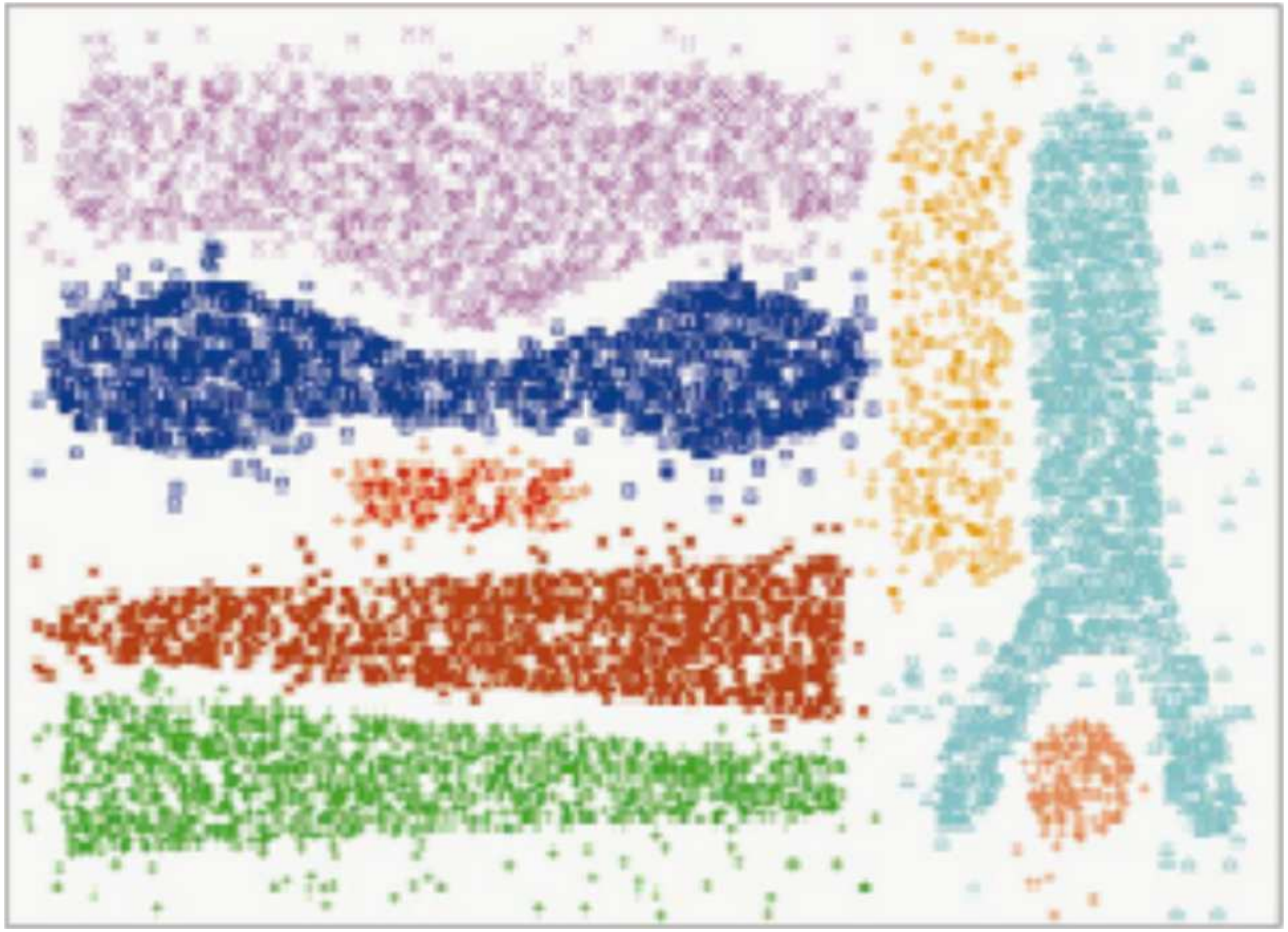
- graph cut algorithms (NCut, RatioCut, MinMaxCut)
- random walks theory (Markov clustering)

Spectral clustering: Toy example



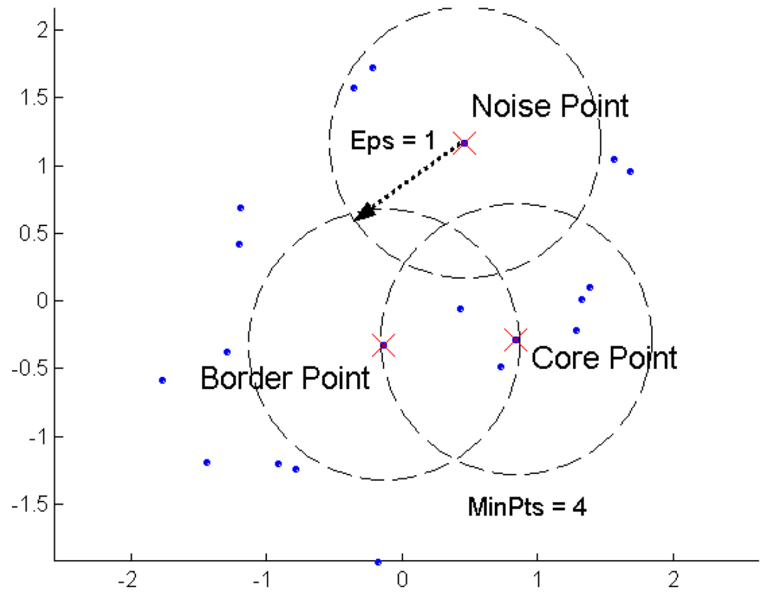
Density-based clustering in a nutshell

- Track density-connected points = neighborhood analysis
- arbitrary shaped clusters, robustness to noise, one pass over the data
- Typical algorithms: DBSCAN, OPTICS, DENCLUE

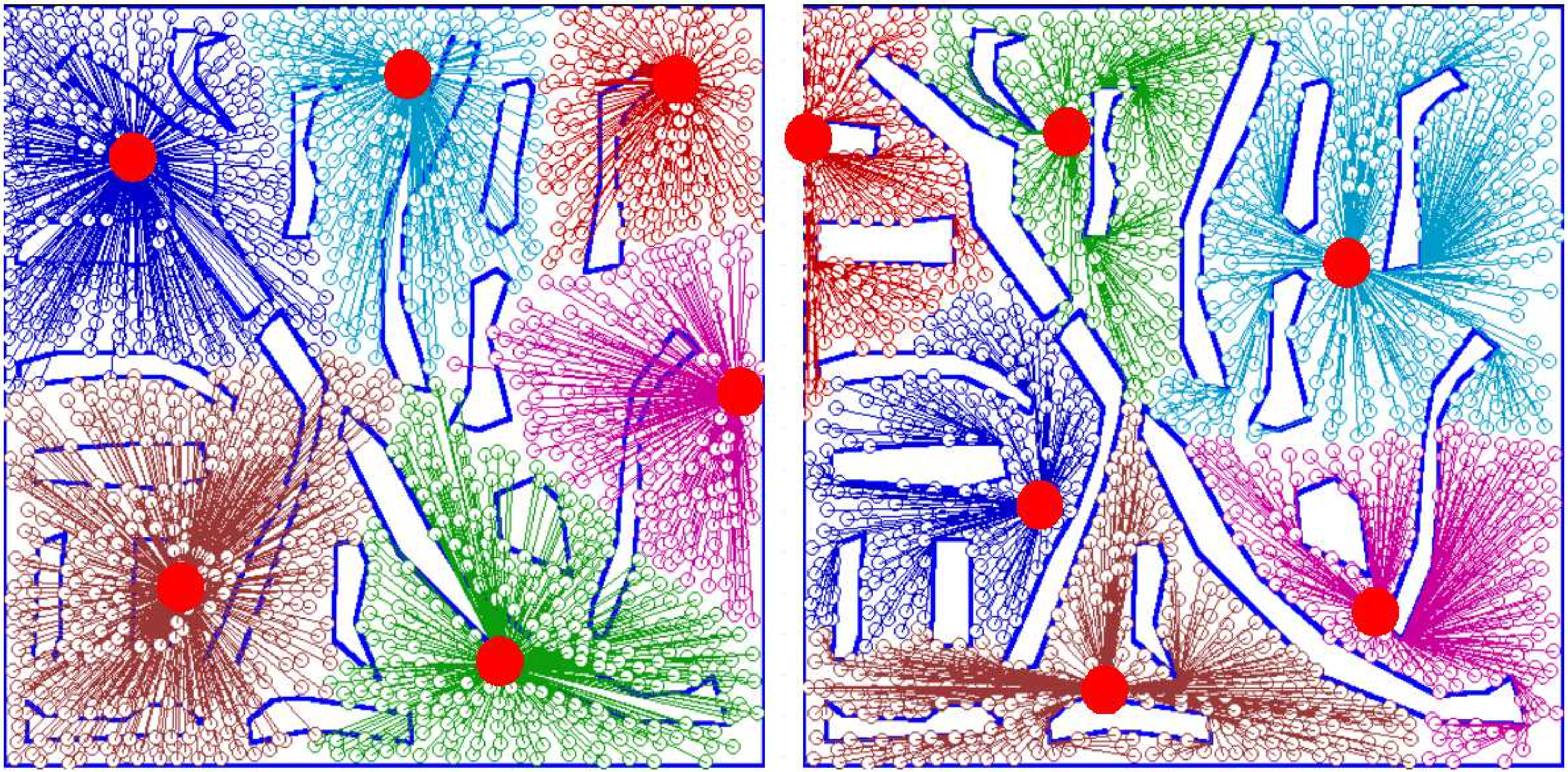


Density-based clustering in a nutshell (cont'd)

1. label points
   - core points: enough points around
   - border points: not enough points around but close to a core point
   - noise points: none of the above
2. eliminate noise points
3. create clusters from core points
4. assign border points to core clusters



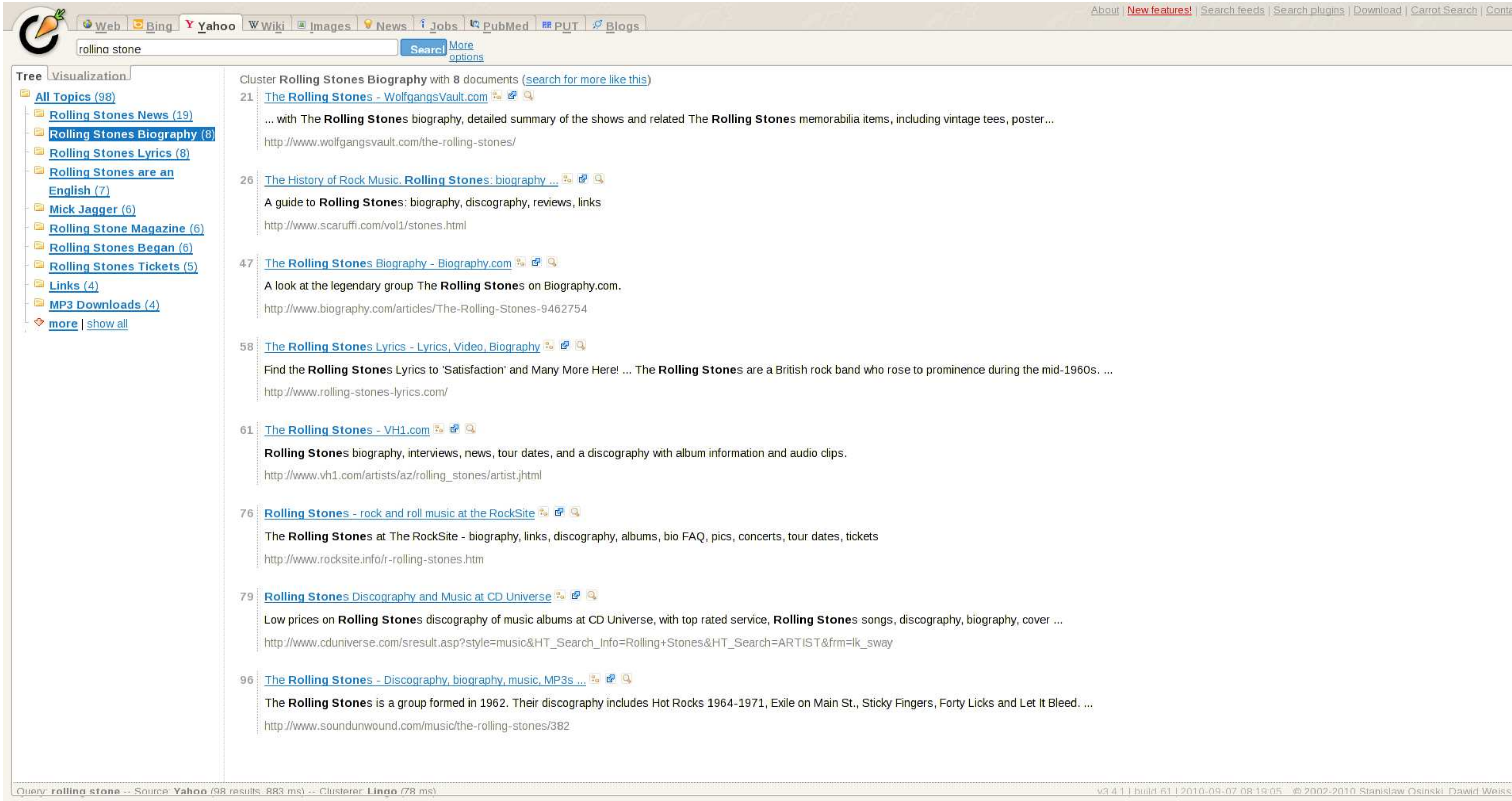
Clustering with obstacles



Clustering in high dimension space

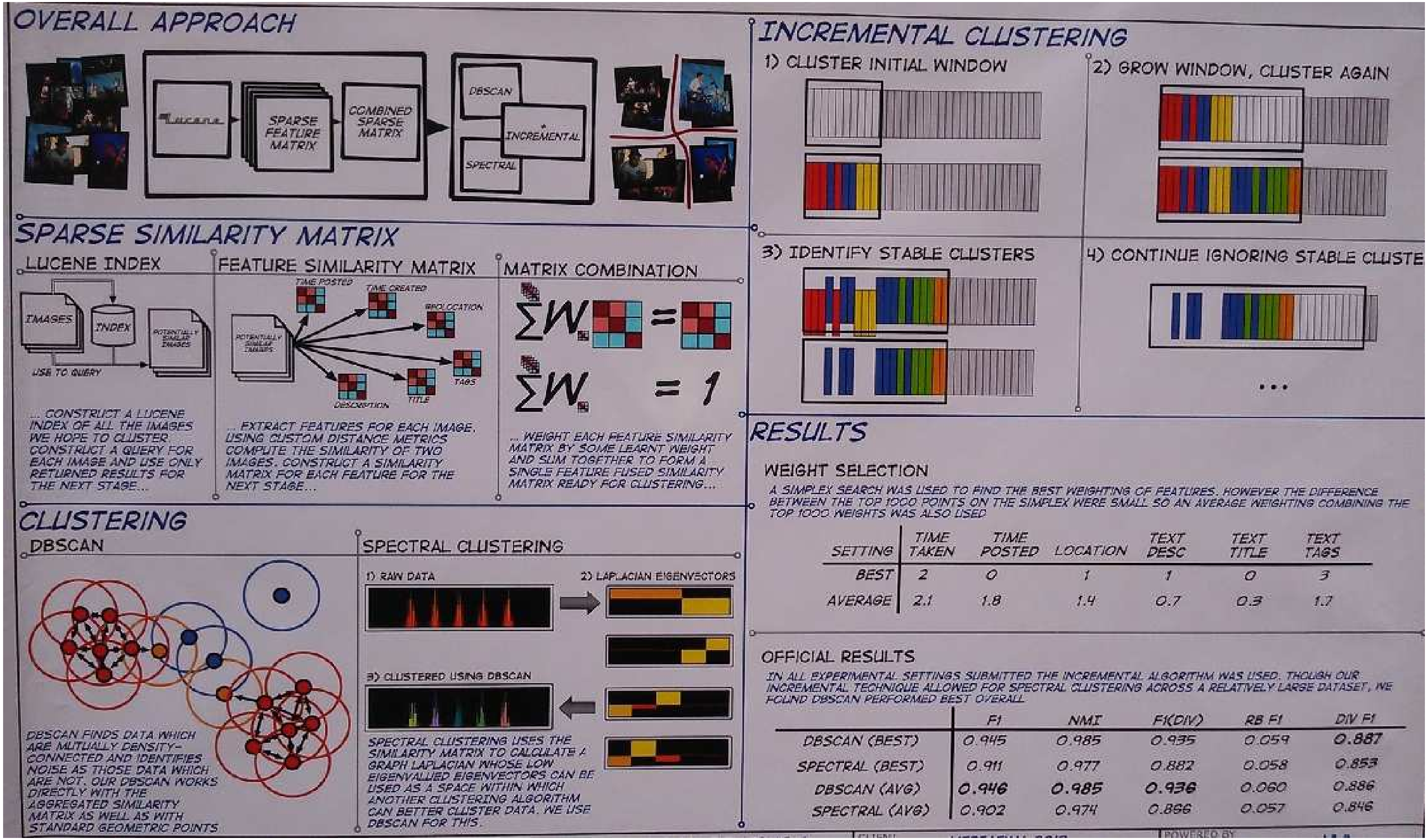
- many applications require high-dimension spaces: text documents, images, DNA data, etc.
- high dimensions raises new challenges
  - many irrelevant dimensions might mask clusters
  - distance measure becomes meaningless (curse of dimensionality)
  - cluster may exist only in some subspaces
- several workarounds
  - feature transformation
    - PCA/SVD if features are correlated/redundant
  - feature selection
    - select feature where nice clusters appear
  - subspace clustering
    - find clusters in all possible subspaces (CLIQUE, Proclus)

A fun (and practical) use of text clustering



http://search.carrot2.org/stable/search

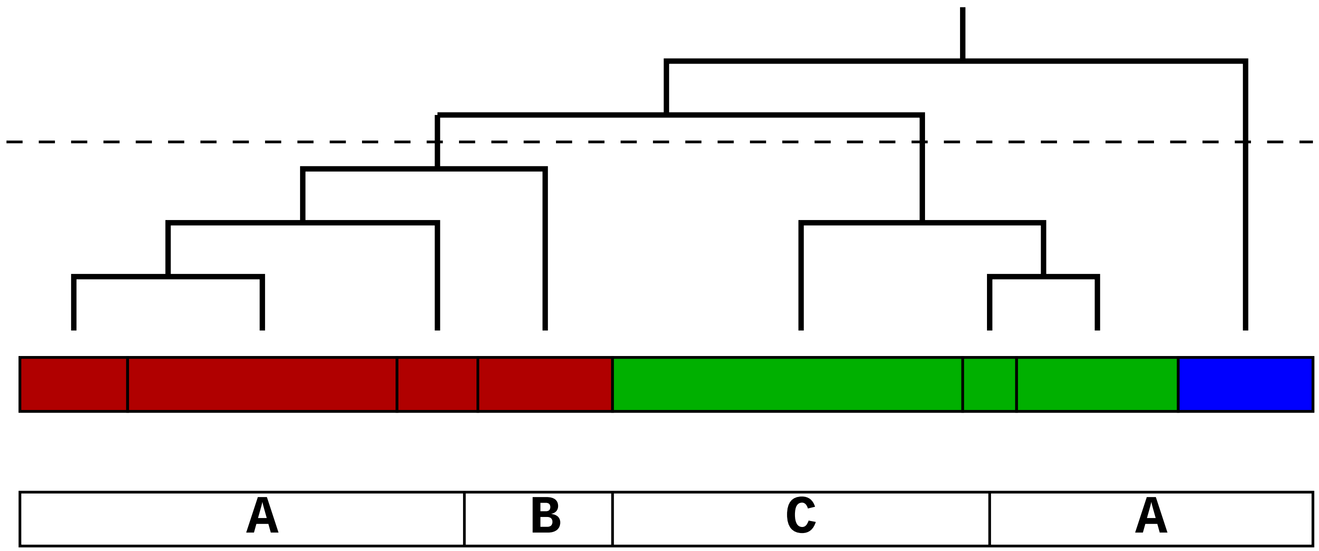
Another example mixing all you've seen



MediaEval 2013 Social Event Detection Task

About the evaluation of clustering

- subjective evaluation by inspecting clusters
- within cluster distortion (if a meaningful metric exists)
- any oritrary objective quality criterion (but none out of the shelf)
- special case with temporal segmentations



To get to know more ...

- So many textbooks (look for clustering and data mining)!
- So many resources on the Web (not in Wikipedia this time)
  - free code available everywhere
  - lot's of tutorials/lessons
    - Dr. HOI Chu's course:

      https://svn.mosuma.net/r4000/doc/course/ci6227/public/lectures/lecture07cluster.pdf
    - Prof. Jiawei Han courses: http://www.cs.uiuc.edu/ hanj