

# Course content

**Main idea: get to know most common techniques (and theory) used in ML together with (some) coding skills.**

Intro

- Basic notions (dataset, task...) + Regression and classification models (A+F)
- Model assessment (A)
- [
  - Unsupervised learning: dimensionality reduction (A)
  - Supervised learning: K-Nearest Neighbor, Trees (A)
- [
  - Kernels (F)
  - Artificial Neural Networks (F)
  - (Probabilistic formulation, recommender systems, RL) (L)

# Unsupervised Machine Learning

+

# What's Unsupervised Machine Learning

- **Unsupervised learning** is a type of algorithm that learns patterns from untagged data.
- Other levels of supervision:
  - Supervised Machine Learning: Data tagged by an expert.
  - Reinforcement Learning: Data untagged, but rules for performance evaluation are provided.
  - Semi-supervised Learning: Small portion of the data is tagged.

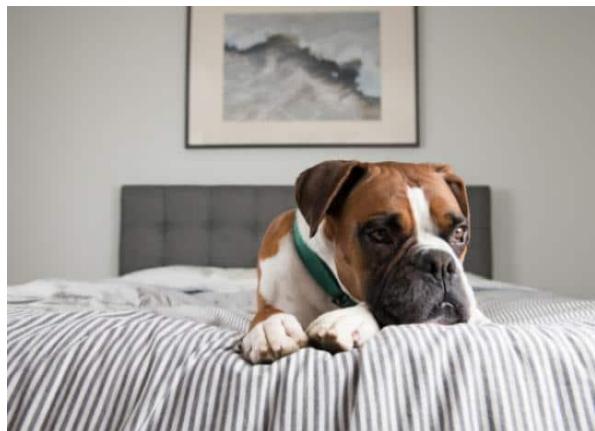
# Which patterns?

- In supervised learning we learn a conditional distribution  $p(y|x)$  while in Unsupervised Learning we learn directly properties of the underlying probability distribution:  $p(x)$ .
- Different properties (patterns) that we learn from  $p(x)$  correspond to different Unsupervised Learning task.

Learning from the structure of the data

# **SOME TASKS WITH UNSUPERVISED MACHINE LEARNING**

# A view from supervised learning



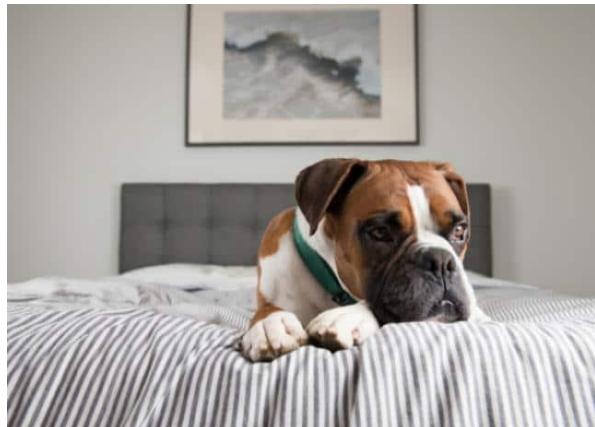
$$D = 1280 \times 960 = 1228800$$

Supervised  
ML

CAT

DOG

# A view from supervised learning



$$D = 1280 \times 960 = 1228800$$

Supervised  
ML

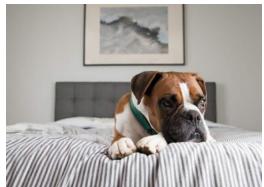
NATURE

HOME

# A view from supervised learning

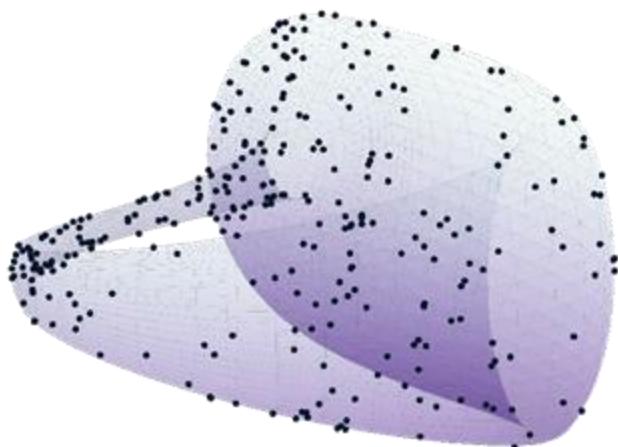
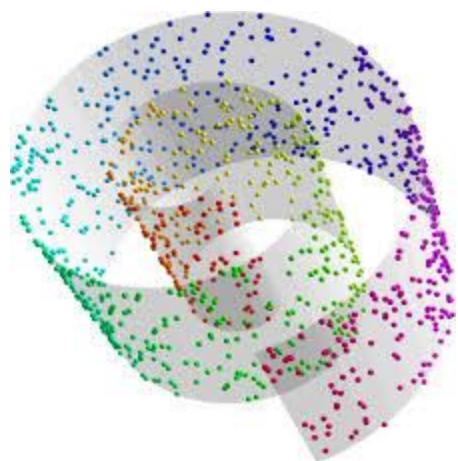
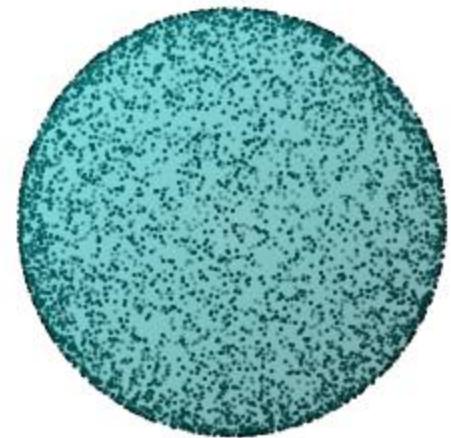
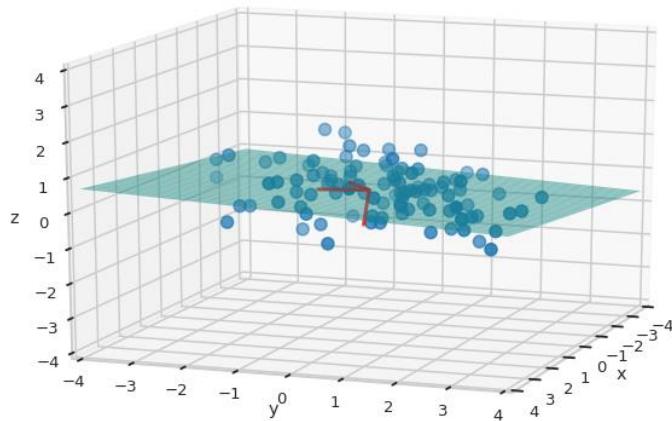
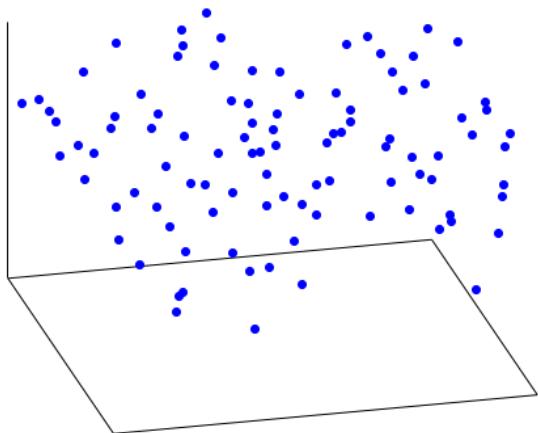


$$D = 1280 \times 960 = 1228800$$



The data lay in a manifold whose dimension is the total number of INDEPENDENT classification tasks that can be performed.

# MANIFOLDS



# Manifold Learning: a task of Unsupervised learning

Two questions can be addressed:

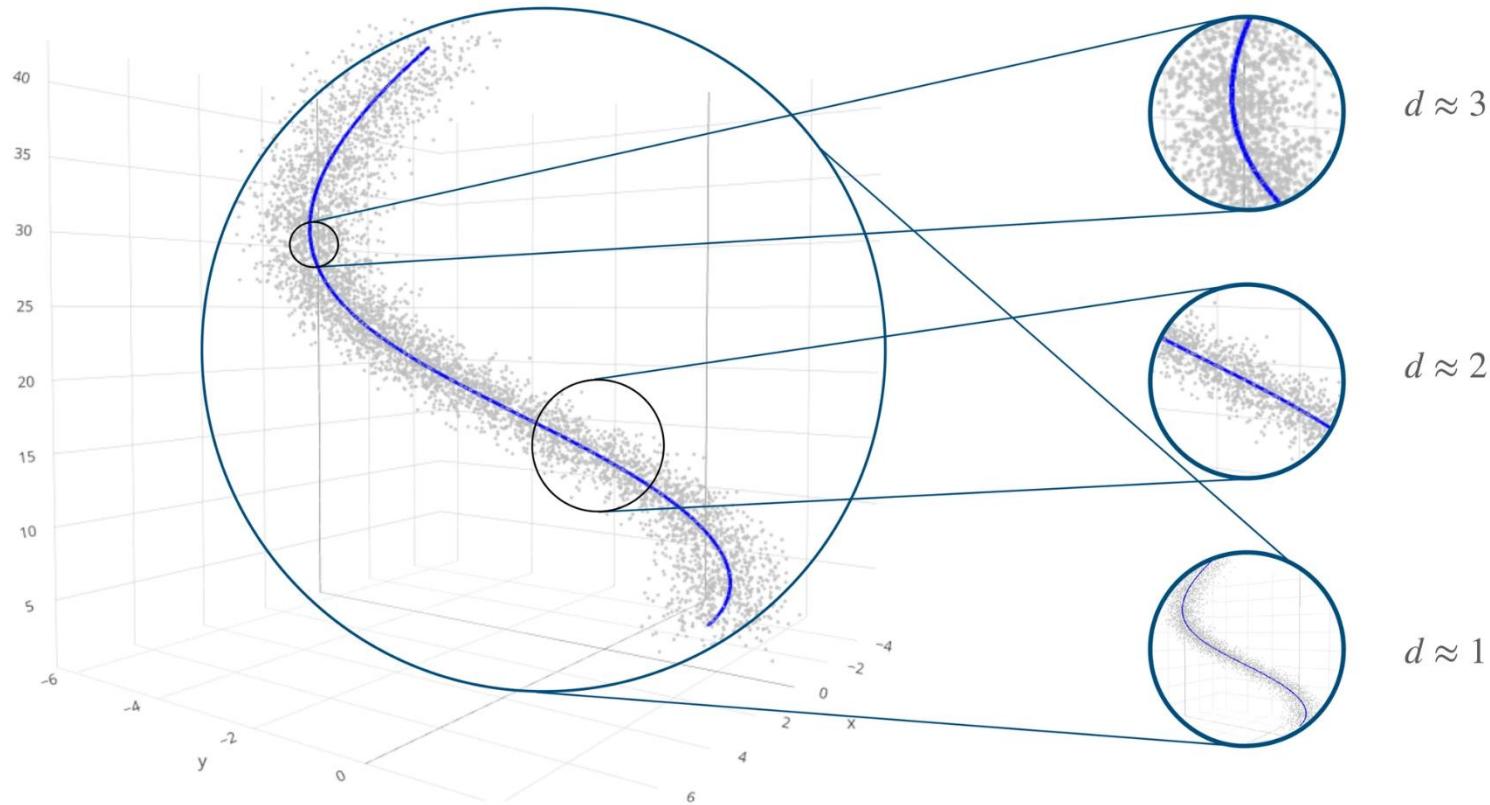
1. Which is the dimension of the manifold in which my data lives? (Intrinsic dimension  $D \rightarrow d$ )
2. Is there a way to *project* my data from  $\mathcal{R}^D$  into  $\mathcal{R}^d$ ?

# Some considerations about manifold learning

- In real cases, the manifold assumption is usually an approximation.

# Some considerations about manifold learning

- $d$  is a scale dependent quantity (!!!).



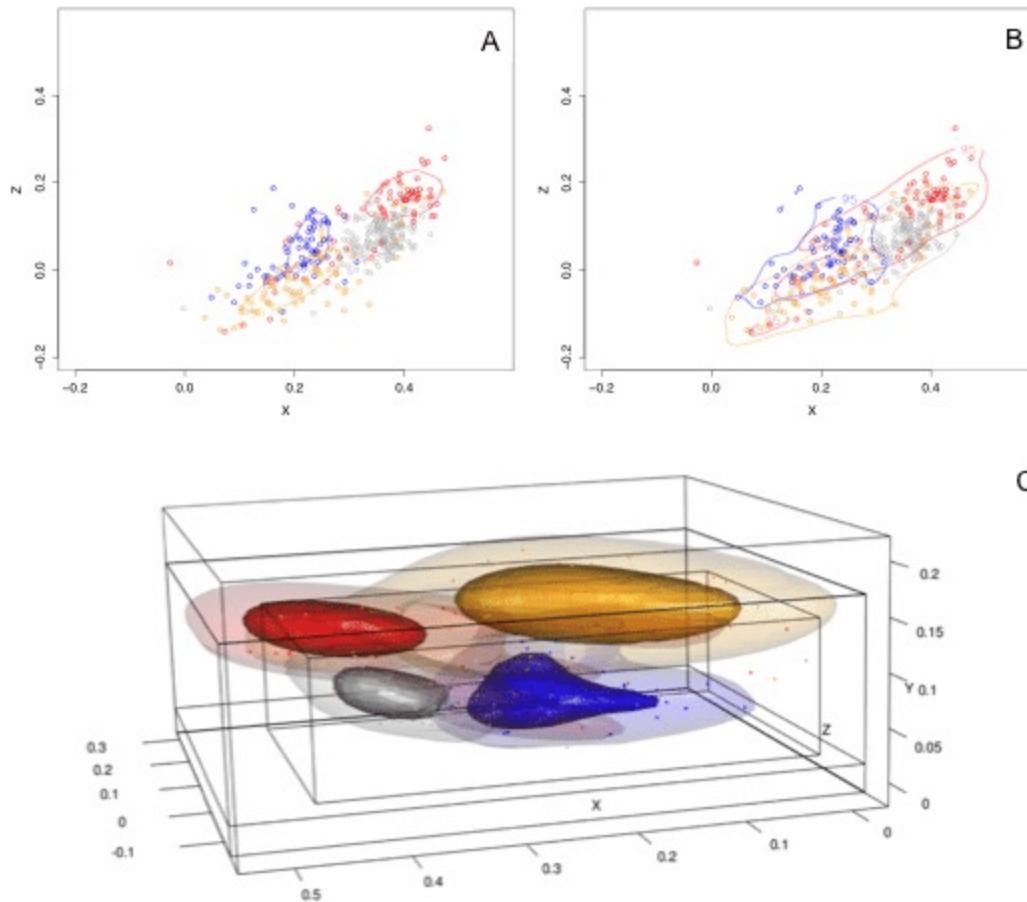
# Some considerations about manifold learning

- Is not always possible to find a closed form for the projection:

# How is my data distributed? Estimation of $p(x)$

# How is my data distributed?

## Estimation of $p(x)$



# The curse of dimensionality

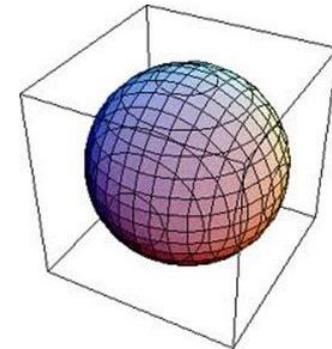
## Combinatorics

I have D binary variables. How many data points I would need in order to have *at least* one example of each variant?

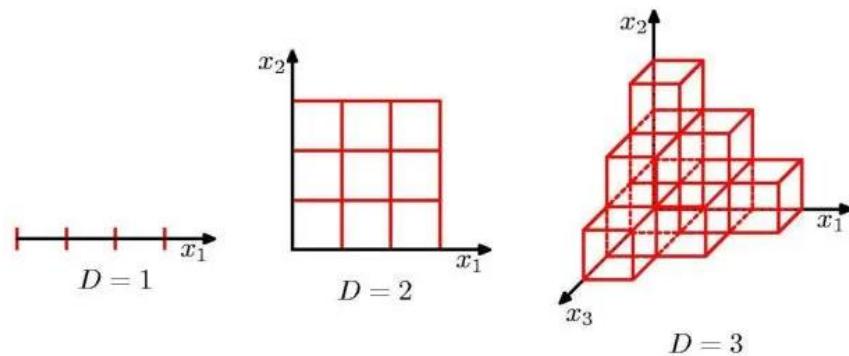
$$2^D$$

## Distance

$$\lim_{D \rightarrow \infty} \left[ \frac{V_{hs}}{V_{hc}} = \frac{\frac{2r^D \pi^{D/2}}{D\Gamma(D/2)}}{(2r)^D} = \frac{\pi^{D/2}}{D 2^{D-1} \Gamma(D/2)} \right] = 0$$



## Sampling



# Clustering

- Can I divide my data into groups?

# Clustering

- There exists groups in my data?

# Clustering

- Are there hierarchies in my data?

# Other tasks from Unsupervised Learning (not for this course)

- Anomaly detection (detect when  $x^i$  does not belong to  $p(x)$ )
- Generation of data: Generate new data points  $x^i$  that belong to the same distribution  $p(x)$  (Variational Autoencoders)
- Analysis of time dependent data (Markov State Models)

# Recap

- We will see algorithms that allow us to perform three main tasks that require Unsupervised Learning:
  - Manifold Learning (Intrinsic Dimension Estimation and Projection).
  - Density Estimation.
  - Clustering.



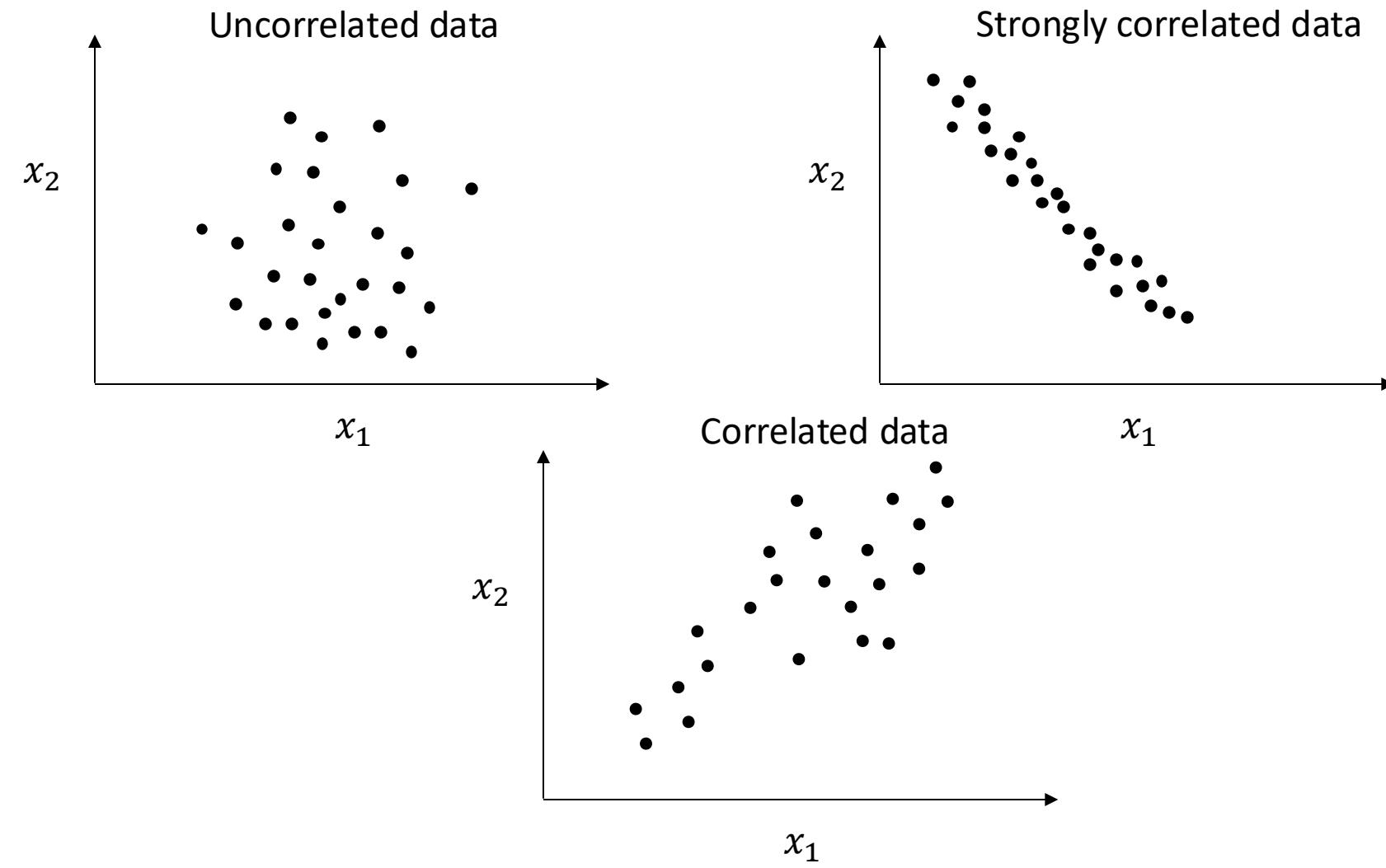
# Yesterday

- Definition of Unsupervised Learning and several types of supervision.
- Concepts of Intrinsic Dimension & Curse of Dimensionality.
- Tasks for Unsupervised Learning:
  - Manifold Learning
  - Density Estimation
  - Clustering

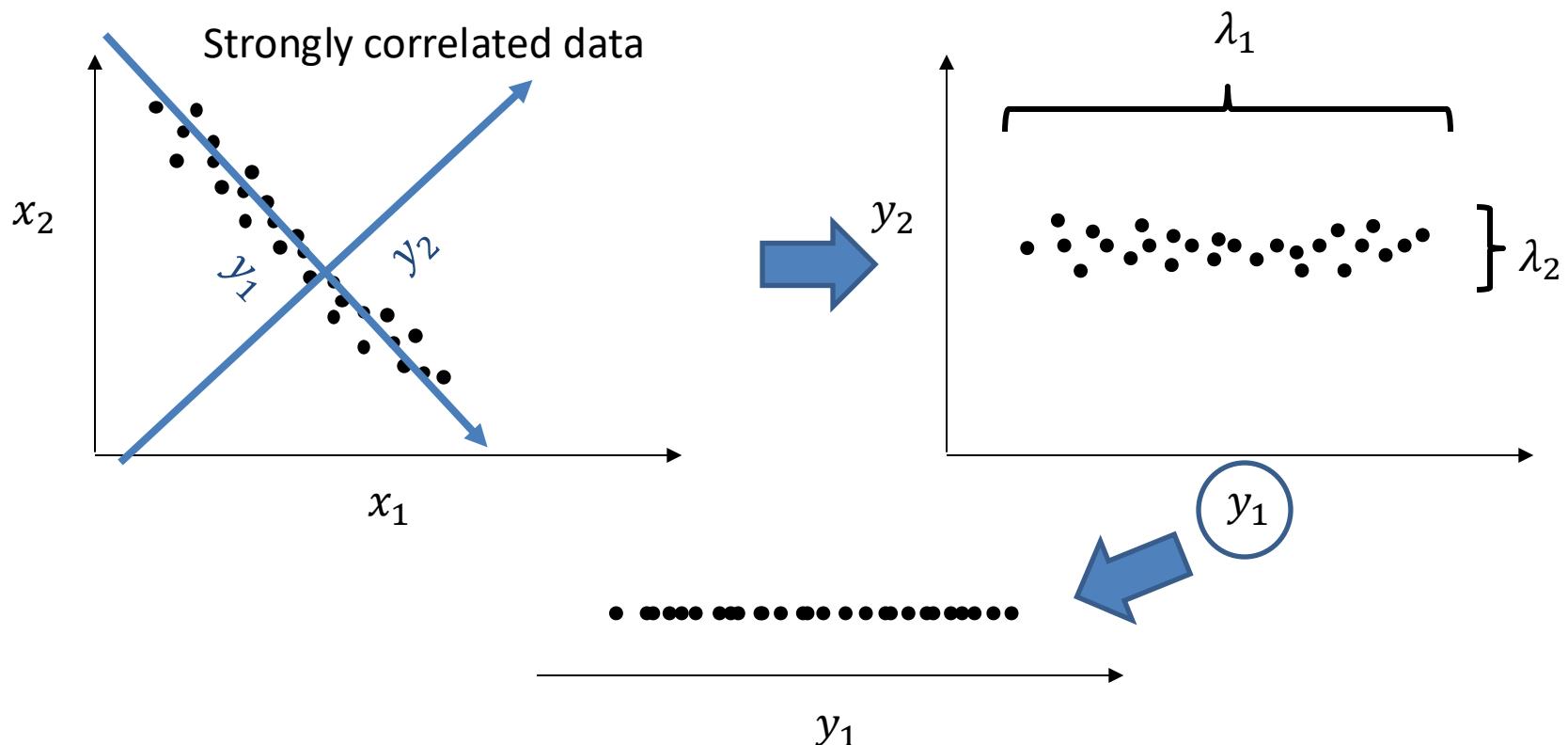
# Algorithms for Unsupervised Learning

- **Manifold Learning (Dim. Red. & ID est.):**
  - PCA
  - K-PCA
  - ISOMAP
  - t-SNE
  - **Autoencoders**
- **Density Estimation:**
  - Histograms
  - Kernel Density Estimation
  - k-Nearest Neighbor
  - **Generative Adversarial NN**
- **Clustering:**
  - k-means/c-means, kernel k-means, spectral clustering...
  - Hierarchical clustering
  - Density Based clustering
  - **Self Organizing Maps**

# STRONG CORRELATIONS REDUCE DIMENSIONALITY



# What is the task that we want to perform?



NOTE: We just need a rotation of the space that reduces (cancels) covariances

DIAGONALIZATION OF THE COVARIANCE MATRIX

# Key elements for deriving PCA

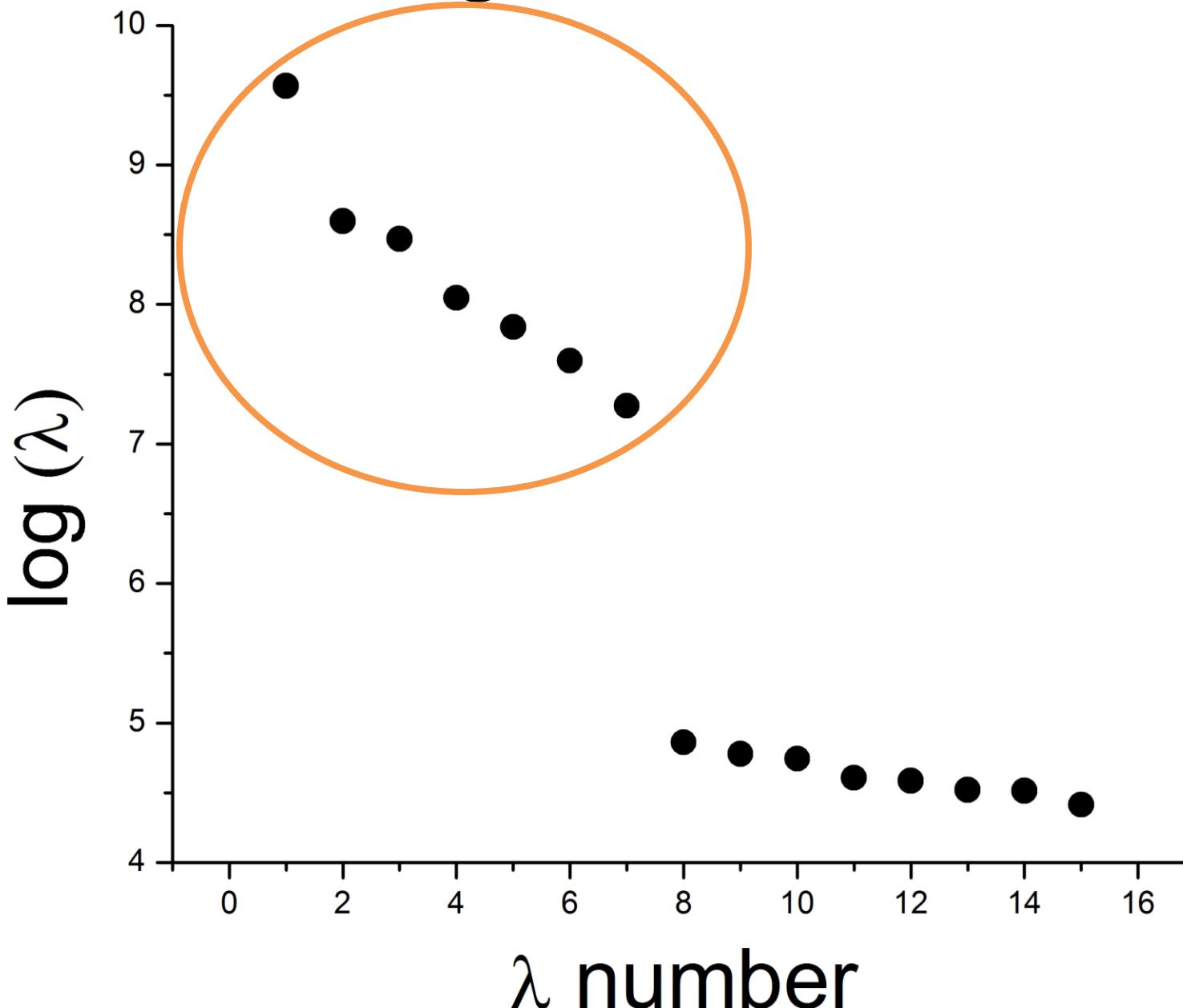
- We want a linear transformation of the form:  
 $y_i^{(l)} = \sum_j u_{ij} x_j^{(l)}$
- Minimizing covariances is equivalent to maximizing the variances (trace of the covariance matrix).
- We have to impose a constraint into the magnitude of  $u$  otherwise one can increase the variance just by multiplying it by an arbitrary number!. It is done by means of *Lagrange multipliers*.

# PCA. How to...

Given  $n$  observations  $\{x^{(1)}, x^{(2)} \dots, x^{(n)}\}$  of  $m$ -dimensional column vectors so  $x^{(l)} = (x_1^{(l)}, x_2^{(l)}, \dots, x_m^{(l)})$

1. Compute the mean vector  $\mu = \frac{1}{n} \sum_1^n x^{(l)}$
2. Compute the covariance matrix by MLE  $\mathbb{C} = \frac{1}{n} \sum_1^n (x^{(l)} - \mu)(x^{(l)} - \mu)^T$
3. Compute the eigenvalue/eigenvector pairs  $(\lambda_i, u_i)$  of  $\mathbb{C}$  with  $\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq \dots \geq \lambda_m$  (Diagonalize/SVD) and sort.
4. Choose the number of dimensions (d) in which project the data
5. Project by  $y_i^{(l)} = \sum_j u_{ij} x_j^{(l)}$  where now  $i$  goes from 1 to d.

# Choosing the number of PC's



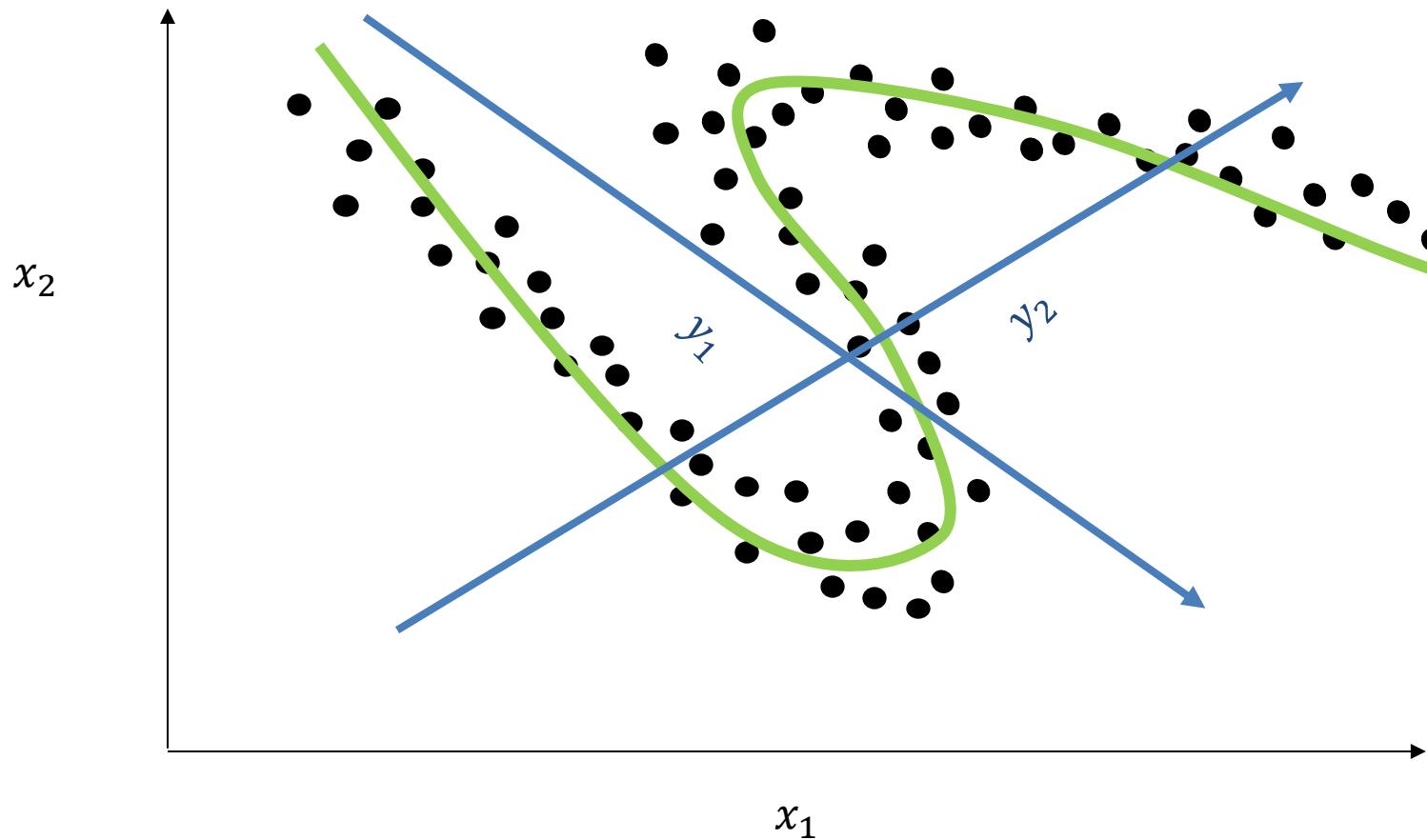
# PCA in matrix form

- Center the data matrix  $X$
- Compute the covariance matrix  $C = X^T X$
- Compute the eigenvalues and eigenvectors of the covariance matrix  $C$  and sort them according with decreasing order of the eigenvalues
- Arrange the eigenvectors as columns of the matrix  $U$
- Compute the principal components as  $Y = X \cdot U$
- Retain only  $d$  components of  $Y$

# PCA problems

- Poorly suited for non-linear transformations

# Addressing the non-linear problem

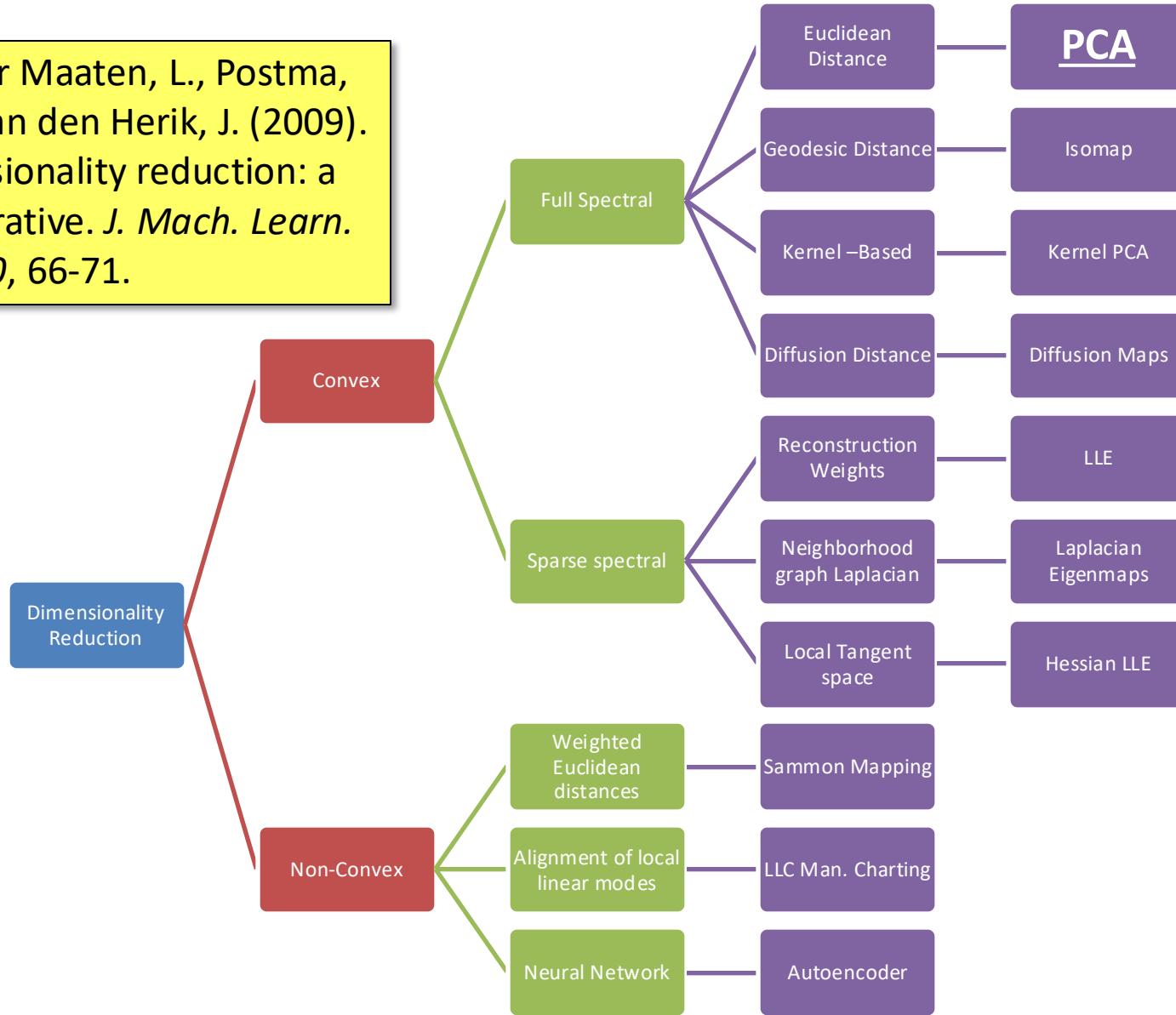


# PCA problems

- Poorly suited for non-linear transformations
- Not able of capturing invariances
- By using the covariance along the whole dataset, is poor suited for problems not well described by this parameter.
- Not scale invariant
- Dominated by large pairwise distances

# Beyond PCA

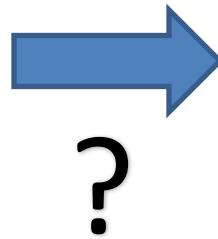
Van Der Maaten, L., Postma, E., & Van den Herik, J. (2009). Dimensionality reduction: a comparative. *J. Mach. Learn. Res.*, 10, 66-71.



# Why working with distances?

My data:

- 1 AACDPGGGAD
- 2 CCDPGGADA
- 3 CCCCCCCCCC

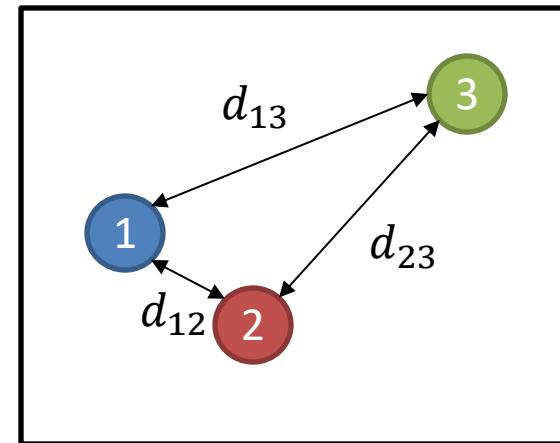


Look at them pairwise

$$\begin{array}{ll} 1 \text{ AACDPGGGAD*} & S_{12} = \frac{7}{11} \\ 2 \text{ *CCDP*GGADA} & \end{array}$$

$$\begin{array}{ll} 2 \text{ CCDPGGADA*} & S_{23} = \frac{2}{10} \\ 3 \text{ CCCCCCCCCC} & d_{ij} = 1 - S_{ij} \end{array}$$

$$\begin{array}{ll} 1 \text{ AACDPGGGAD} & \\ 3 \text{ CCCCCCCCCC*} & S_{13} = \frac{1}{10} \end{array}$$



- Similarities and distances are much more flexible than simple feature representation.
- Each type of data has its own optimal distance.

# Similarity and Distances

- Clustering tries to separate data “*naturally*”, in such a way that *similar* elements lay in the same cluster while *dissimilar* elements belong to a different one
- *Similarity* ( $S_{ij}$ ) is a pairwise function of the features of the elements  $i$  and  $j$ .
- In terms of space, it can be thought that similar elements are near while dissimilar are far. So many times it is useful to talk about “distances between elements” ( $D_{ij}$ )
- Its definition depends on the nature of the features

# Similarity and Distances

almost the same but...

A (metric) distance must accomplish:

1. Symmetry:  $d(x, y) = d(y, x)$
2. Non-negativity:  $d(x, y) \geq 0$
3. Identity of indiscernibles:  $d(x, y) = 0 \Leftrightarrow x = y$
4. Triangle inequality:  $d(x, z) + d(z, y) \geq d(x, y)$

We have to take this into account for some clustering algorithms

# Quantitative Features: *Metric Distances*

- Minkowski distance:

$$d_{ij} = \left( \sum_{l=1}^d |x_{il} - x_{jl}|^p \right)^{1/p}$$

- Special cases are:

- Euclidean ( $p=2$ )
- City-block ( $p=1$ )
- Sup ( $p \rightarrow \infty$ ) . Eqv to  $d_{ij} = \max_l |x_{il} - x_{jl}|$

- Mahalanobis distance

$$d_{ij} = (x_i - x_j)^T \mathbb{C}^{-1} (x_i - x_j)$$

Invariant with respect to any non-singular linear transformation of the coordinates.  $\mathbb{C}$  is the covariance matrix.

# Quantitative Features: *Not metric distances*

- Pearson correlation:

$$d_{ij} = \frac{1 - r_{ij}}{2}; r_{ij} = \frac{\sum_{k=1}^m (x_{k,i} - \bar{x}_i)(x_{k,j} - \bar{x}_j)}{\sqrt{\sum_{k=1}^m (x_{k,i} - \bar{x}_i)^2} \sqrt{\sum_{k=1}^m (x_{k,j} - \bar{x}_j)^2}}$$

- Cosine similarity:

$$S_{ij} = \frac{{x_i}^T \cdot x_j}{\|x_j\| \|x_i\|}$$

# Qualitative Features

- Jaccard similarity:

$$S_{ij} = \frac{|i \cap j|}{|i \cup j|}$$

$i = \boxed{1}000\boxed{1}00\boxed{1} j = 0\boxed{1}00\boxed{1}0\boxed{1}1; S_{ij} = \frac{2}{5}$

- Hamming distance:

$$D_{ij} = |i \cup j| - |i \cap j|$$

i.e. minimum number of changes that you need to turn  $i$  in  $j$ .

# More complicated distances

- Working in the metric can extremely simplify the clustering work.
- A good metric can dramatically improve the performance of an algorithm.
- However, usually they need to compute a simplest distance as starting point.
- Example: Geodesic distance

# Focusing on preserving the distances: Multidimensional Scaling

$\Delta^{ij}$  Distance in the ambient space

$\delta^{ij}$  Distance in the projected space

Stress: Measure of the discrepancy between both distances.  
We try to minimize it.

# Focusing on preserving the distances: Multidimensional Scaling

$$S = \sum_{i,j} \left( (\Delta^{ij})^2 - (\delta^{ij})^2 \right)^2$$

**Classical MDS:** Formally equivalent to PCA when using Euclidean Distance.  
Analytical minimization.

$$S = \frac{\sum_{i,j} \left( (\Delta^{ij}) - (\delta^{ij}) \right)^2}{\sum_{i,j} \delta^{ij}}$$

**Metric MDS:** Preserves the relative importance of the distances.  
Numerical minimization.

$$S = \frac{\sum_{i,j} \left( g(\Delta^{ij}) - f(\delta^{ij}) \right)^2}{\sum_{i,j} f(\delta^{ij})^2}$$

**Non-metric MDS:** You choose what to preserve. Numerical minimization.

# Classical MDS

- 1.- Obtain the Gram (inner product) matrix

$$\hat{G}^{ij} = -\frac{1}{2} \left( \Delta^{ij} + \frac{1}{N^2} \sum_{i,j} \Delta^{ij} - \frac{1}{N} \left( \sum_i \Delta^{ij} + \sum_j \Delta^{ij} \right) \right)$$

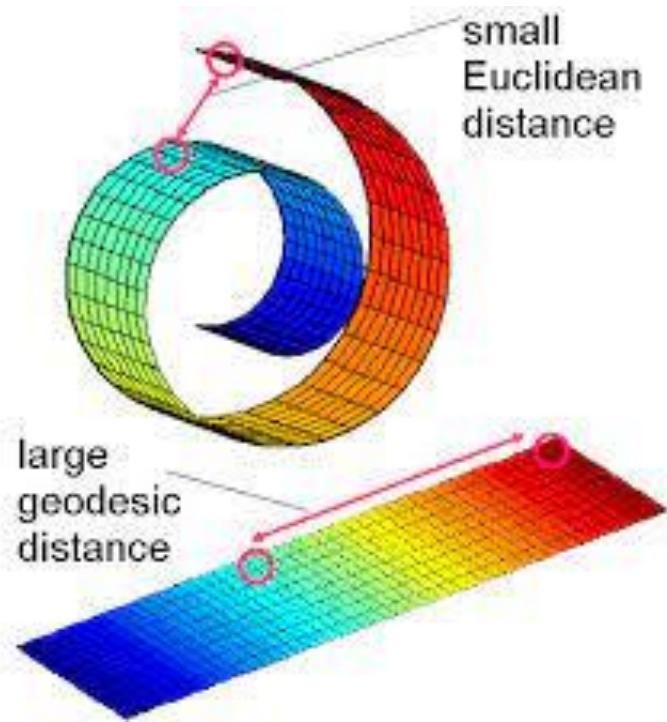
*This is called double centering.*

- 2.- Diagonalize it

- 3.- From the spectrum, decide  $d$

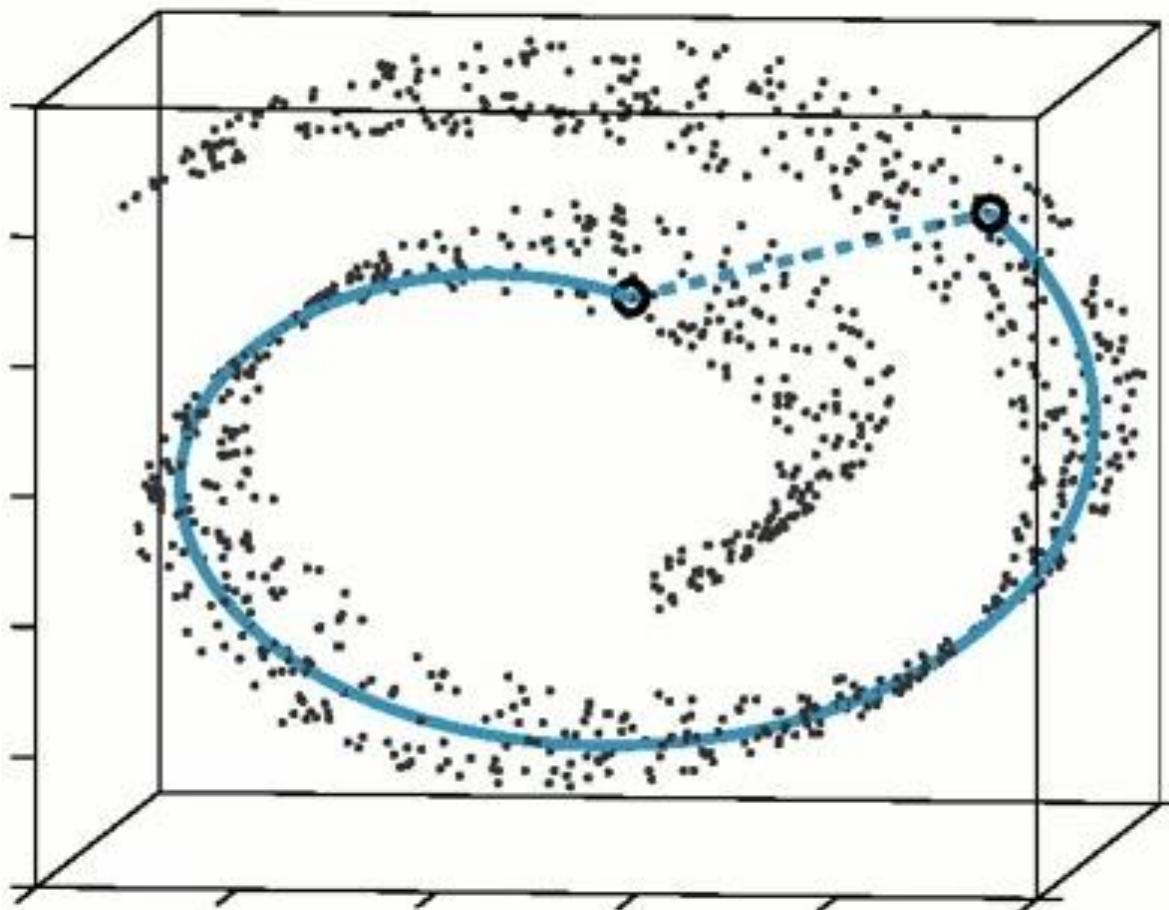
Note that this procedure does not require  $\Delta^{ij}$  being computed as the Euclidean distance!!!

# ISOMAP



Uses classical MDS setting  
 $\Delta^{ij}$  equal to the geodesic  
distance instead of the  
Euclidean distance.

# Geodesic distance



# Geodesic distance

- Determine the neighbors.
  - All points in a fixed radius.
  - K nearest neighbors
- Construct a neighborhood graph.
  - Each point is connected to the other if it is a K nearest neighbor.
  - Edge Length equals the Euclidean distance
- Compute the shortest paths between two nodes

# Kernel PCA

- You will see kernel PCA in detail next semester.
- The idea is, that through classical MDS, we have a formulation of PCA that relies on the inner-product.
- This allows to use the kernel trick (use of a kernel to function that parameterizes this product).
- Extends PCA to non-linear projections by expanding the dimensionality (in the feature space).

# t-SNE/UMAP

- These methods aim to reproduce the probabilities of neighborhood between data points.
- Optimize a loss function that measures the overlap between these probabilities in the ambient and projection spaces.
- Generally excellent performance for visualization, take care when using them for pre-processing.
- Random initialization, not always reproducible results.

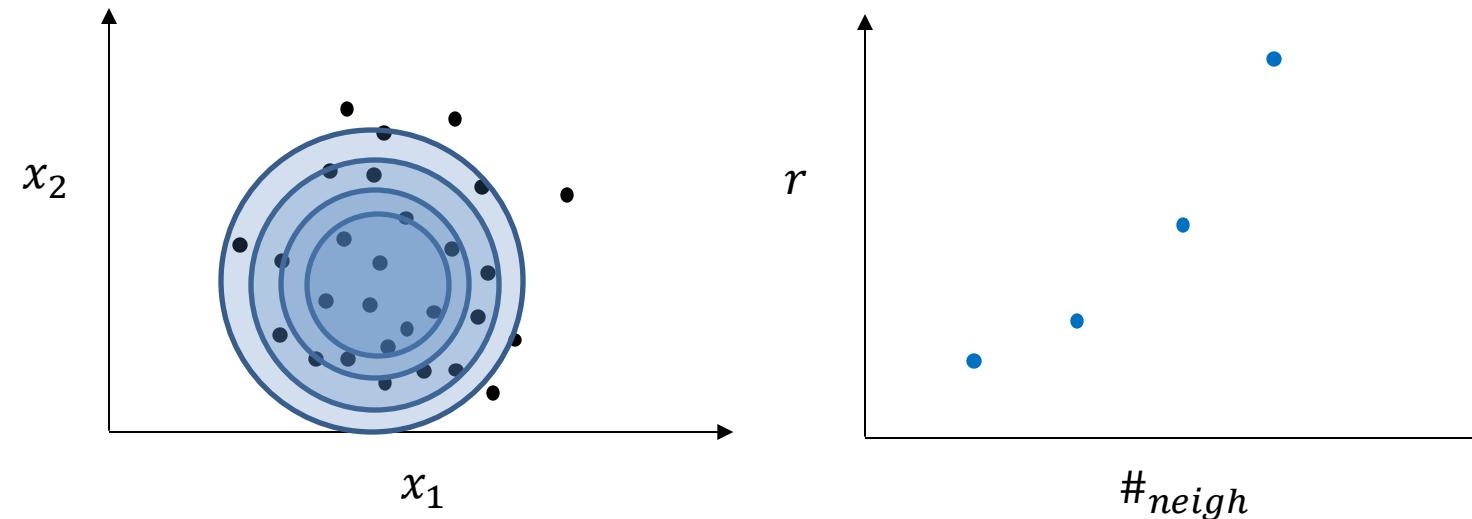
# t-SNE

[https://www.youtube.com/watch?app=desktop&v=o\\_cAOa5fMhE](https://www.youtube.com/watch?app=desktop&v=o_cAOa5fMhE)

# Directly estimation of the Intrinsic dimension

Minimum number of parameters required to describe the data while minimizing the information loss.

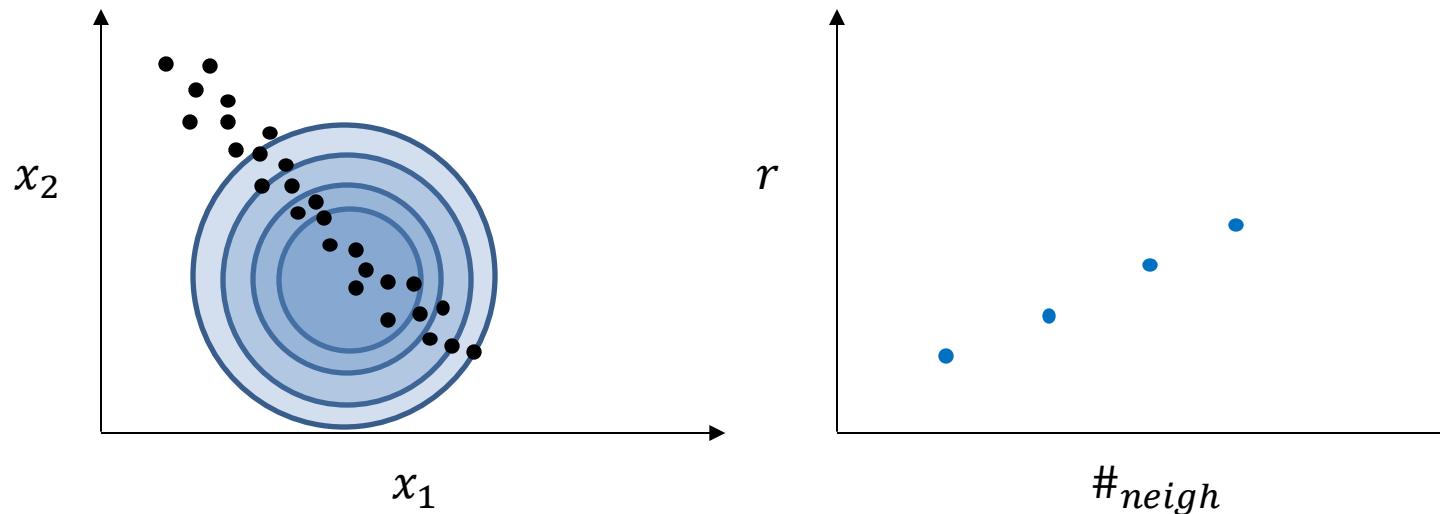
Classical approach  $\#_{neigh} = \rho r^d$



# Directly estimation of the Intrinsic dimension

Minimum number of parameters required to describe the data while minimizing the information loss.

Classical approach  $\#_{neigh} = \rho r^d$



# Directly estimation of the Intrinsic dimension

In practice:

$$\#_{neigh} = \rho r^d \longrightarrow \log(\#_{neigh}) = \log \rho + d \log r \quad \text{Linear fit}$$

Only true at constant density:

Modern methods try to  
disentangle both measures

## TWO-NN

$$\mu_i = \frac{r_{i,2}}{r_{i,1}}$$

$$P(\mu|\rho) = P(\mu) = \frac{d}{\mu^{1+d}}$$



# First lecture on UL

- Definition of Unsupervised Learning and several types of supervision.
- Concepts of Intrinsic Dimension & Curse of Dimensionality.
- Tasks for Unsupervised Learning:
  - Manifold Learning
  - Density Estimation
  - Clustering
- PCA. Concept, Derivation and flaws.

# Second lecture on UL

- Distances and similarities:
  - Mathematical properties of distances
  - Several distances types according to the data type
- Beyond PCA:
  - Multidimensional Scaling
  - Kernel PCA
  - ISOMAP (MDS with geodesic distance)
  - t-SNE

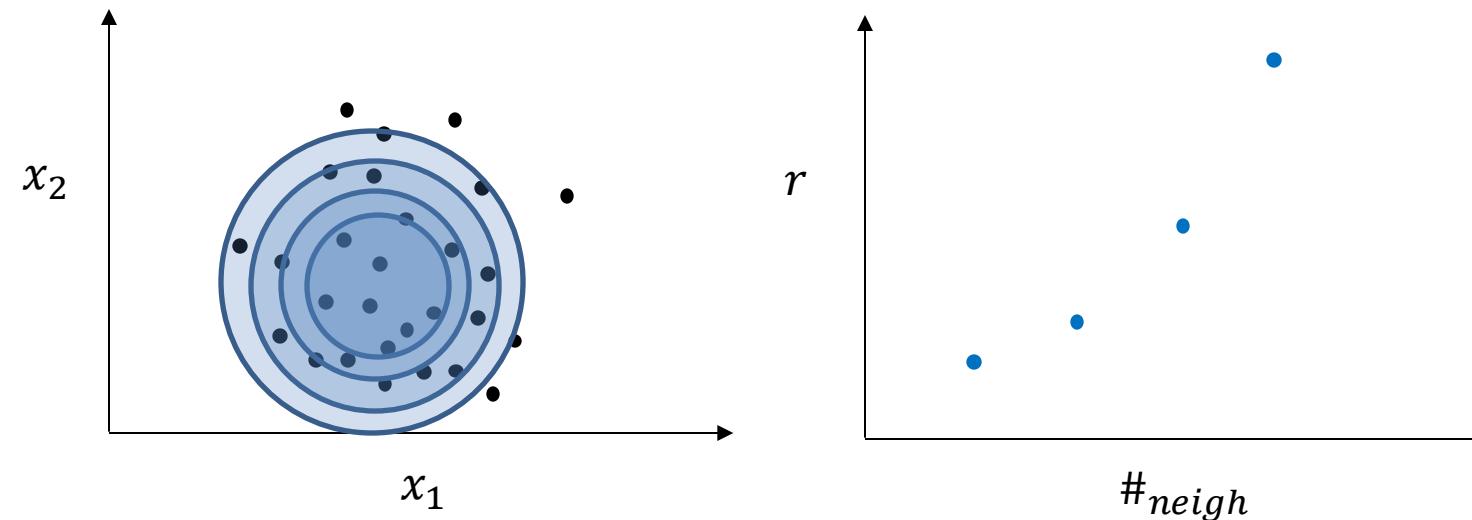
# Algorithms for Unsupervised Learning

- **Manifold Learning (Dim. Red. & ID est.):**
  - PCA
  - K-PCA
  - ISOMAP
  - t-SNE & UMAP
  - Two-NN (ID estimation)
  - Autoencoders
- **Density Estimation:**
  - Histograms
  - Kernel Density Estimation
  - k-Nearest Neighbor
  - Generative Adversarial NN
- **Clustering:**
  - k-means/c-means, kernel k-means, spectral clustering...
  - Hierarchical clustering
  - Density Based clustering
  - Self Organizing Maps

# Directly estimation of the Intrinsic dimension

Minimum number of parameters required to describe the data while minimizing the information loss.

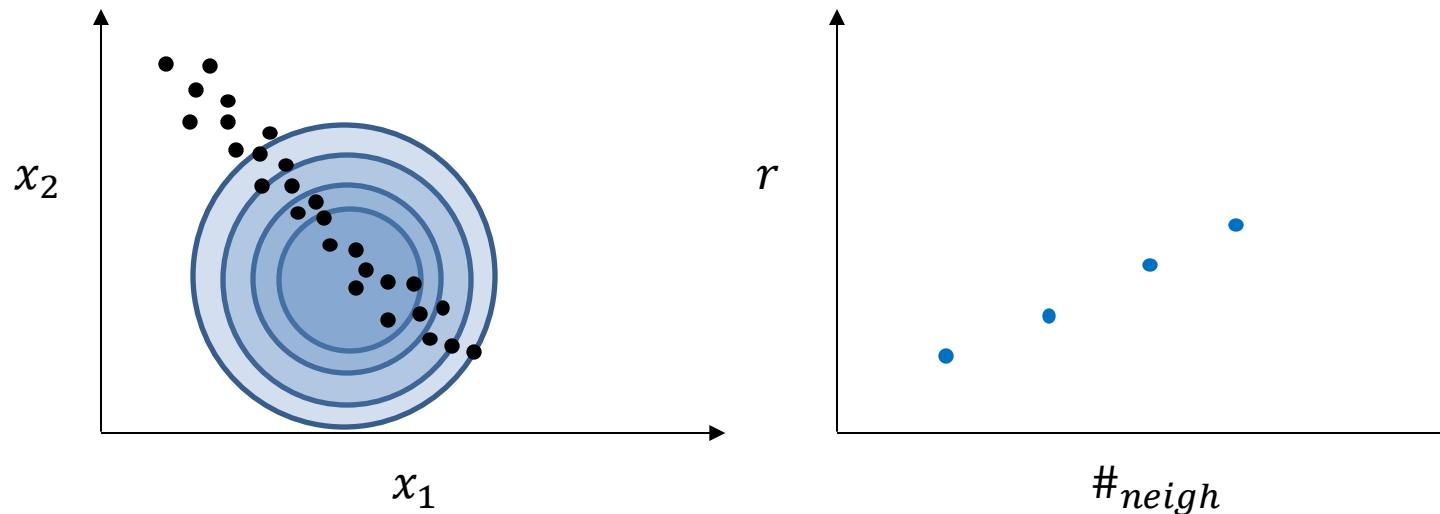
Classical approach  $\#_{neigh} = \rho r^d$



# Directly estimation of the Intrinsic dimension

Minimum number of parameters required to describe the data while minimizing the information loss.

Classical approach  $\#_{neigh} = \rho r^d$



# Directly estimation of the Intrinsic dimension

In practice:

$$\#_{neigh} = \rho r^d \longrightarrow \log(\#_{neigh}) = \log \rho + d \log r \quad \text{Linear fit}$$

Only true at constant density:

Modern methods try to  
disentangle both measures

## TWO-NN

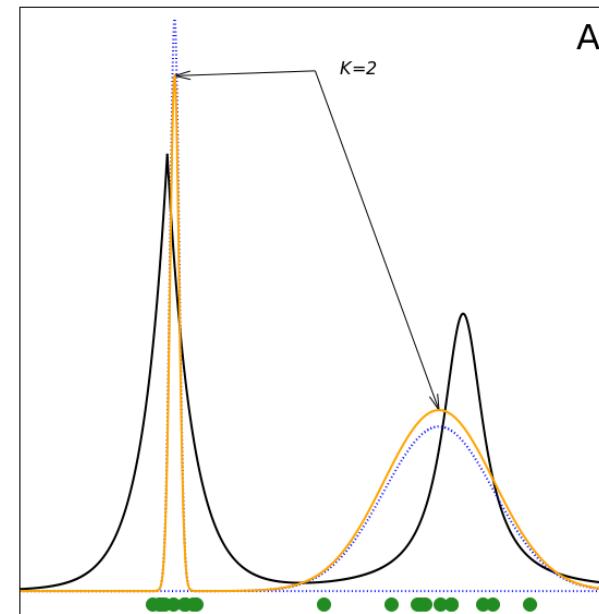
$$\mu_i = \frac{r_{i,2}}{r_{i,1}}$$

$$P(\mu|\rho) = P(\mu) = \frac{d}{\mu^{1+d}}$$

# **DENSITY ESTIMATION**

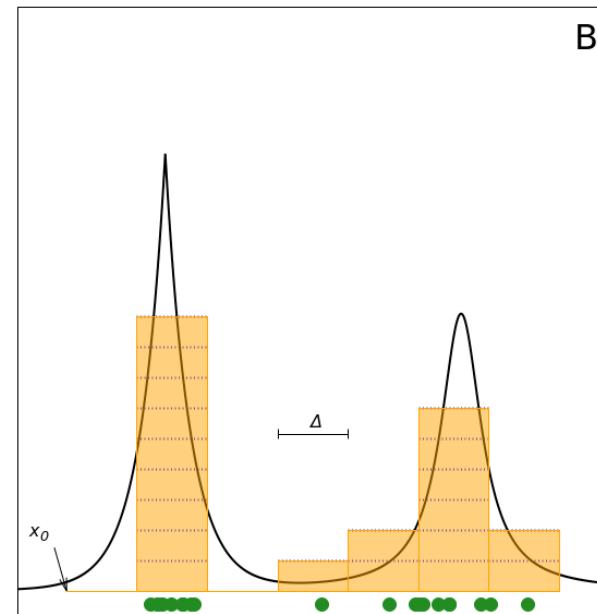
# Gaussian mixture models

- Parametric approach  
(We enforce a functional form to  $p(x)$ )
- $p(x) = \sum_{i=1}^K w_i e^{-\left(\frac{x-\mu_i}{s_i}\right)^2}$
- The only free parameter is  $K$



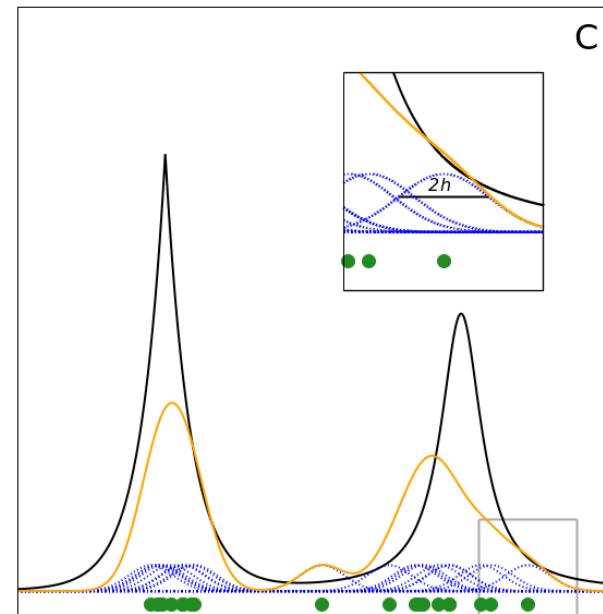
# Histograms

- Non-parametric approach  
(We don't enforce a functional form to  $p(x)$ )
- Count the number of points within a binwidth  $\Delta$
- Two parameters:  $\Delta$  &  $x_0$
- Poor performance at  $d>2-3$

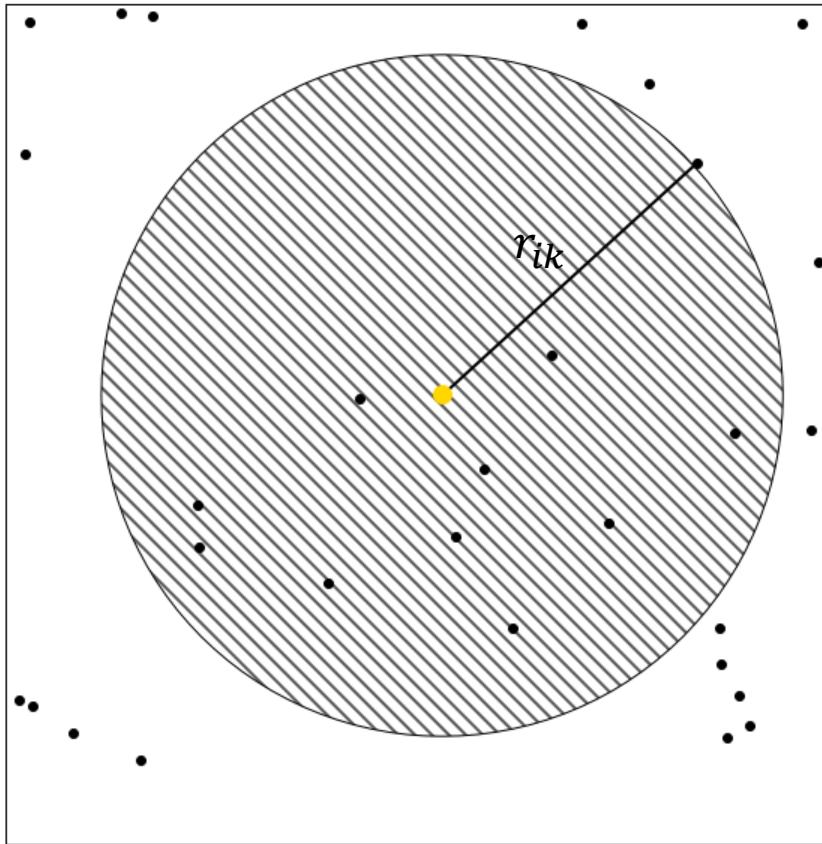


# Kernel density estimation

- Non-parametric approach (We don't enforce a functional form to  $p(x)$ )
- $p(x) = \sum_i \frac{1}{h\sqrt{2\pi}} e^{-\left(\frac{x-x_i}{h}\right)^2}$
- Puts a gaussian in the top of each point, then sum.
- Single parameter:  $h$
- The kernel could not be Gaussian.



# The k-Nearest Neighbor estimator



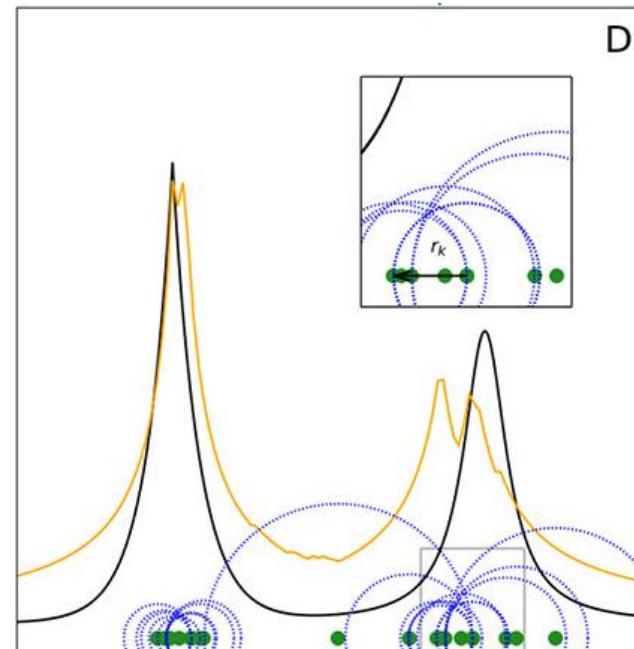
$$\hat{\rho}_i = \frac{k/N}{V_k} = \frac{k/N}{r_{ik}^d}$$

$$\sigma_{\hat{\rho}_i} = \frac{\hat{\rho}_i}{\sqrt{k}}$$

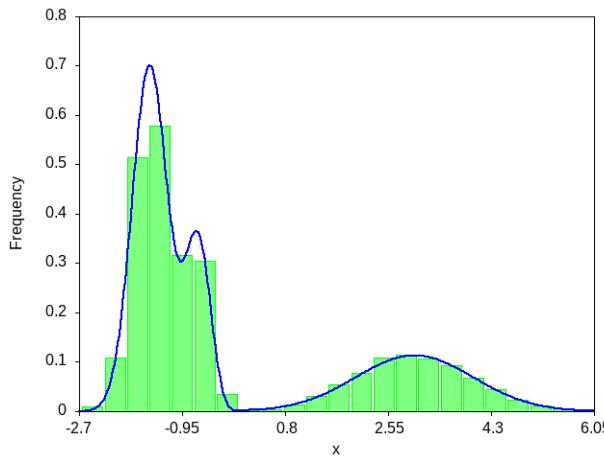
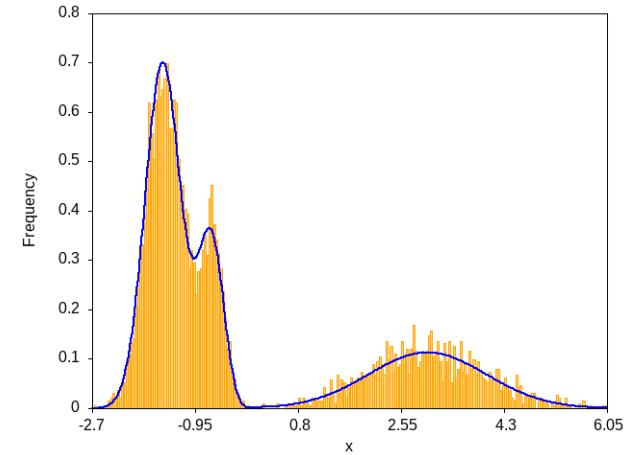
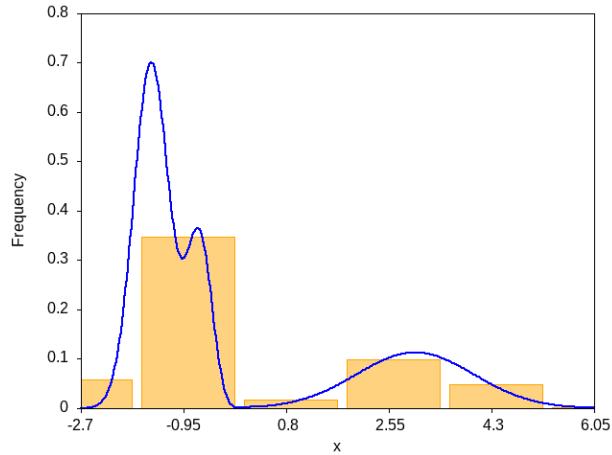
Derived under constant density conditions!

# K-Nearest neighbor density estimation

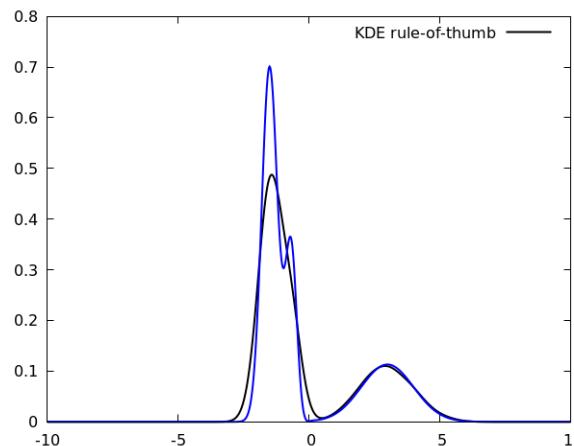
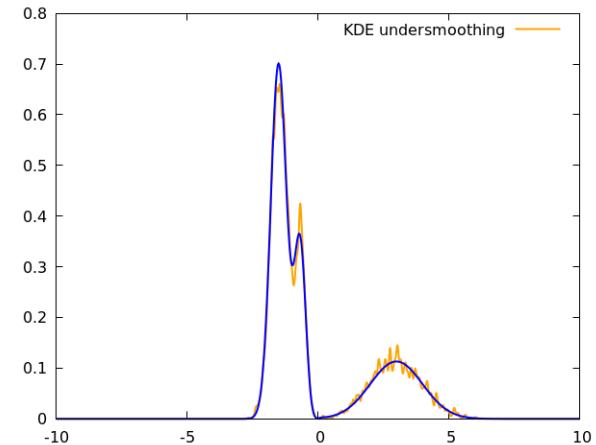
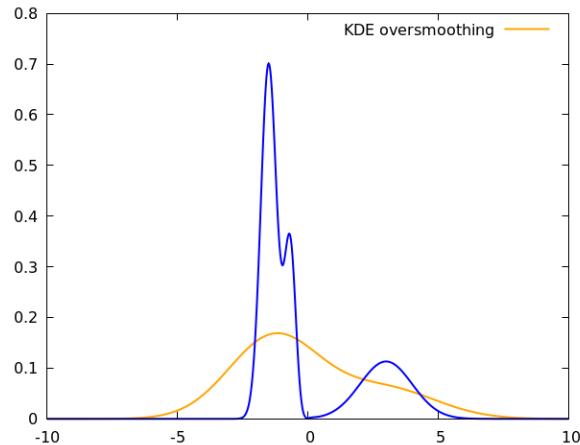
- Non-parametric approach (We don't enforce a functional form to  $p(x)$ )
- $p(x) = \frac{k}{Nr_k^d}$
- The density at a point is the inverse of  $k$  times the volume occupied by its  $k$  nearest neighbors.
- Single parameter:  $k$



# Bias-Variance tradeoff applied to density estimation (over and under smoothing)



# Bias-Variance tradeoff applied to density estimation (over and under smoothing)



# Some “rules” for setting the “optimal” parameters

- Histograms (Freedman-Diaconis rule):

$$\Delta \approx 2 \frac{IQR}{\sqrt[3]{N}}$$

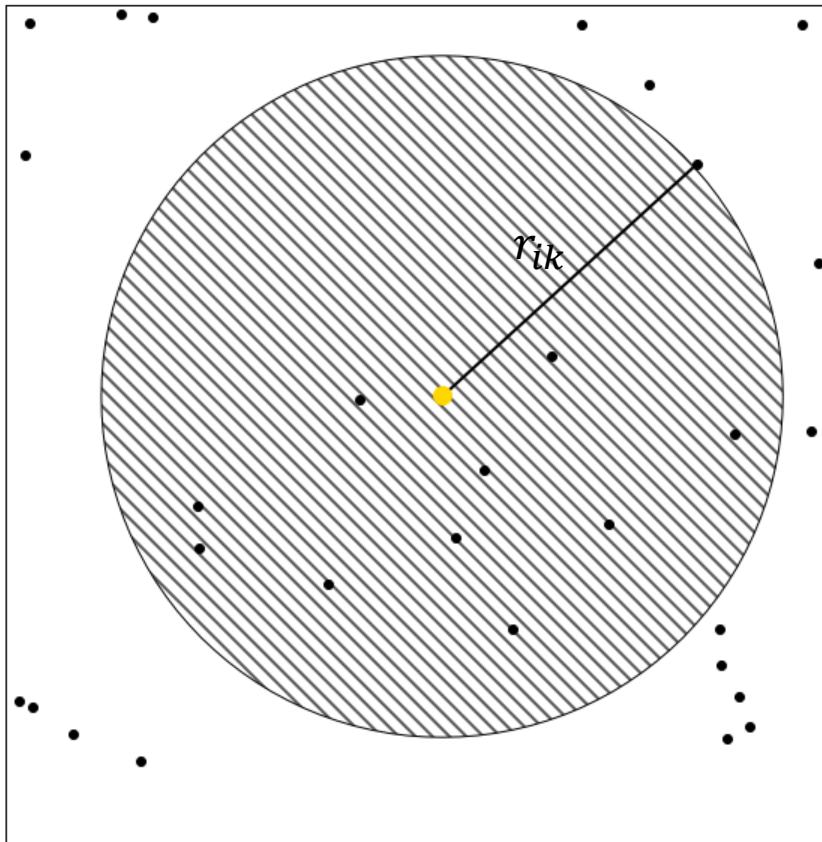
- KDE (Scott’s rule):

$$h = \frac{1}{N^{1/d+4}}$$

- k-NN:

$$k \approx N^{\frac{4}{4+d}}$$

# The k-Nearest Neighbor estimator



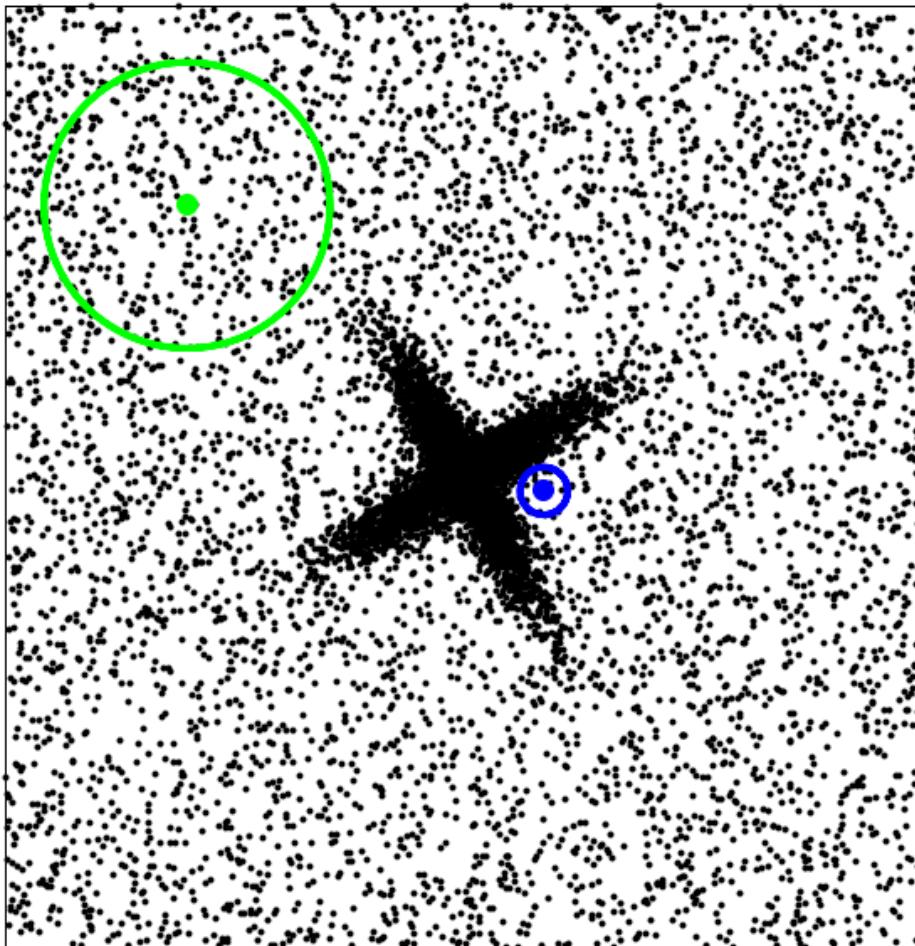
$$\hat{\rho}_i = \frac{k/N}{V_k} = \frac{k/N}{r_{ik}^d}$$

$$\sigma_{\hat{\rho}_i} = \frac{\hat{\rho}_i}{\sqrt{k}}$$

Derived under constant density conditions!

# Ideally a point adaptive method will

## *The bias-variance tradeoff*

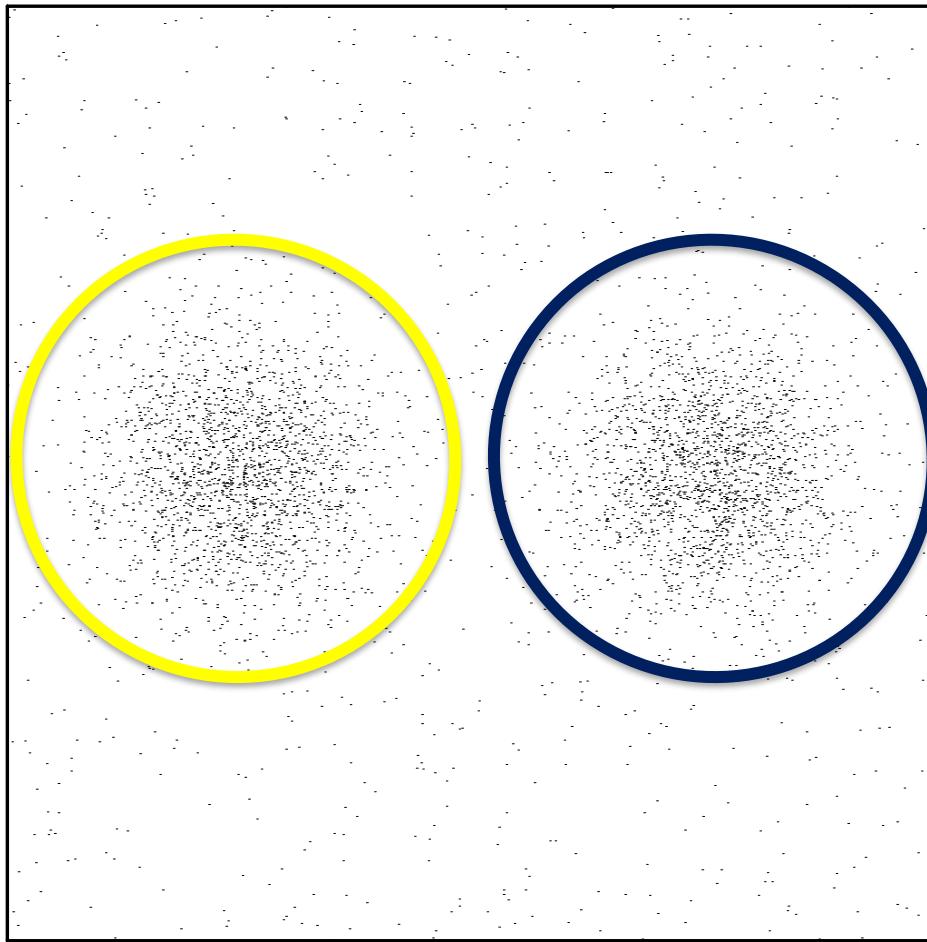


- Small rad: Big variance
- Big rad: Big bias  
(Error due to variations in the density )

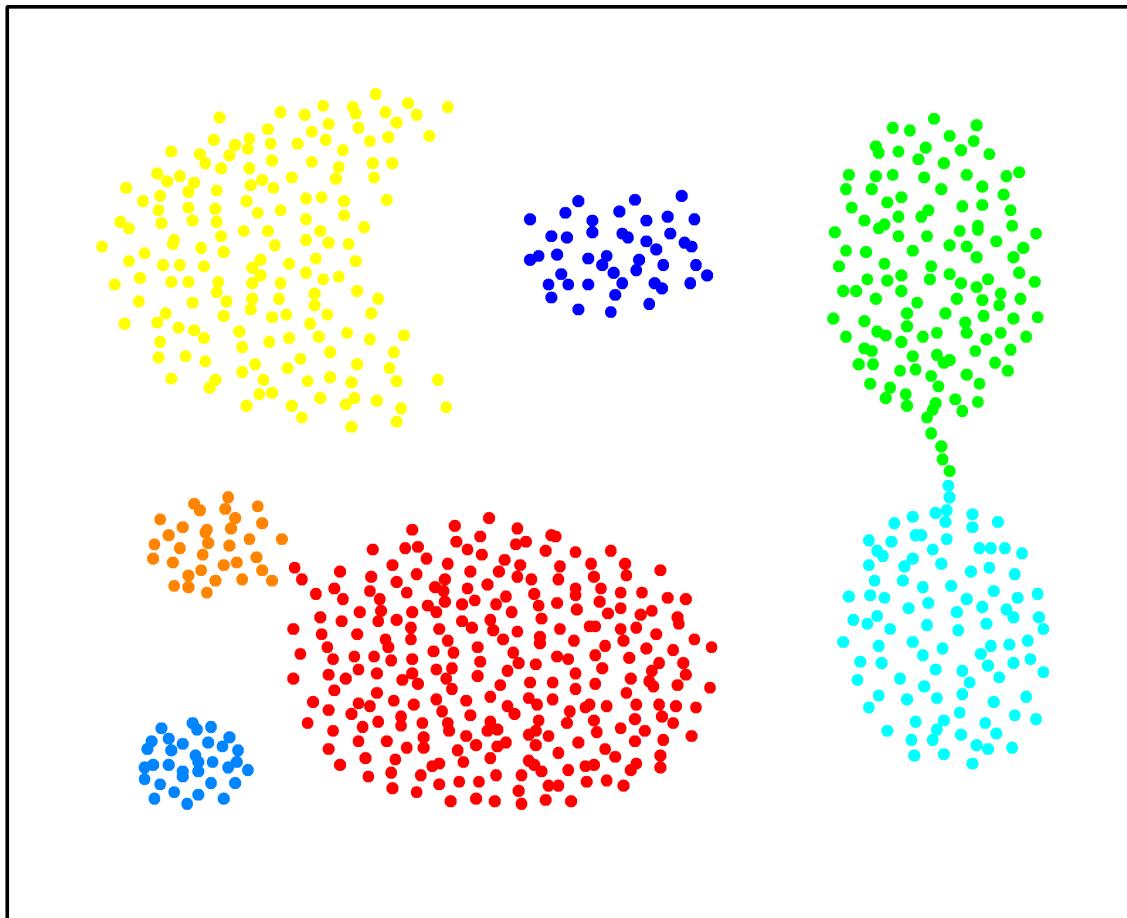
# Algorithms for Unsupervised Learning

- **Manifold Learning (Dim. Red. & ID est.):**
  - PCA
  - K-PCA
  - ISOMAP
  - t-SNE
  - Autoencoders
- **Density Estimation:**
  - Histograms
  - Kernel Density Estimation
  - k-Nearest Neighbor
  - Generative Adversarial NN
- **Clustering:**
  - k-means/c-means, kernel k-means, spectral clustering...
  - Hierarchical clustering
  - Density Based clustering
  - Self Organizing Maps

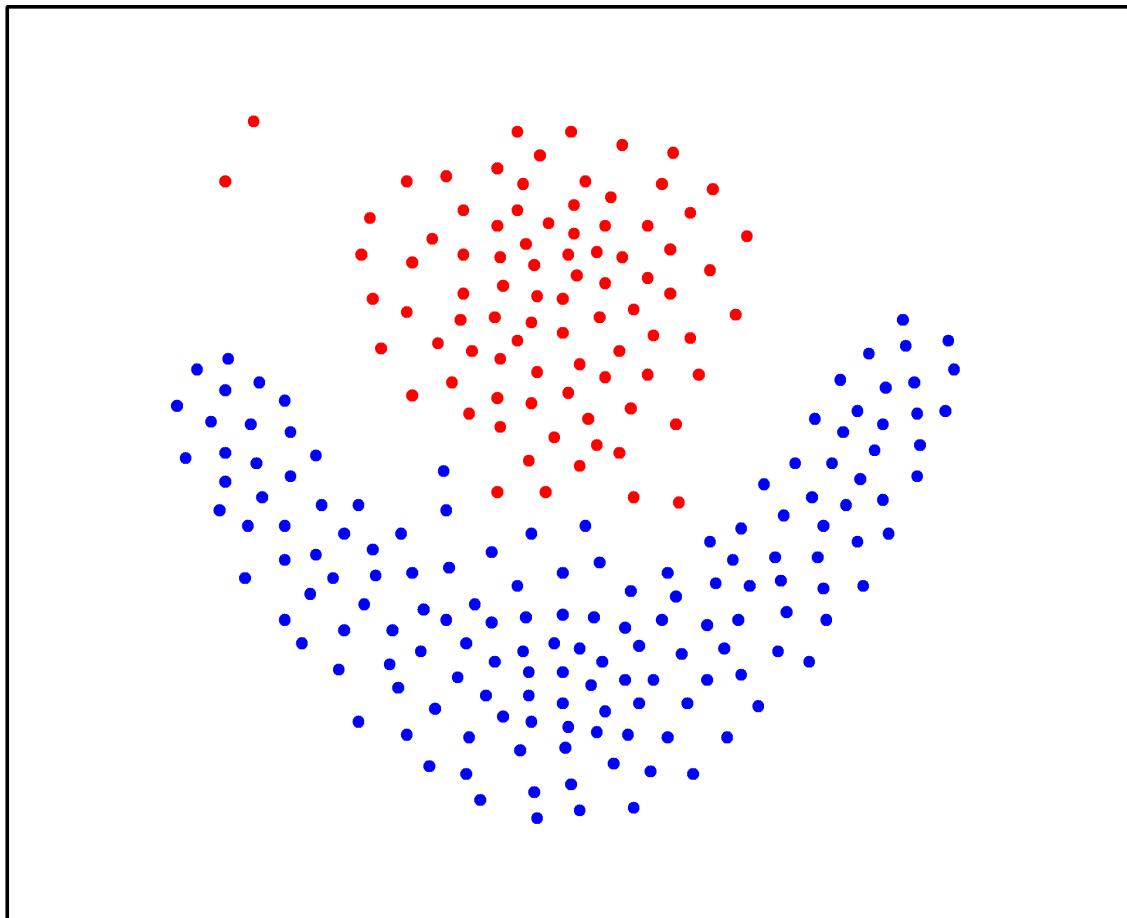
# What is a cluster???



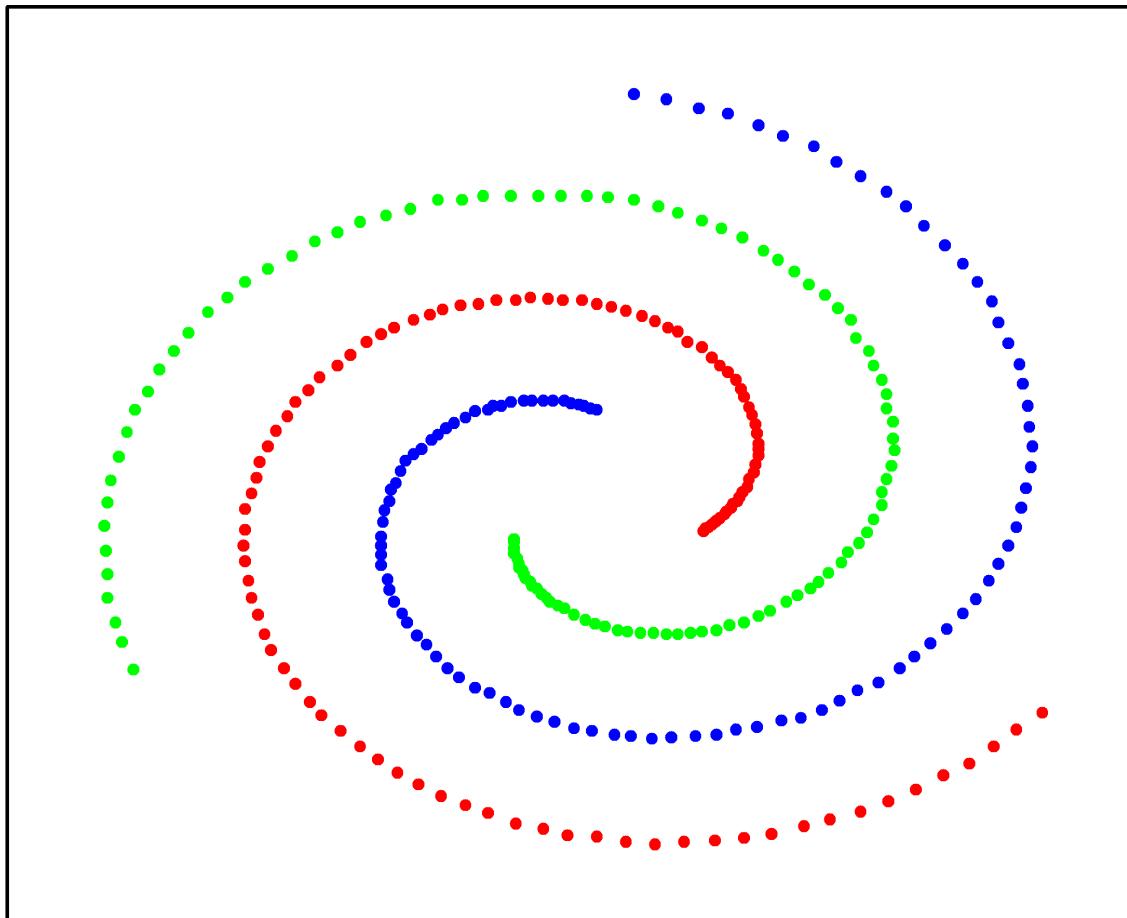
# Other cluster examples...



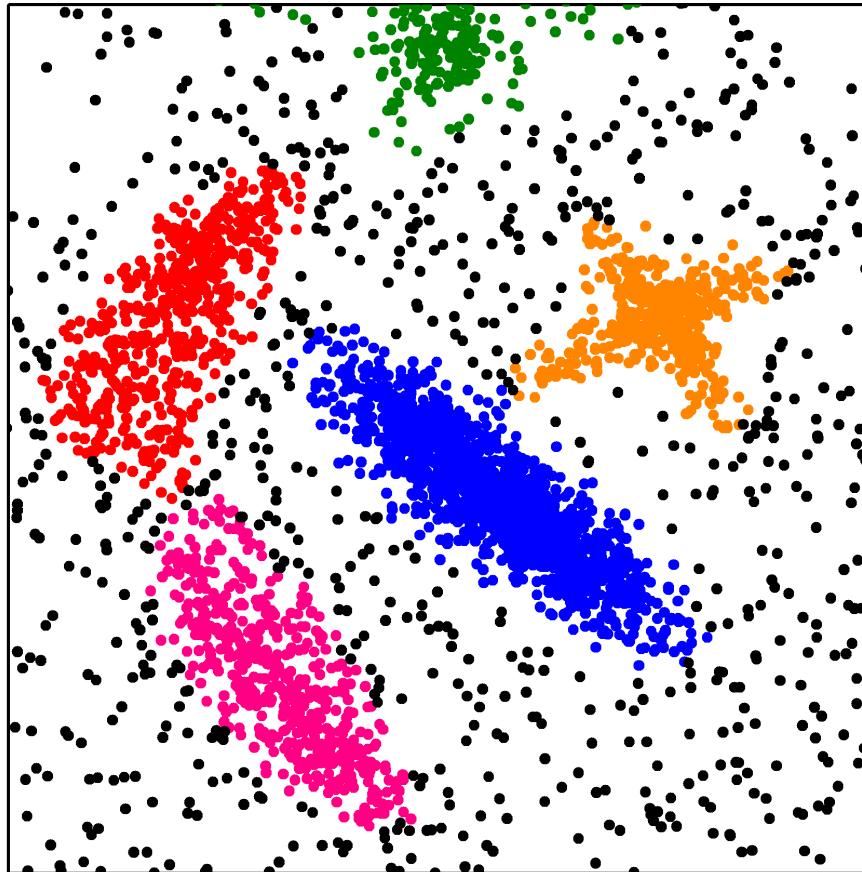
# Other cluster examples...



# Other cluster examples...

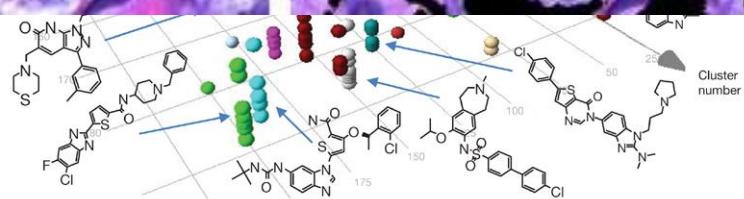
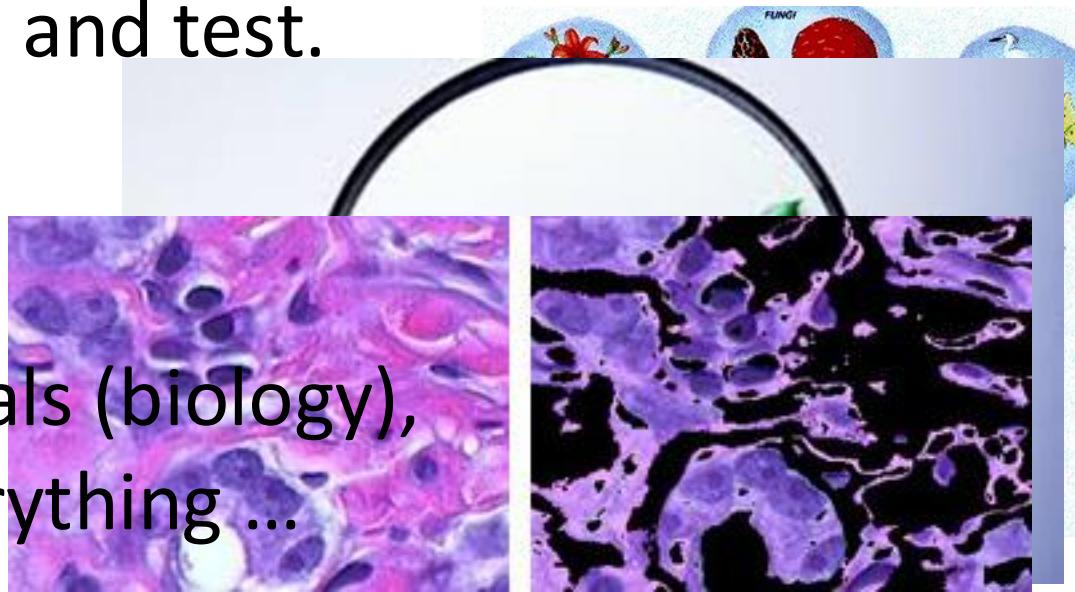


# Other cluster examples...



# Data mining technique that can be used for:

- Decide which set of drugs from a big library we should synthesize and test.
- WWW (googling...)
- Image recognition
- Classify plants, animals (biology), books(libraries), everything ...

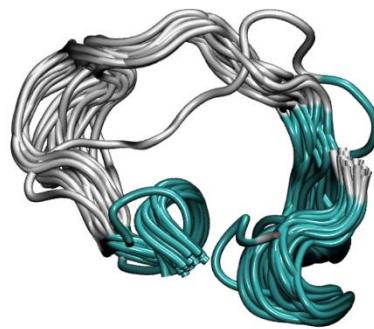
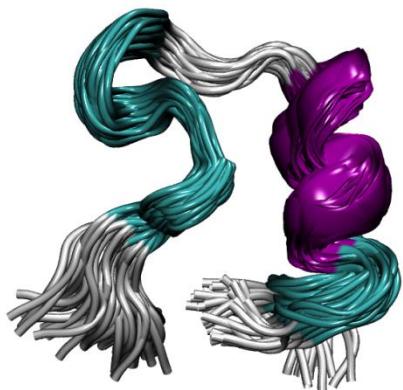
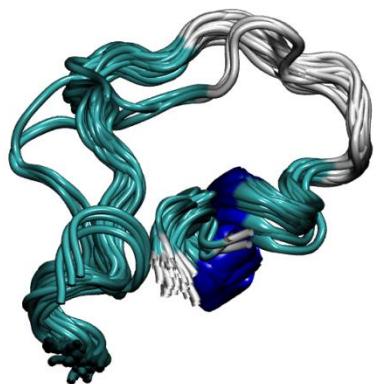
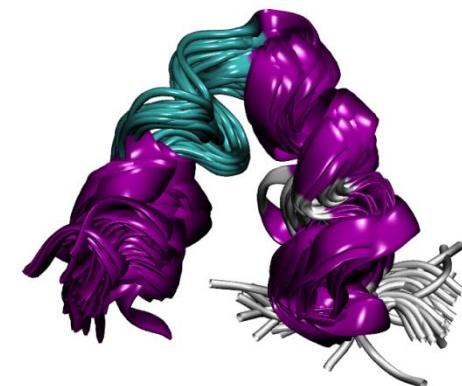
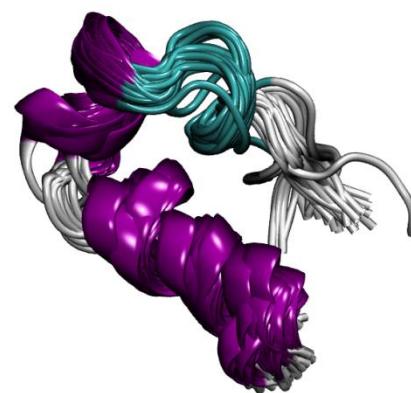
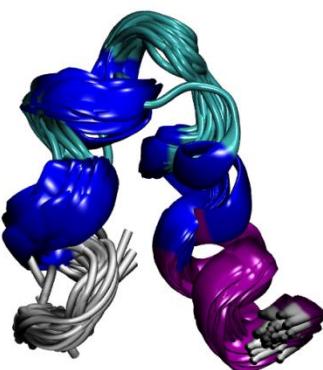
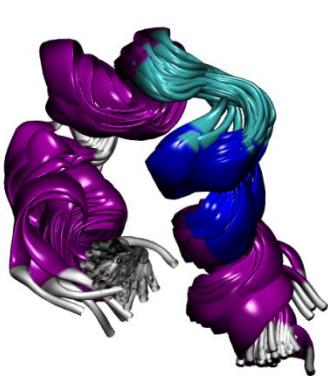


# Why Clustering?

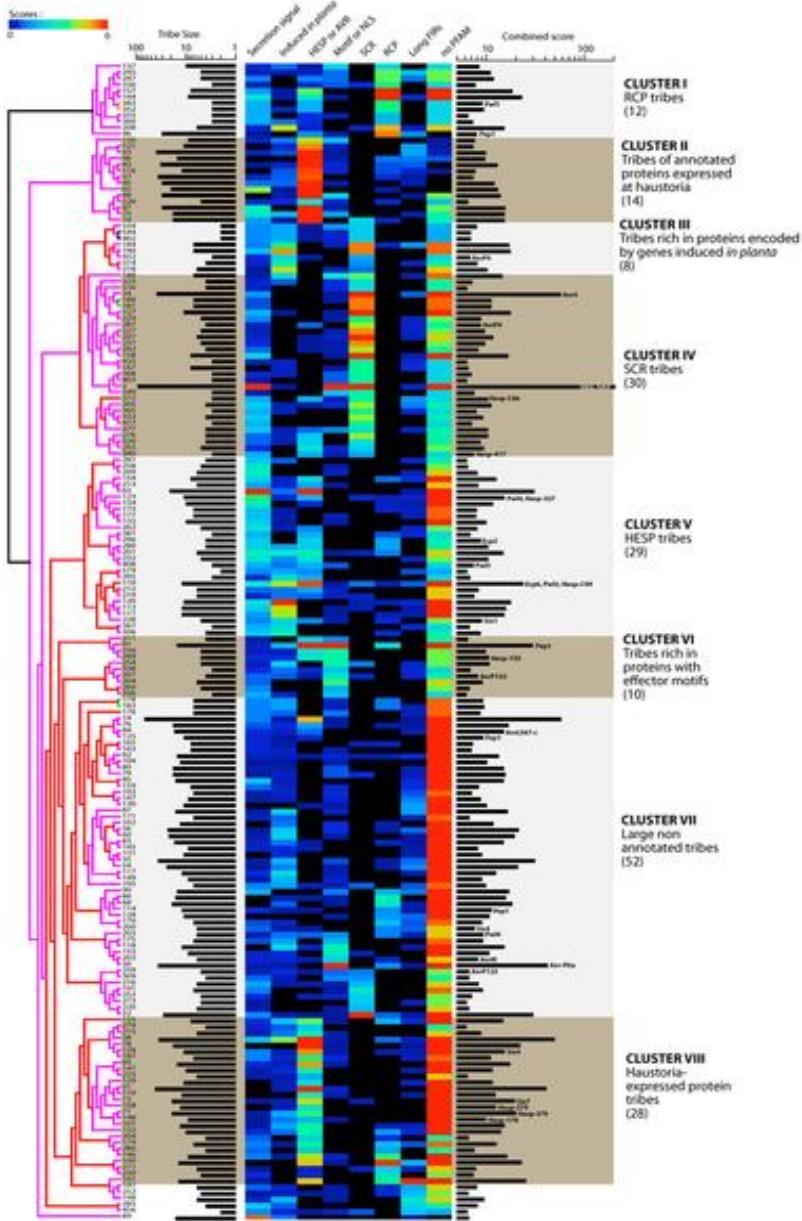
*“Understanding our world requires conceptualizing the similarities and differences between the entities that compose it” (Tyron and Bailey, 1970).*

- Clustering ≠ Classification
  - Clustering generate groups
  - Classification generate groups & labels

# Some clustering examples (1)



# Some clustering examples (2)



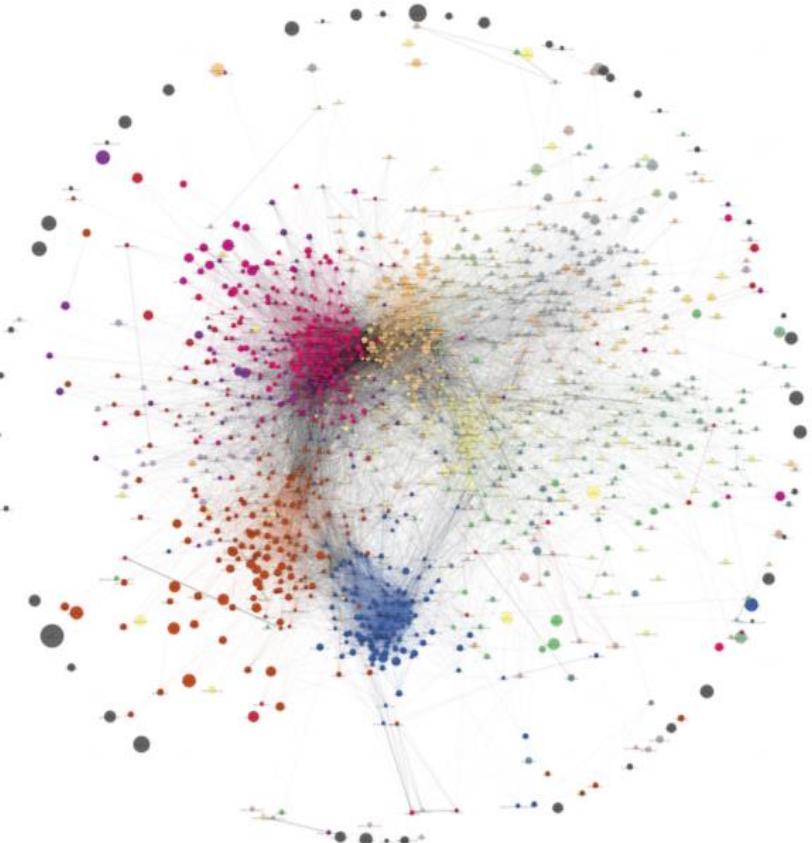
## Hierarchical clustering of the secretome reveals clusters of secreted protein families as high priority effector candidates.

Saunders DGO, Win J, Cano LM, Szabo LJ, Kamoun S, et al. (2012) Using Hierarchical Clustering of Secreted Protein Families to Classify and Rank Candidate Effectors of Rust Fungi. PLoS ONE 7(1): e29847. doi:10.1371/journal.pone.0029847

<http://journals.plos.org/plosone/article?id=info:doi/10.1371/journal.pone.0029847>

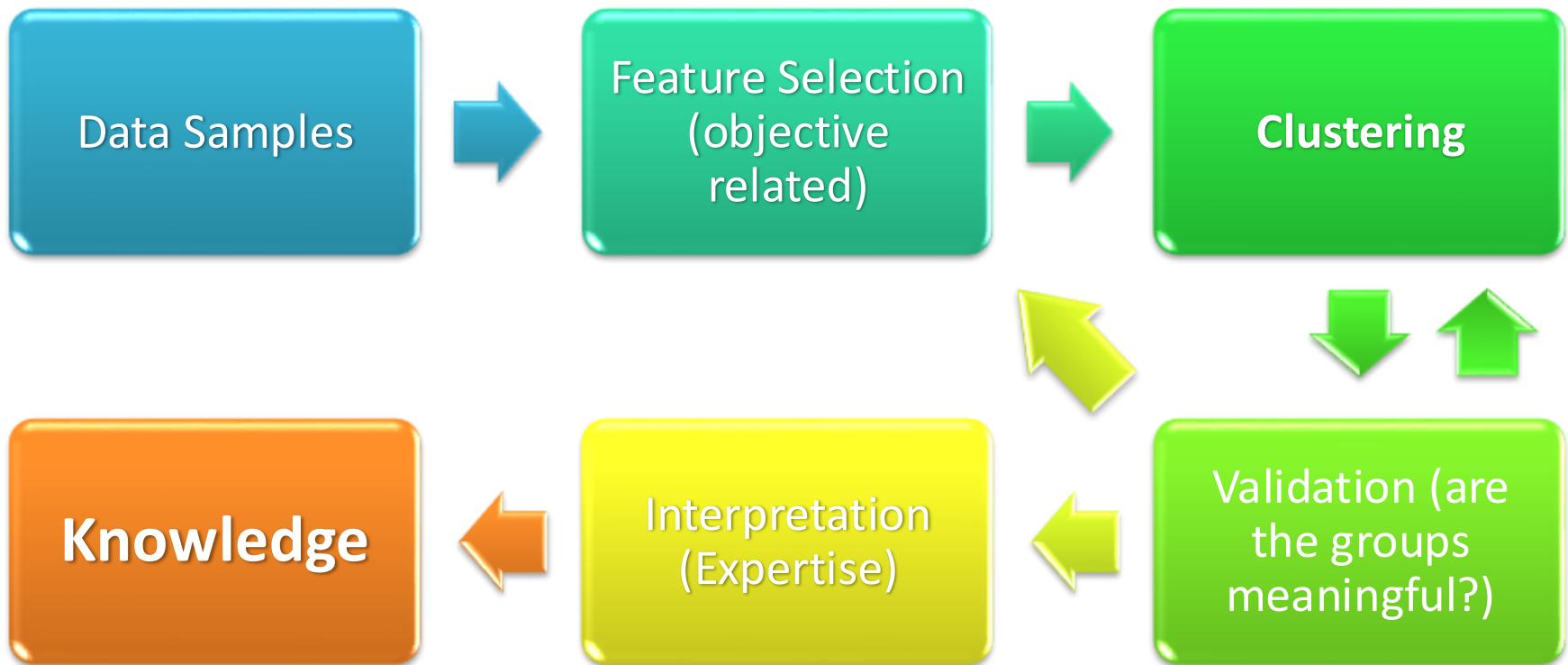
By clustering and classifying plant proteins detect protein groups (families) that will probably act against mold

# Some clustering examples (3)

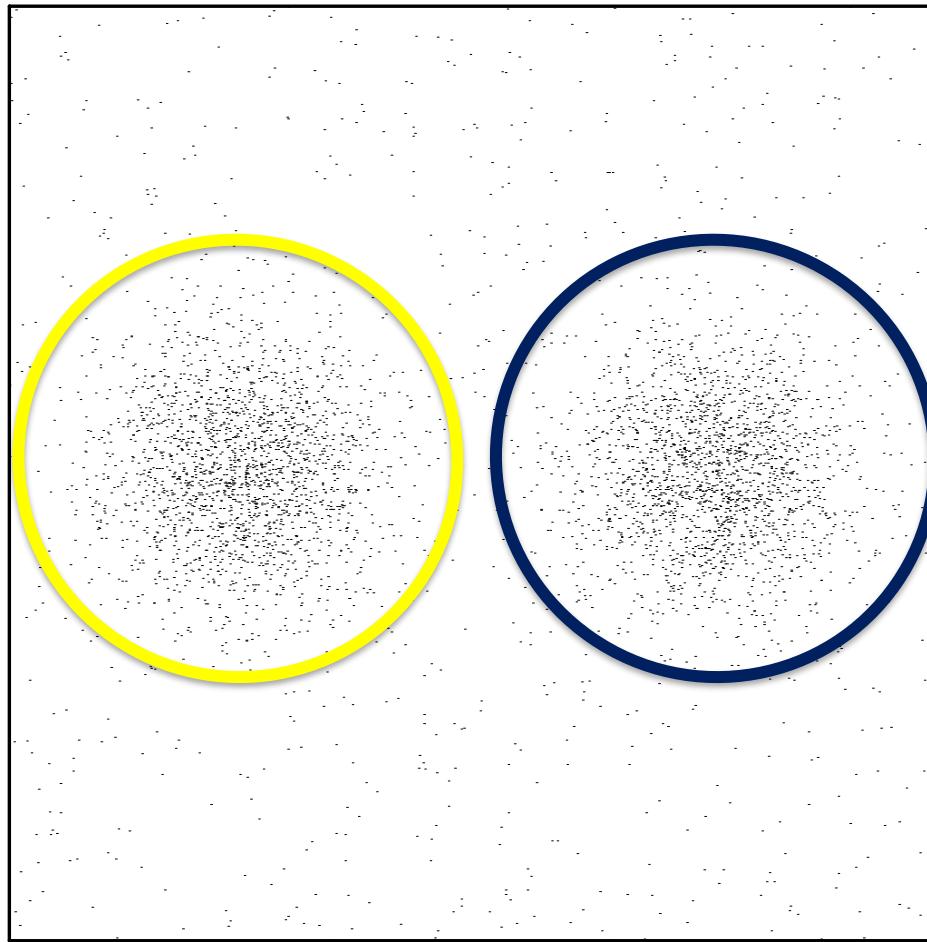


- Web search algorithms need to cluster.
- Not always known
- Cluster can be done in many ways (cluster the net by connections, cluster the results by text mining...)

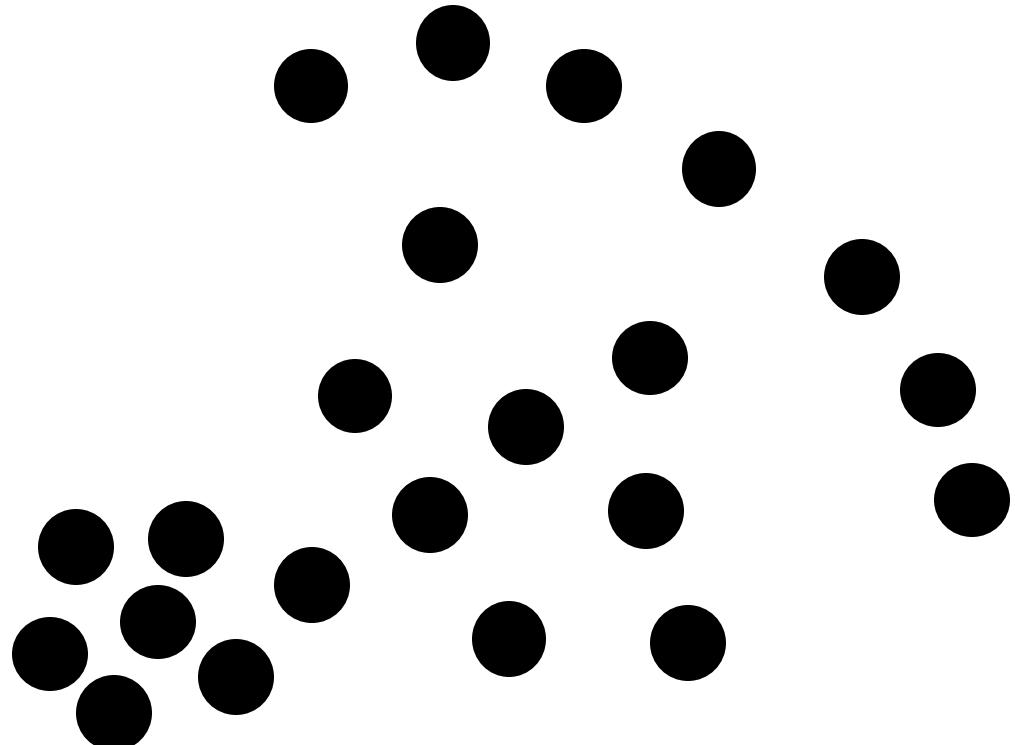
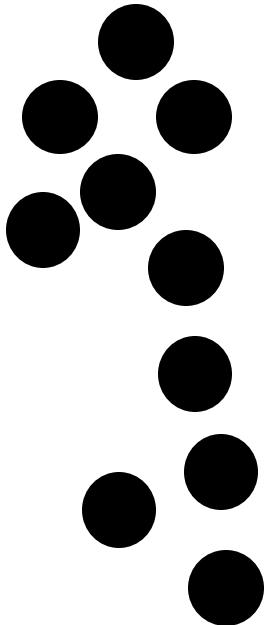
# Clustering procedure



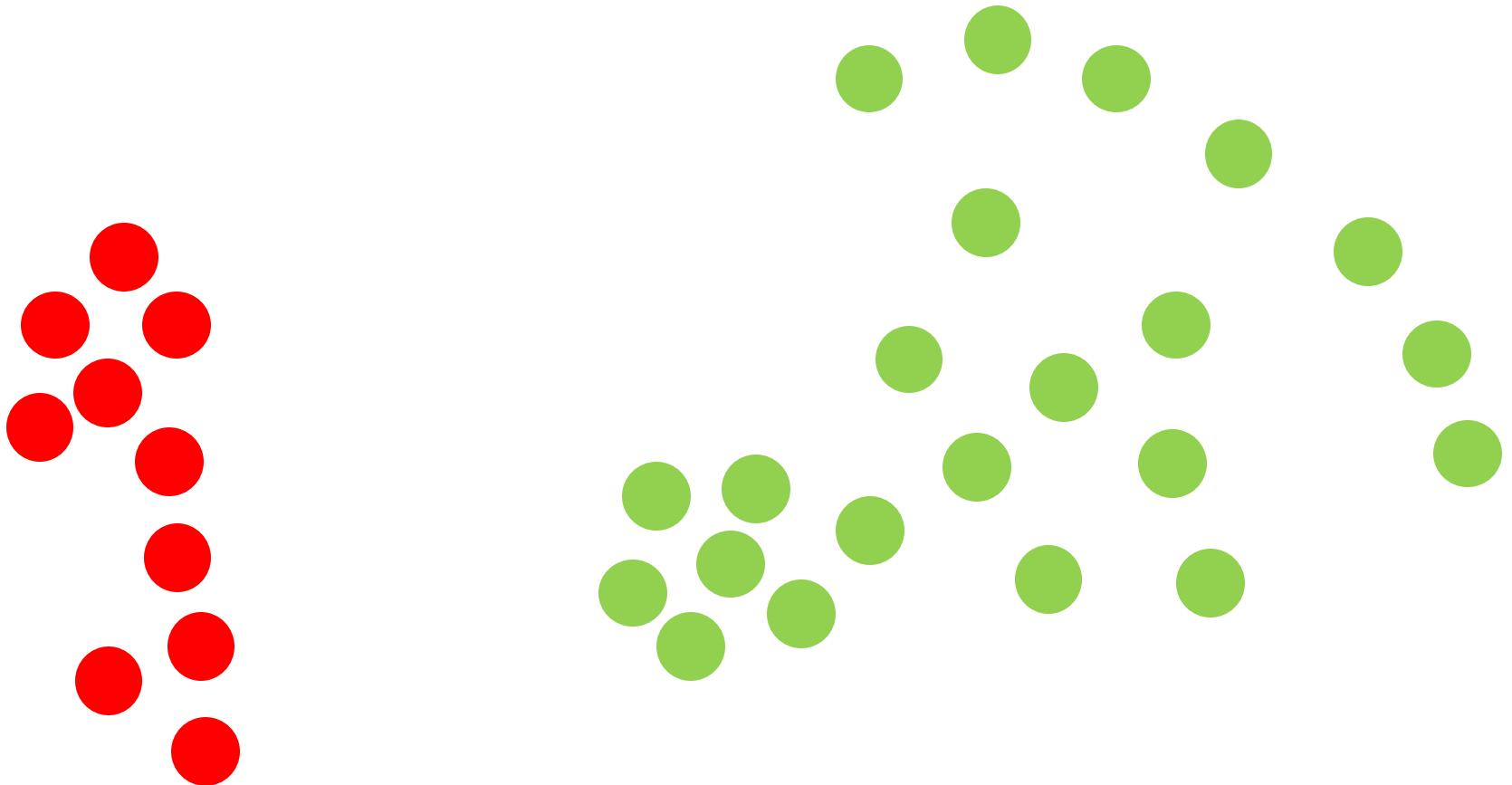
# What is a cluster (revisited)



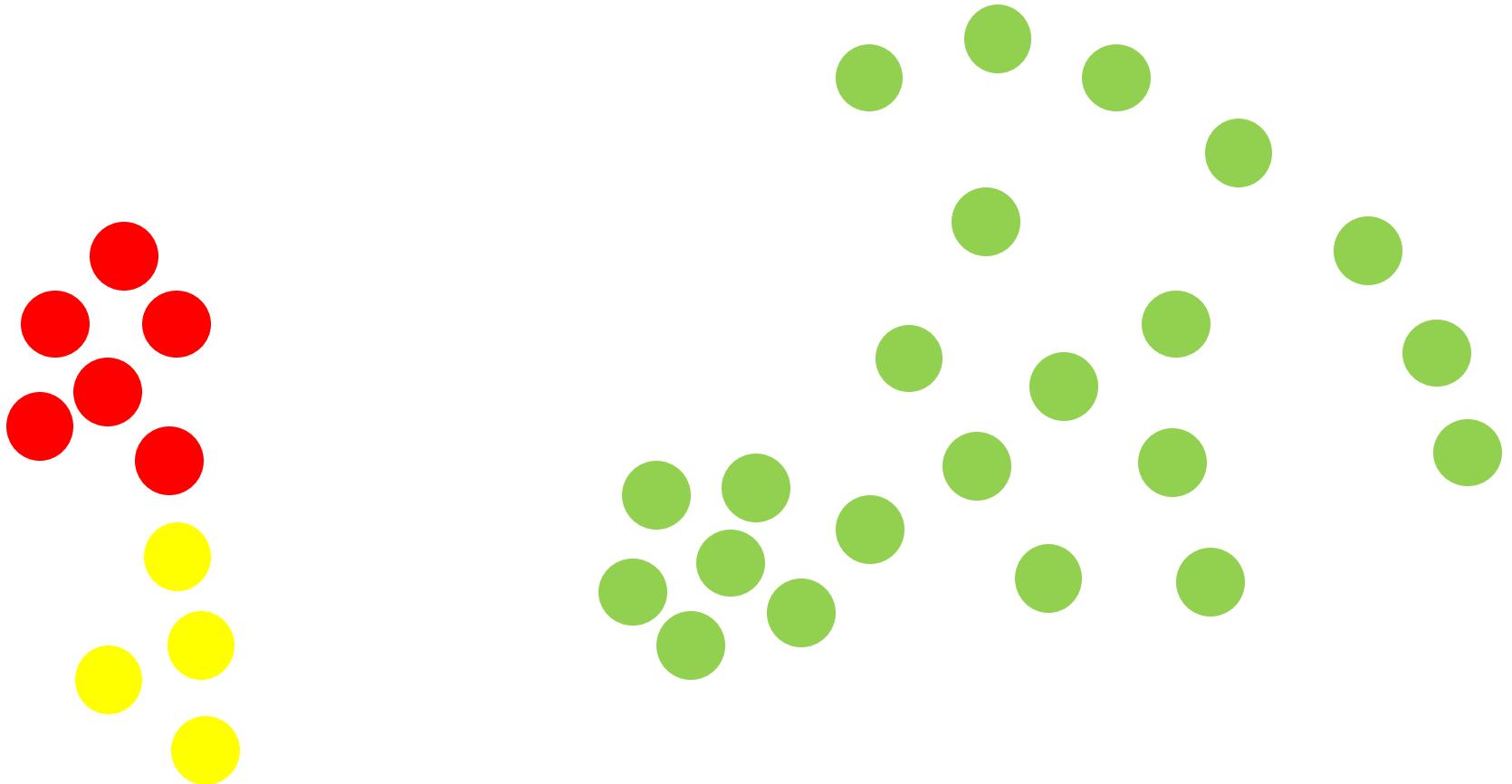
# What is a cluster (revisited)



# What is a cluster (revisited)



# What is a cluster (revisited)



# What is a cluster (revisited)



# What is a cluster (revisited)



# What is a cluster (revisited)



# Some consideration about Clustering

- Tautology: the result of the clustering process depends on your cluster definition
- And it depends on the metric
- And it also depends on the features that you have chosen



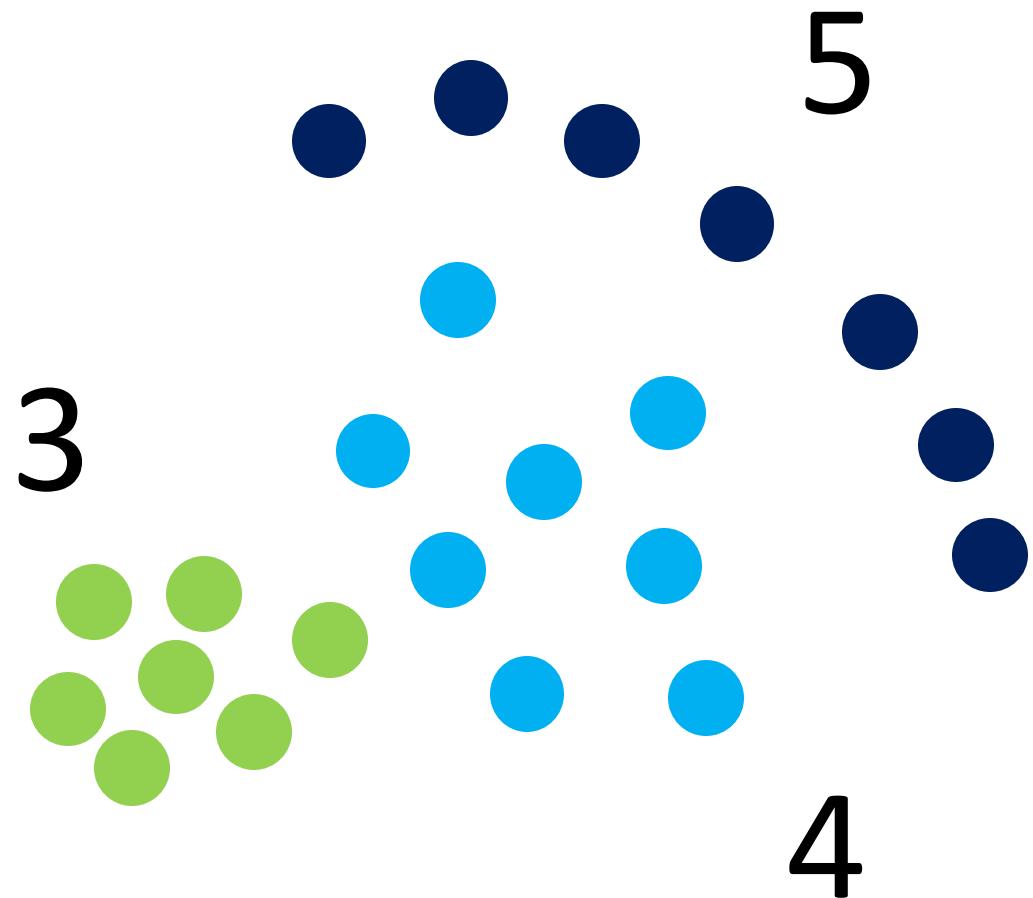
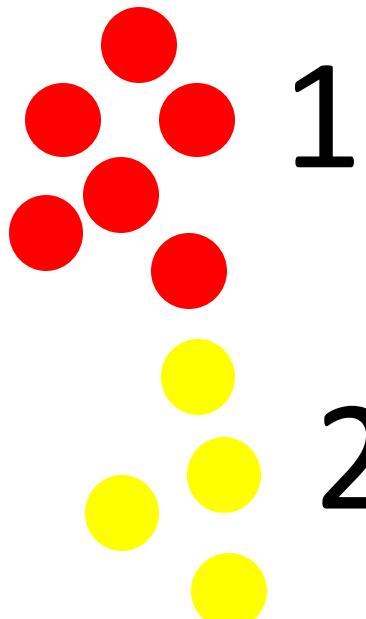
# Flat, fuzzy and hierarchical clustering

- Flat clustering performs a hard partition of the data
- Fuzzy clustering is a flat clustering algorithm with soft element assignation
- Hierarchical clustering generates a tree instead of a single partition. Can be agglomerative (joining elements) or divisive (dividing the dataset)

# Flat clustering

- Each element is assigned to a single cluster.  
$$Cl(i) = l$$
- Traditionally, the number of clusters ( $k$ ) should be given to the algorithm as an external parameter. Nowadays, many clustering algorithms estimate internally this parameter.
- usually when one performs clustering one looks for a hard partition.
- Can not deal well with multilevel structures

# Flat clustering



# Fuzzy clustering

- Each element is assigned to *all* the clusters in the dataset with a given degree of membership.

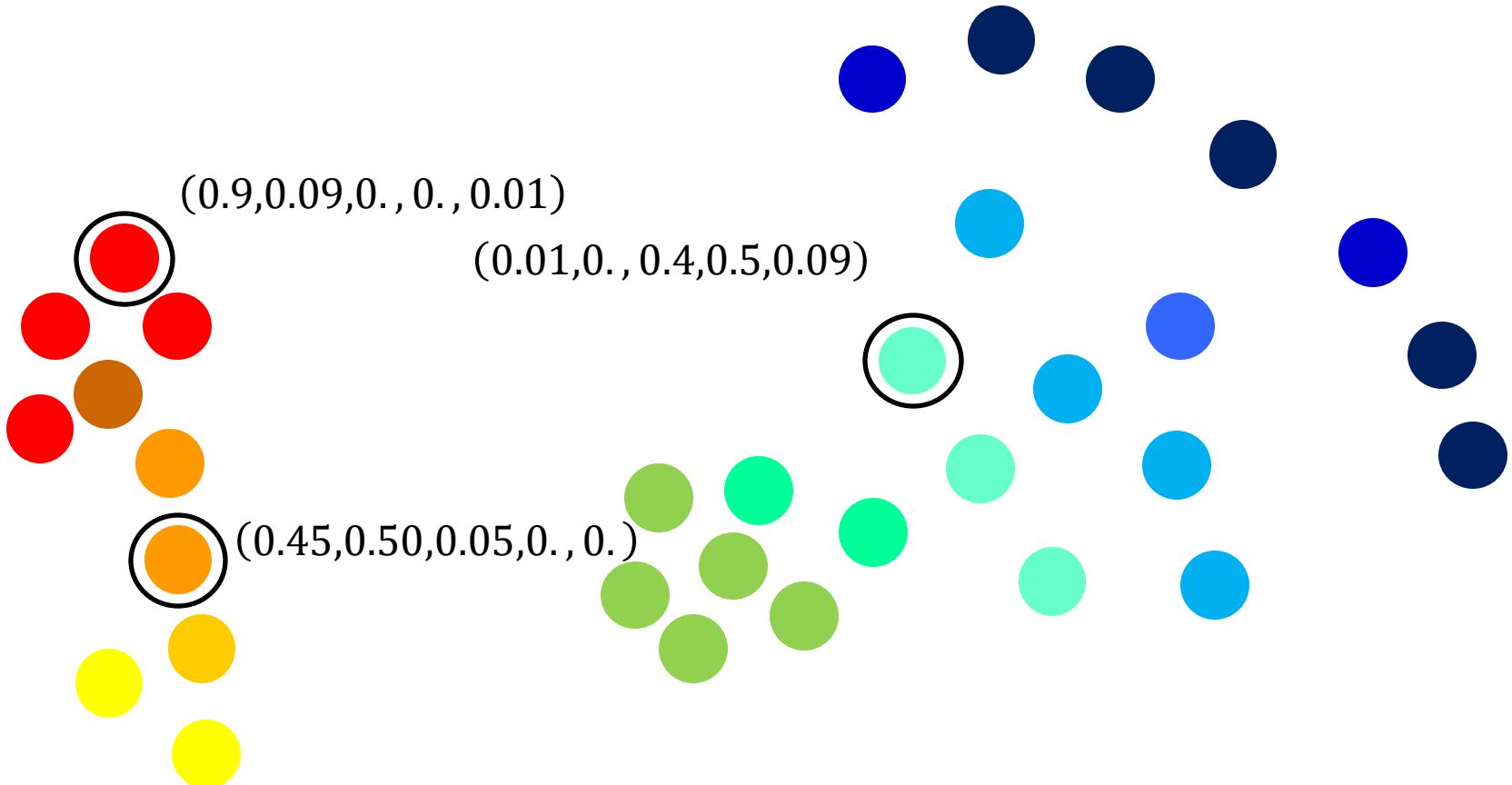
$$\vec{Cl}(i) = (u_1, u_2, \dots, u_l \dots, u_k)$$

- The membership vector is normalized

$$\sum_{l=1}^k u_l = 1$$

- Again, the number of clusters should be provided as an external parameter.
- It may be difficult to transform in a hard partition

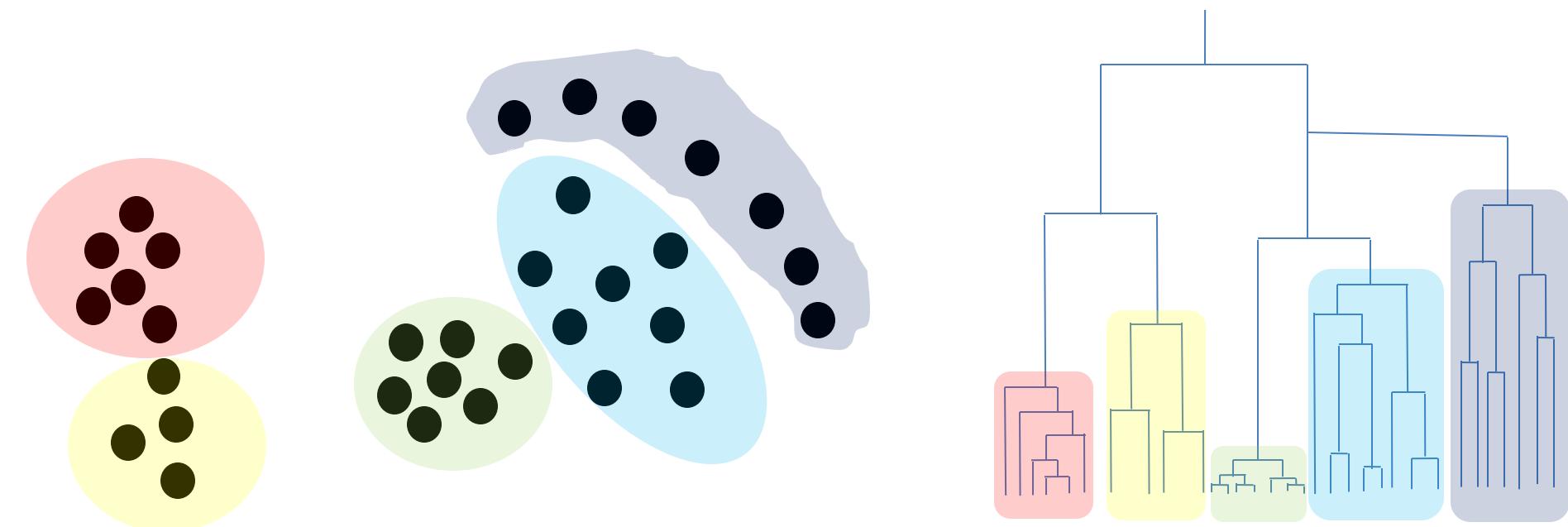
# Fuzzy clustering



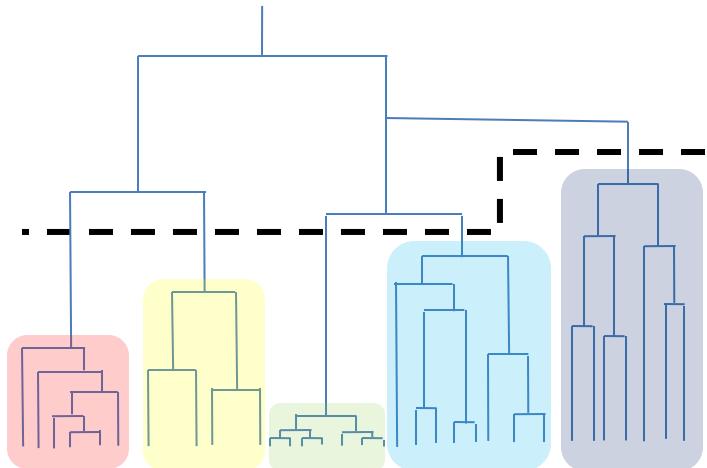
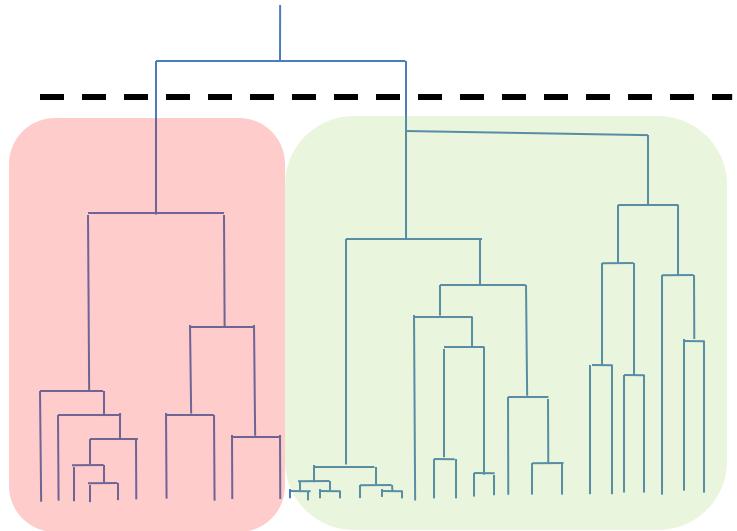
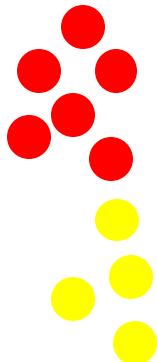
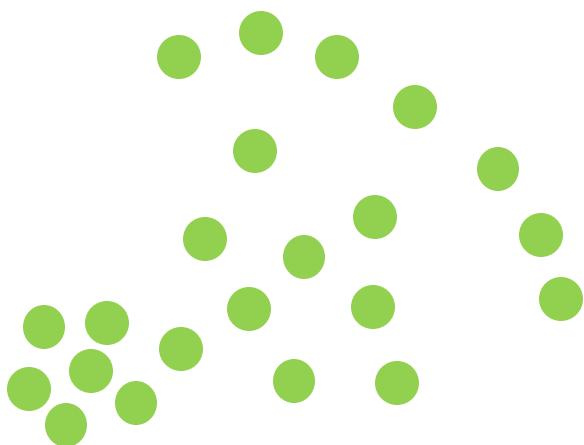
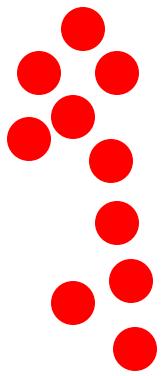
# Hierarchical clustering

- Produces a set of nested clusters organized as a hierarchical tree
- Can be visualized as a dendrogram (A tree-like diagram that records the sequences of merges or splits)
- No assumptions on the number of clusters
- Hierarchical clusterings may correspond to meaningful taxonomies
- It can be transformed in many hard partitions

# Hierarchical clustering



# Flattening the hierarchical clustering



# **CLASSICAL CLUSTERING ALGORITHMS**

# K-means: A flat clustering algorithm

MacQueen, J. (1967, June). Some methods for classification and analysis of multivariate observations. In Proceedings of the fifth Berkeley symposium on mathematical statistics and probability (Vol. 1, No. 14, pp. 281-297).

# K-means clustering

- Attempts to minimize the *intracluster* distance while maximizing the *intercluster* distance.
- It is based on the concept of cluster centroid, i.e. the average position of the cluster elements.
- Still widely used.
- It can be easily parallelized and linearized.
- User must provide  $k$

# K-means objective function

- Loss function:

$$O(z) = \sum_{l=1}^k \sum_{i=1}^n \delta(z_i, l) \|\vec{x}_i - \vec{c}_l\|^2$$

- $z$  is an array with  $n$  components that reflects the assignation in clusters.
- $\vec{c}_l$  is the vector with the coordinates of the  $l$ -th cluster centroid.  $\vec{c}_l = \frac{\sum_{i=1}^n \delta(z_i, l) \vec{x}_i}{\sum_{i=1}^n \delta(z_i, l)}$

# $k$ -means algorithm

**Input:** cluster size  $k$ , instances  $\{\vec{x}_i\}_{i=1}^n$

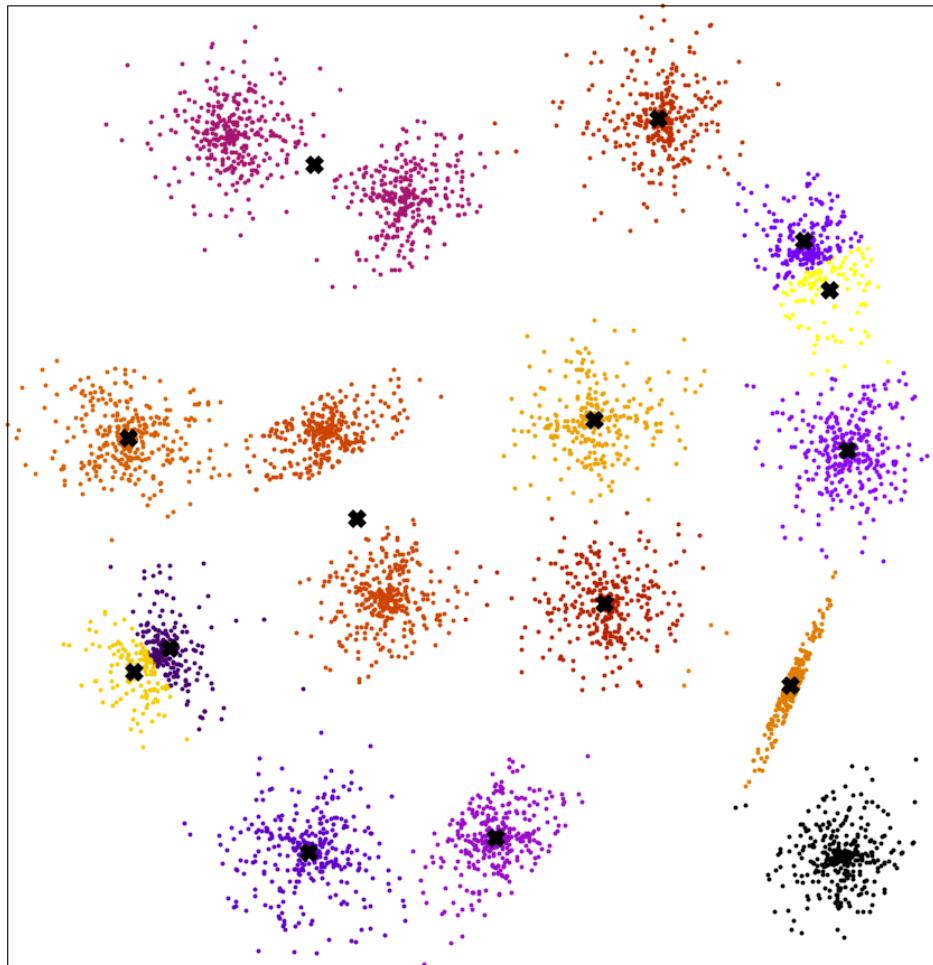
**Output:** cluster membership assignments  $\{z_i\}_{i=1}^n$

1. Initialize  $k$  cluster centroids  $\{\vec{c}_l\}_{l=1}^k$  (randomly from the data set).
2. Repeat until no instance changes its cluster membership:
  - Decide the cluster membership of instances by assigning them to the nearest cluster centroid
$$z_i = \operatorname{argmin}_l(d(\vec{c}_l, \vec{x}_i)) \quad \text{Minimize intra distance}$$
  - Update the  $k$  cluster centroids based on the assigned cluster membership

$$\vec{c}_l = \frac{\sum_{i=1}^n \delta_{z_{il}} \vec{x}_i}{\sum_{i=1}^n \delta_{z_{il}}} \quad \text{Maximize inter distance}$$

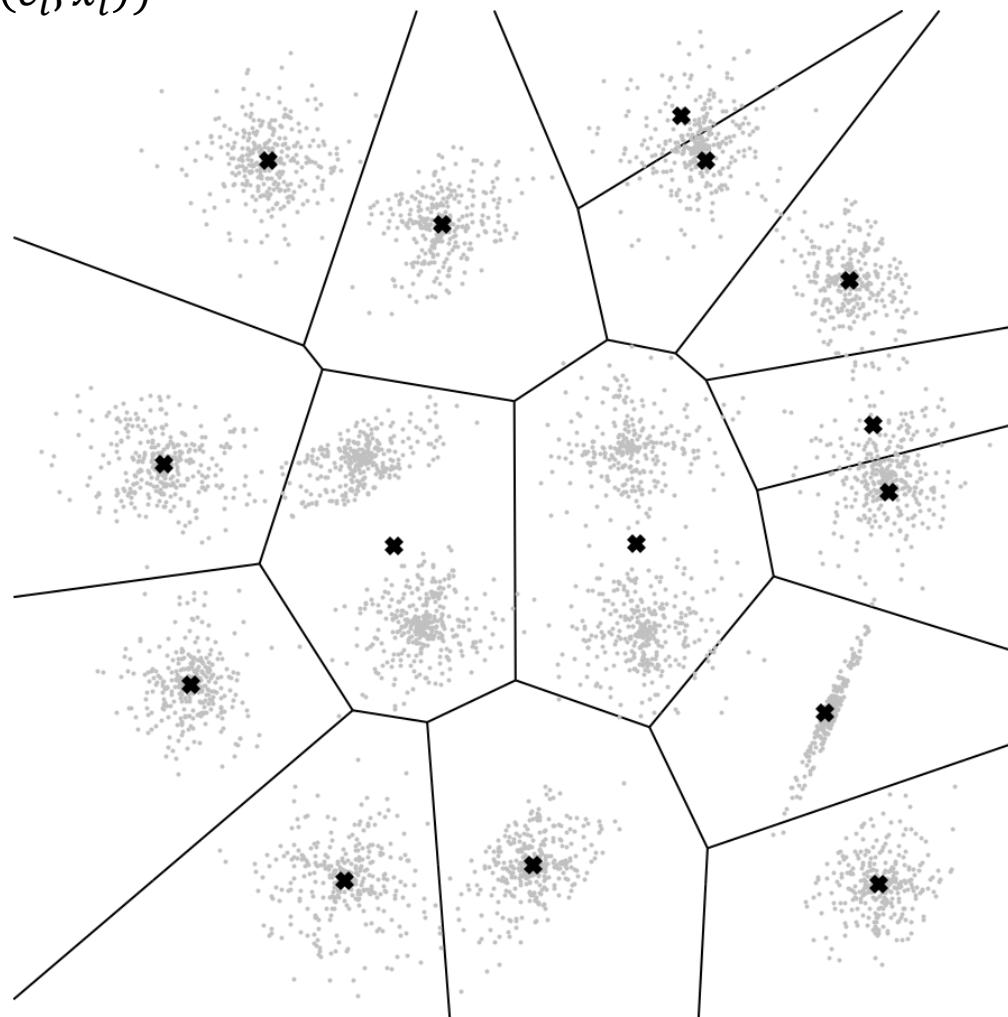
# *k*-means illustration

- Randomly pick  $k$  centers.
- Assign each point to its nearest center.
- Recompute centers.
- Iterate...



# Partitioning the space

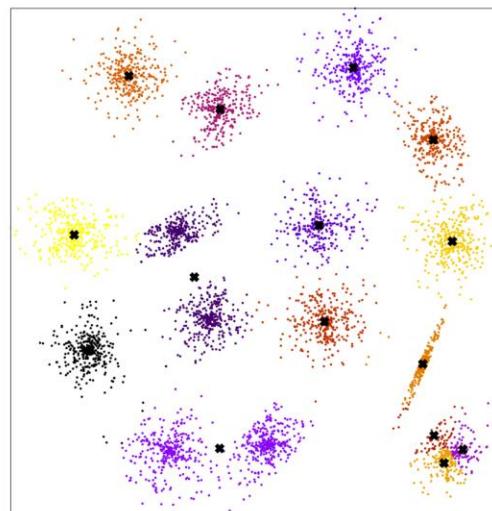
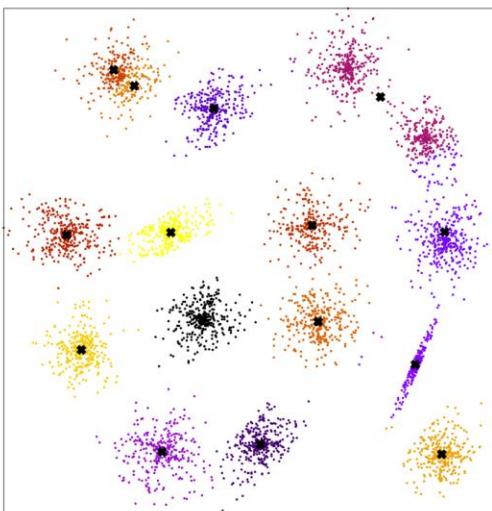
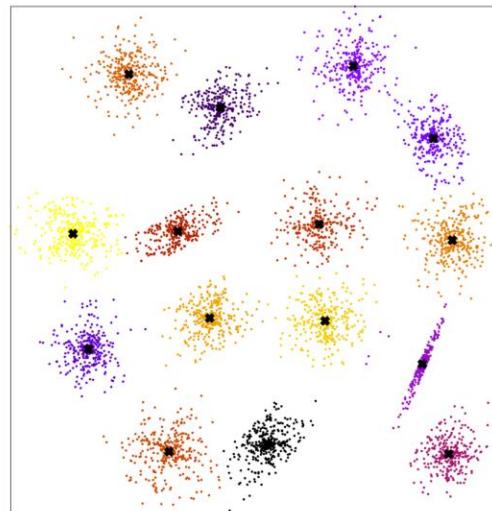
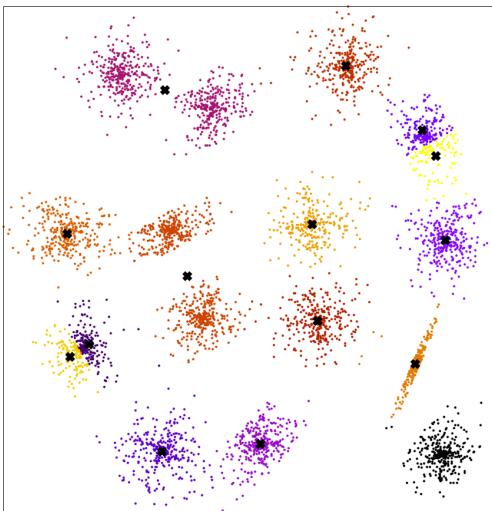
$$z_i = \operatorname{argmin}_l(d(\vec{c}_l, \vec{x}_i))$$



# K-means weakness

- Initialization sensitive (local optimization) → k-means++
- Which k-employed → Scree test
- Sensitive to outliers → k-medoids
- employs Euclidean distance → k-medoids
- Only for spherical clusters → kernel k-means (advanced)

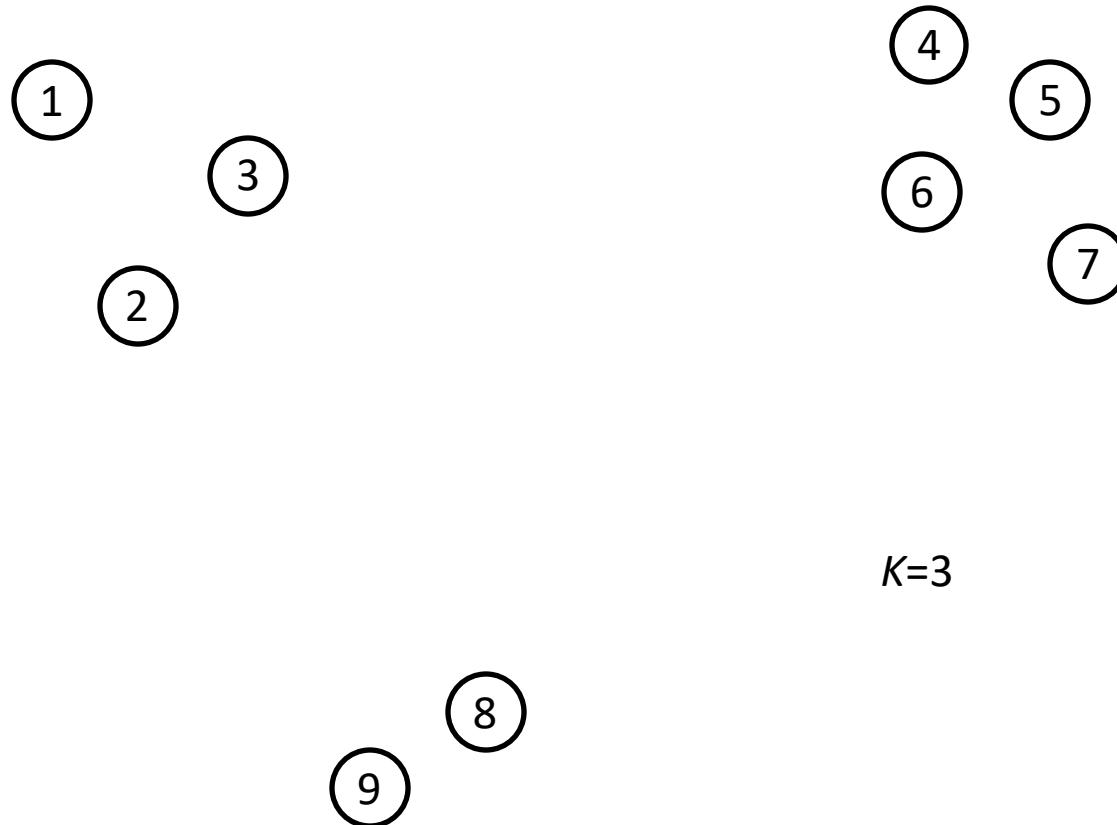
# Different initializations



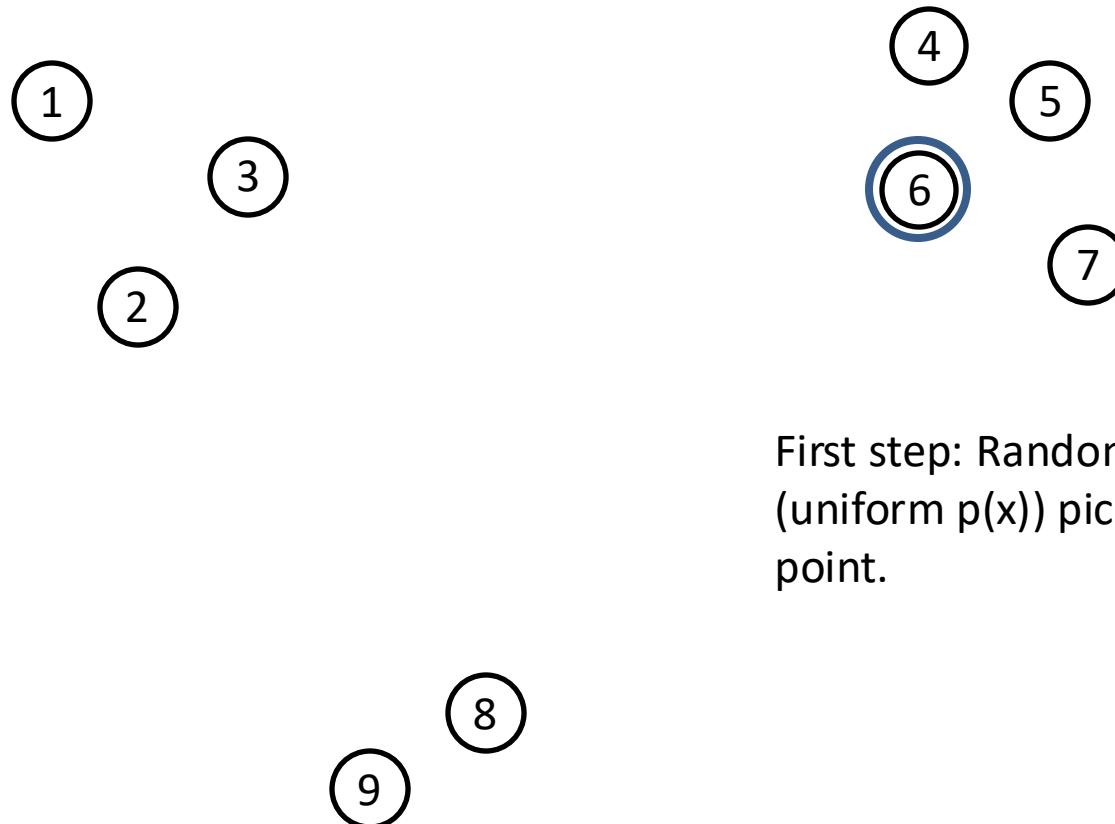
# Better initialization: $k$ -means++

1. Choose the first cluster center at random
2. Repeat until all  $k$  centers have been found
  - For each instance compute  $D_x = \min_k [d(x, c_k)]$
  - Choose a new cluster center with probability
$$p(x) \propto D_x^2 \quad \text{← } \begin{matrix} \text{new center should be far} \\ \text{away from existing centers} \end{matrix}$$
3. Run  $k$ -means with selected centers as initialization

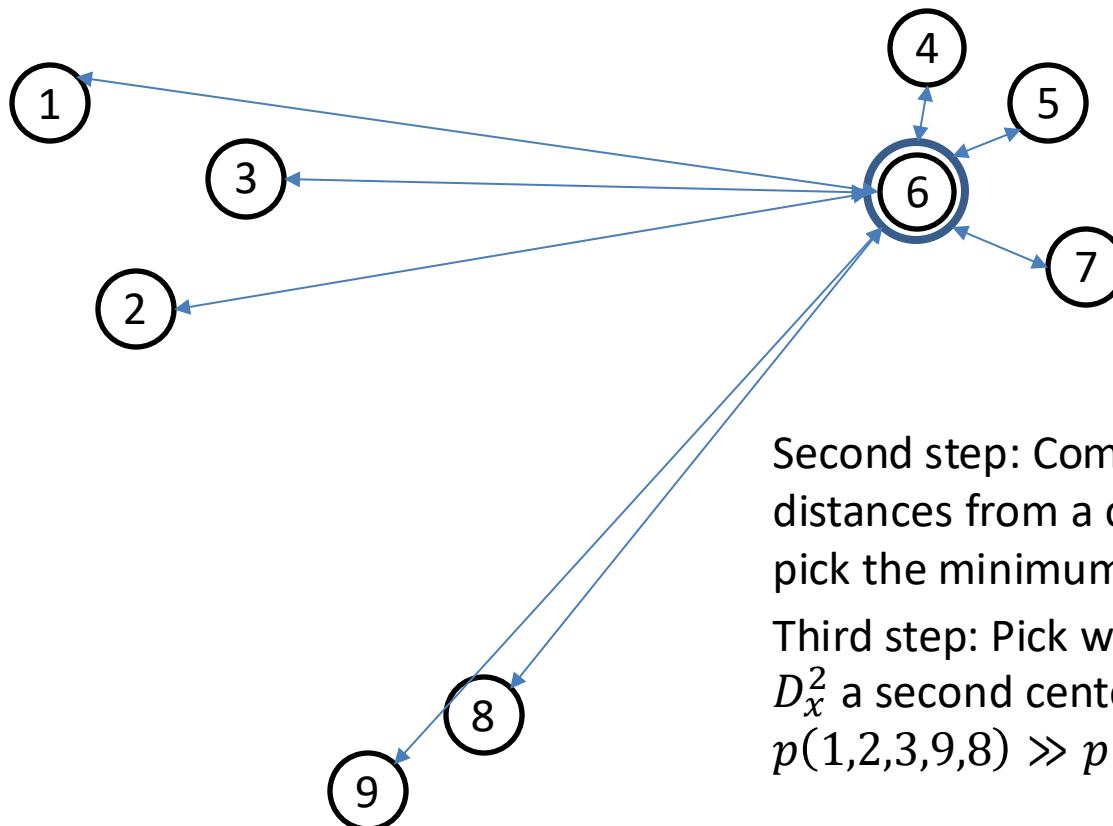
# K-means ++



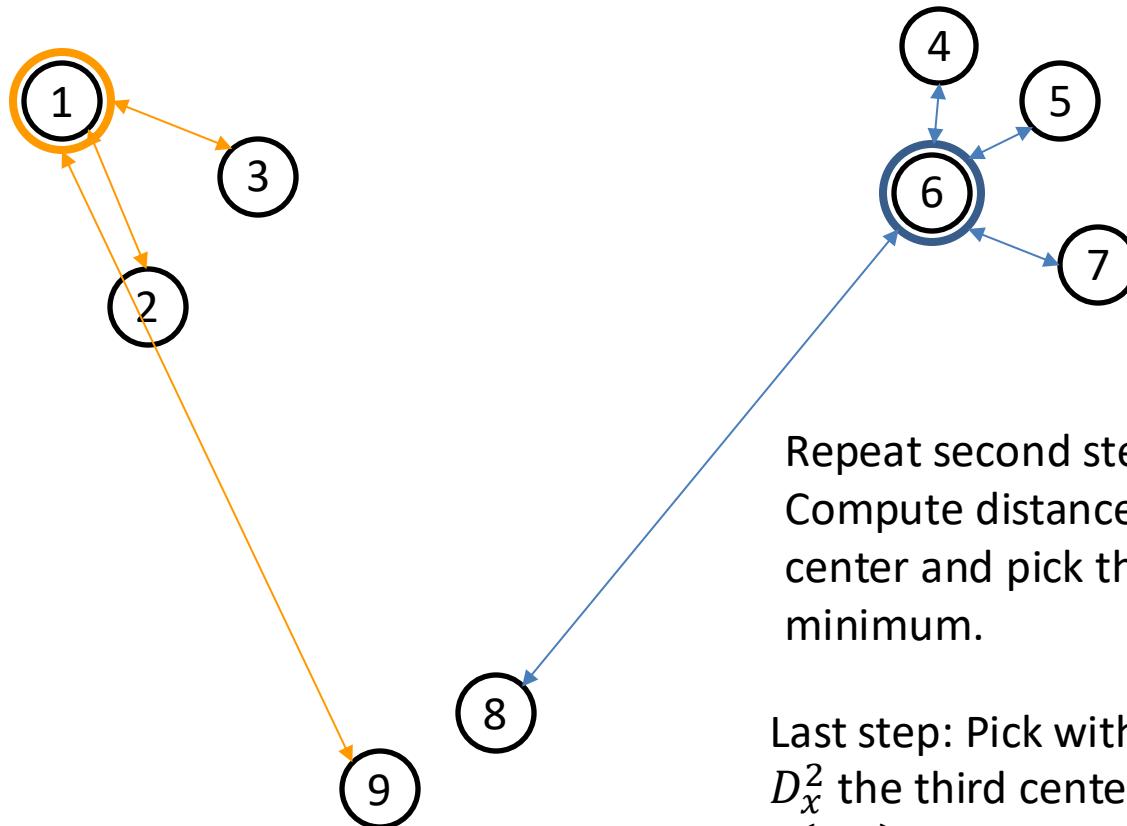
# K-means ++



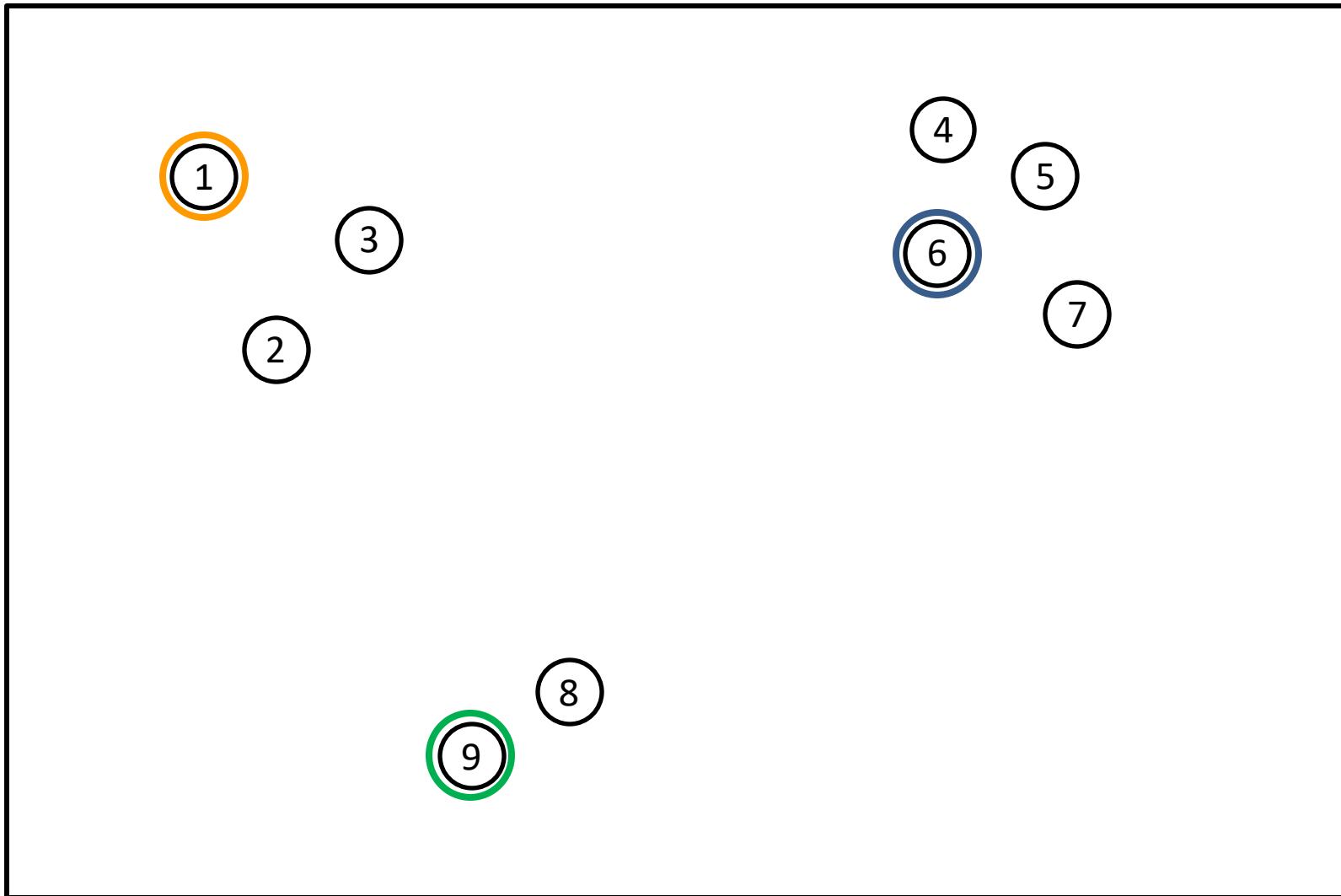
# K-means ++



# K-means ++



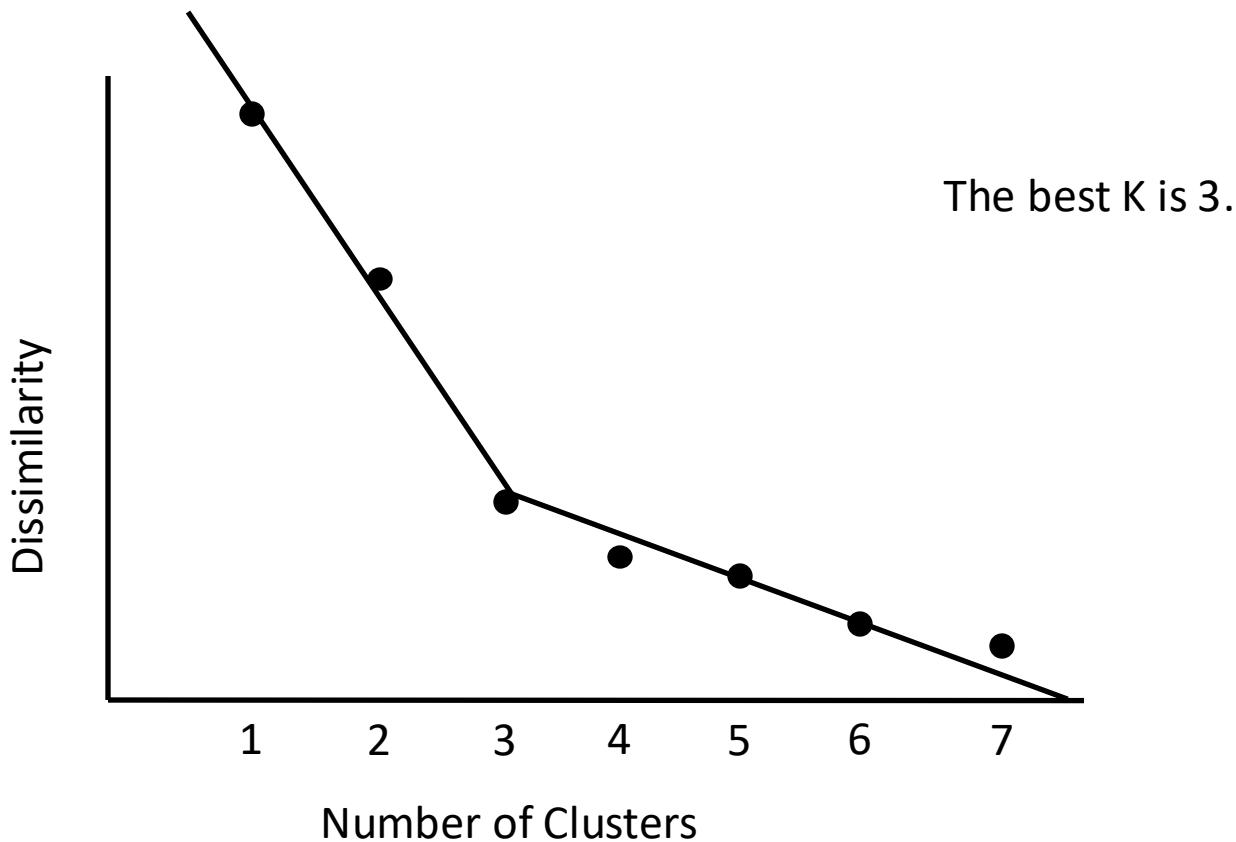
# K-means ++



# K-means weakness

- Initialization sensitive (local optimization) → k-means++
- Which k-employed → Scree test
- Sensitive to outliers → k-medoids
- employs Euclidean distance → k-medoids
- Only for spherical clusters → kernel k-means (advanced)

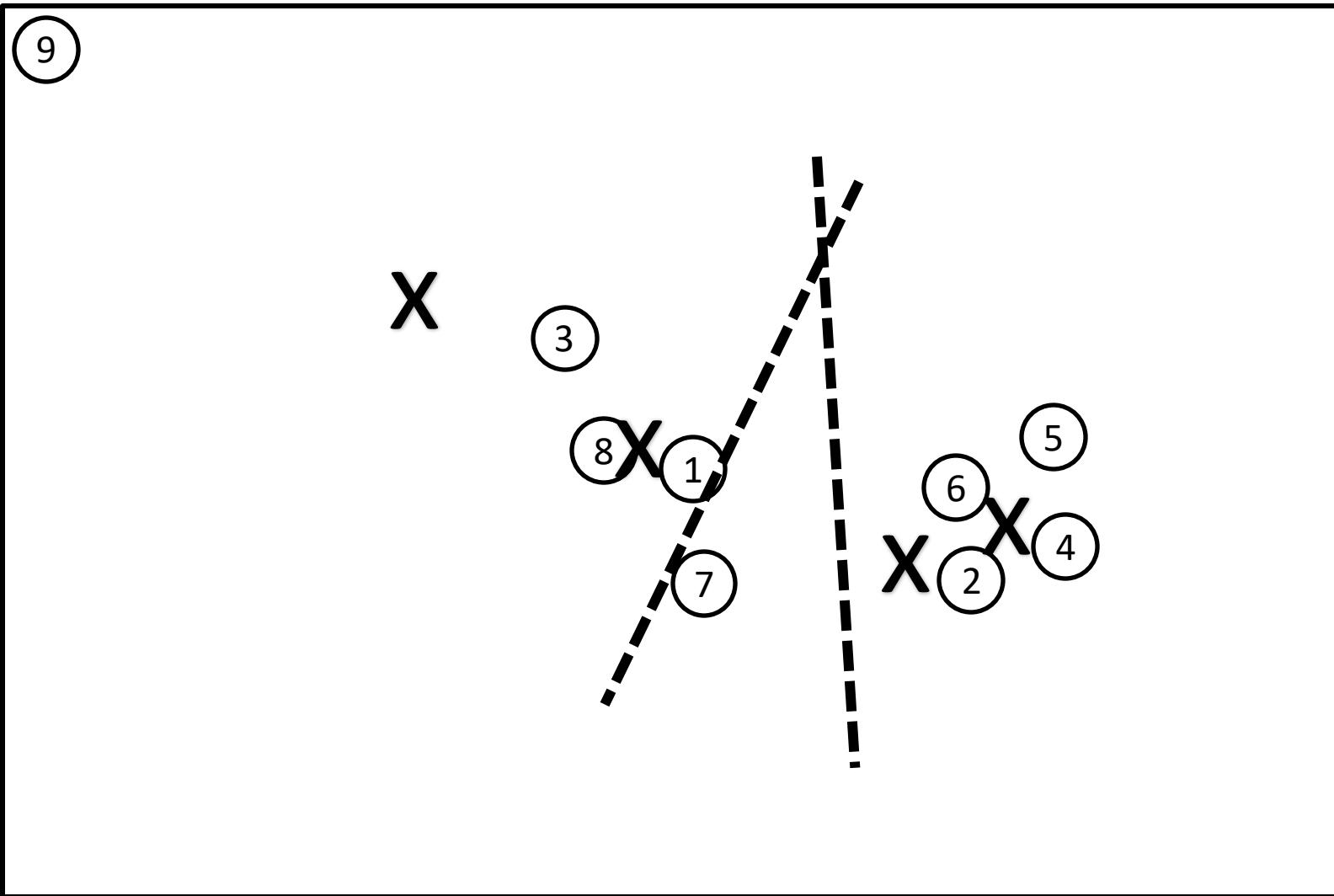
# Scree test for k-means



# K-means weakness

- Initialization sensitive (local optimization) → k-means++
- Which k-employed → Scree test
- Sensitive to outliers → k-medoids
- employs Euclidean distance → k-medoids
- Only for spherical clusters → kernel k-means (advanced)

# Sensitivity to outliers

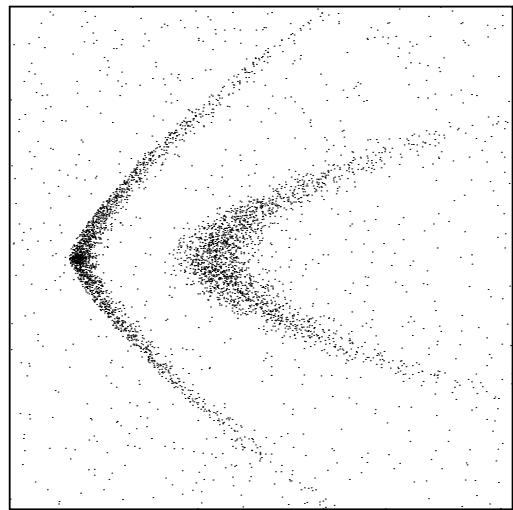


# K-medoids

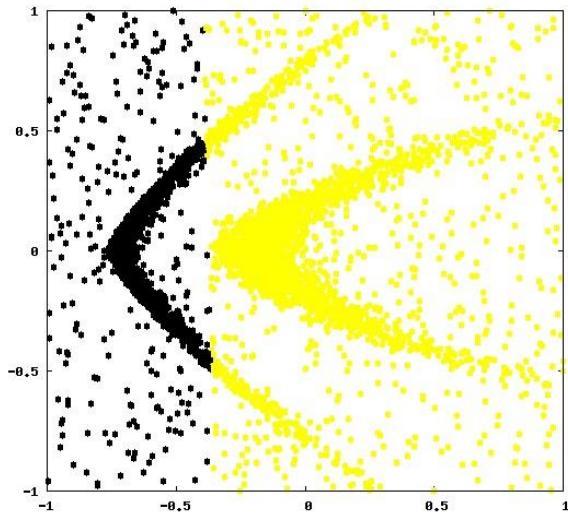
- It can be used with all kind of distances.
- Reduces the impact of outliers.
- Instead of working with centroids, it works with medoids, i.e. the most central element of the cluster.
- The cluster medoid is the element with minimum sum of distances to the rest of the elements of the cluster

# K-means weakness

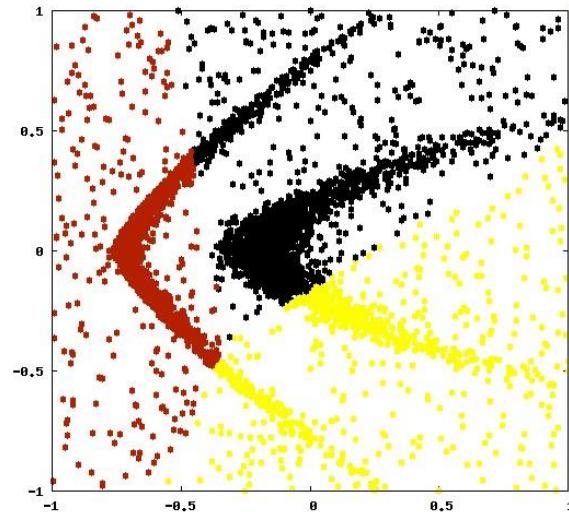
- Initialization sensitive (local optimization) → k-means++
- Which k-employed → Scree test
- Sensitive to outliers → k-medoids
- employs Euclidean distance → k-medoids
- Only for spherical clusters → kernel k-means (advanced)



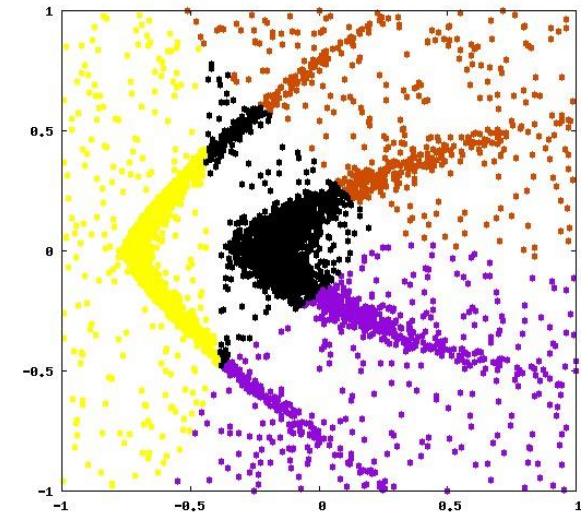
K=2



K=3



K=4





# Yesterday

- Considerations about clustering: Flat, Fuzzy and Hierarchical clustering.
- Clustering methods:
  - k-means (Flat clustering)

# Kernel k-means

**Observation:** The k-means algorithm only needs to compute the distances from the centers of the clusters to the data points.

*Let's use the kernel trick!*

- Project the data set into a feature space by means of a nonlinear mapping  $\vec{x}_i \rightarrow \vec{\phi}_i$ .
- Distances in the feature space can be easily computed with the kernel trick:

$$\|\vec{\phi}_i - \vec{\phi}_j\|^2 = K^{ii} + K^{jj} - 2K^{ij}$$

- The loss function and the centroids are analogous to the ones in standard k-means, but in the feature space.

$$O(z) = \sum_{l=1}^k \sum_{i=1}^n \delta(z_i, l) \|\vec{\phi}_i - \vec{v}_l\|^2$$
$$\vec{v}_l = \frac{\sum_{i=1}^n \delta(z_i, l) \vec{\phi}_i}{\sum_{i=1}^n \delta(z_i, l)}$$

# Kernel k-means

- We can compute the Gramm matrix as in the case of kernel PCA.

$$G^{ij} = -\frac{1}{2} \left( K^{ij} - \frac{1}{n} \sum_{i=1}^n K^{ij} - \frac{1}{n} \sum_{j=1}^n K^{ij} + \frac{1}{n^2} \sum_{j=1}^n \sum_{i=1}^n K^{ij} \right)$$

- How to compute the distance from each element to a cluster center?
  - We cannot compute directly  $\vec{v}_l = \frac{\sum_{i=1}^n \delta(z_i, l) \vec{\phi}_i}{\sum_{i=1}^n \delta(z_i, l)}$
  - But we can compute the value of  $\|\vec{\phi}_i - \vec{v}_l\|^2$

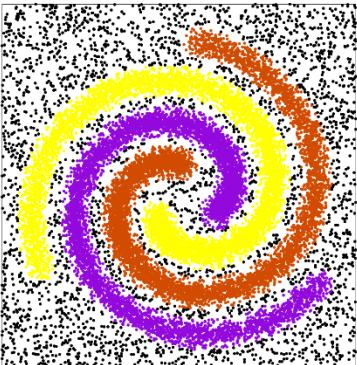
# Kernel k-means

$$\begin{aligned} d_{il}^2 &= \|\vec{\phi}_i - \vec{v}_l\|^2 = (\vec{\phi}_i - \vec{v}_l) \cdot (\vec{\phi}_i - \vec{v}_l) = \vec{\phi}_i \cdot \vec{\phi}_i - 2\vec{\phi}_i \cdot \vec{v}_l + \vec{v}_l \cdot \vec{v}_l \\ &= G^{ii} - 2 \frac{\sum_{j=1}^n \delta(z_i, l) G^{ij}}{\sum_{j=1}^n \delta(z_j, l)} + \frac{\sum_{j=1}^n \sum_{k=1}^n \delta(z_k, l) \delta(z_j, l) G^{kj}}{\left(\sum_{j=1}^n \delta(z_j, l)\right)^2} \end{aligned}$$

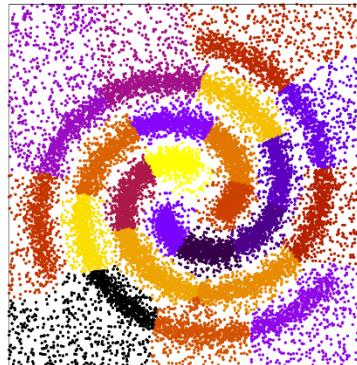
The algorithm:

1. Compute the Gramm matrix.
2. Randomly pick k centers.
3. Iterate the equation  $z_i = \operatorname{argmin}_l(d_{il}^2)$  until convergence

# Why $k$ -means?

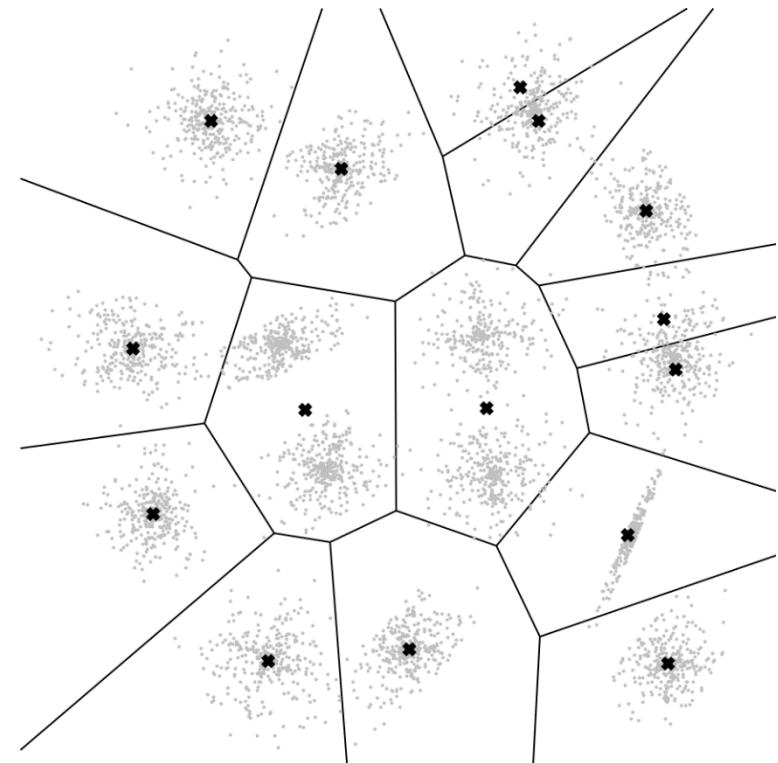


Expected result



K-means result

- K-means provides an optimal Voronoi partition of the data.
- Many other methods are somehow related to k-means.



# Fuzzy c-means: A fuzzy clustering algorithm

Bezdek, J. C., Ehrlich, R., & Full, W. (1984). FCM:  
The fuzzy c-means clustering  
algorithm. *Computers & Geosciences*, 10(2),  
191-203.

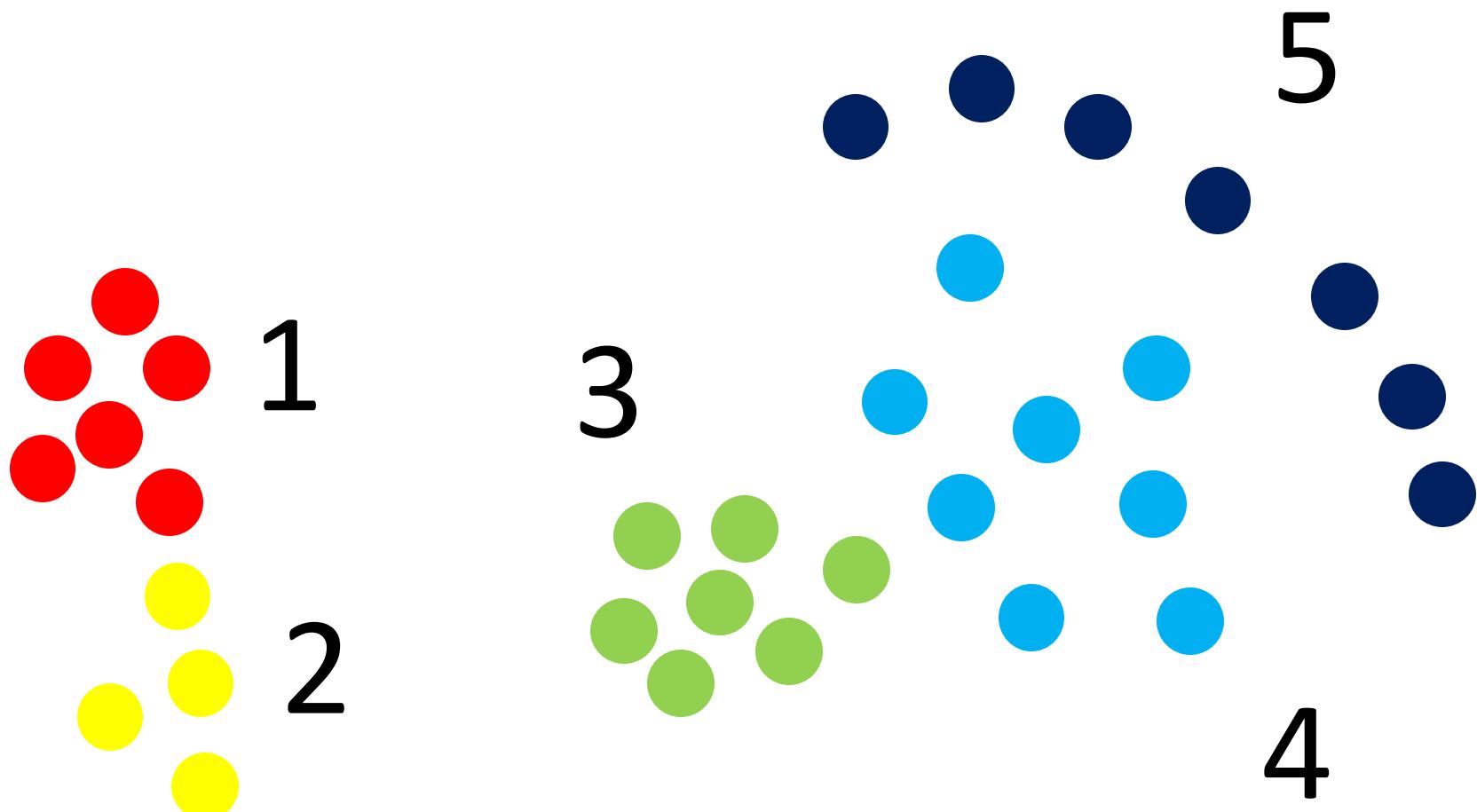
# Fuzzy c-means

- It can be considered a version of k-means with a soft assignation criteria.
- Just consider the one-hot encoding of the cluster assignation and relax the binary condition.

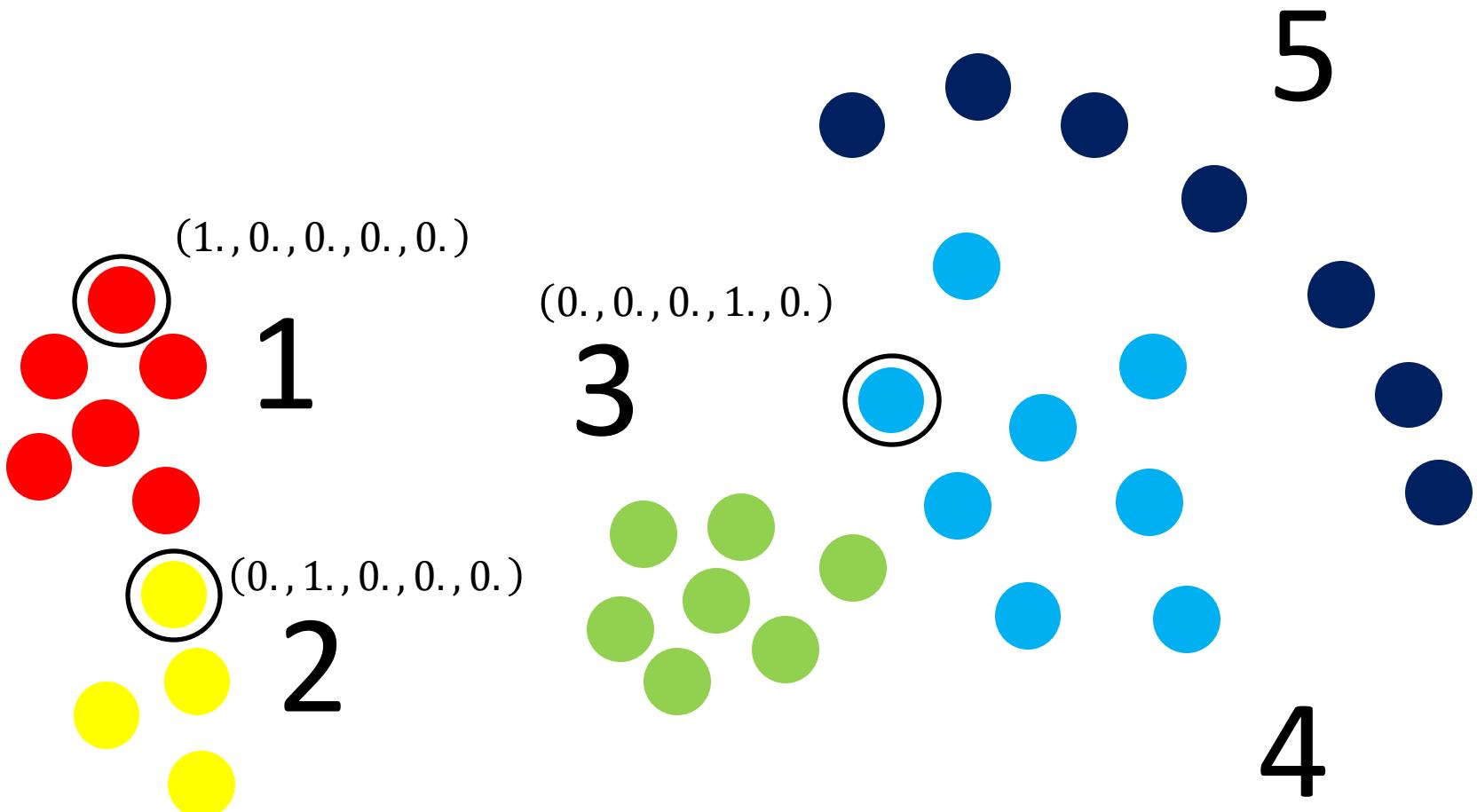
$$\vec{Cl}(i) = (u_1, u_2, \dots, u_l, \dots, u_k)$$
$$\sum_{l=1}^k u_l = 1$$

- Therefore, the objective function and the optimization algorithm must be adapted to fulfill these conditions.

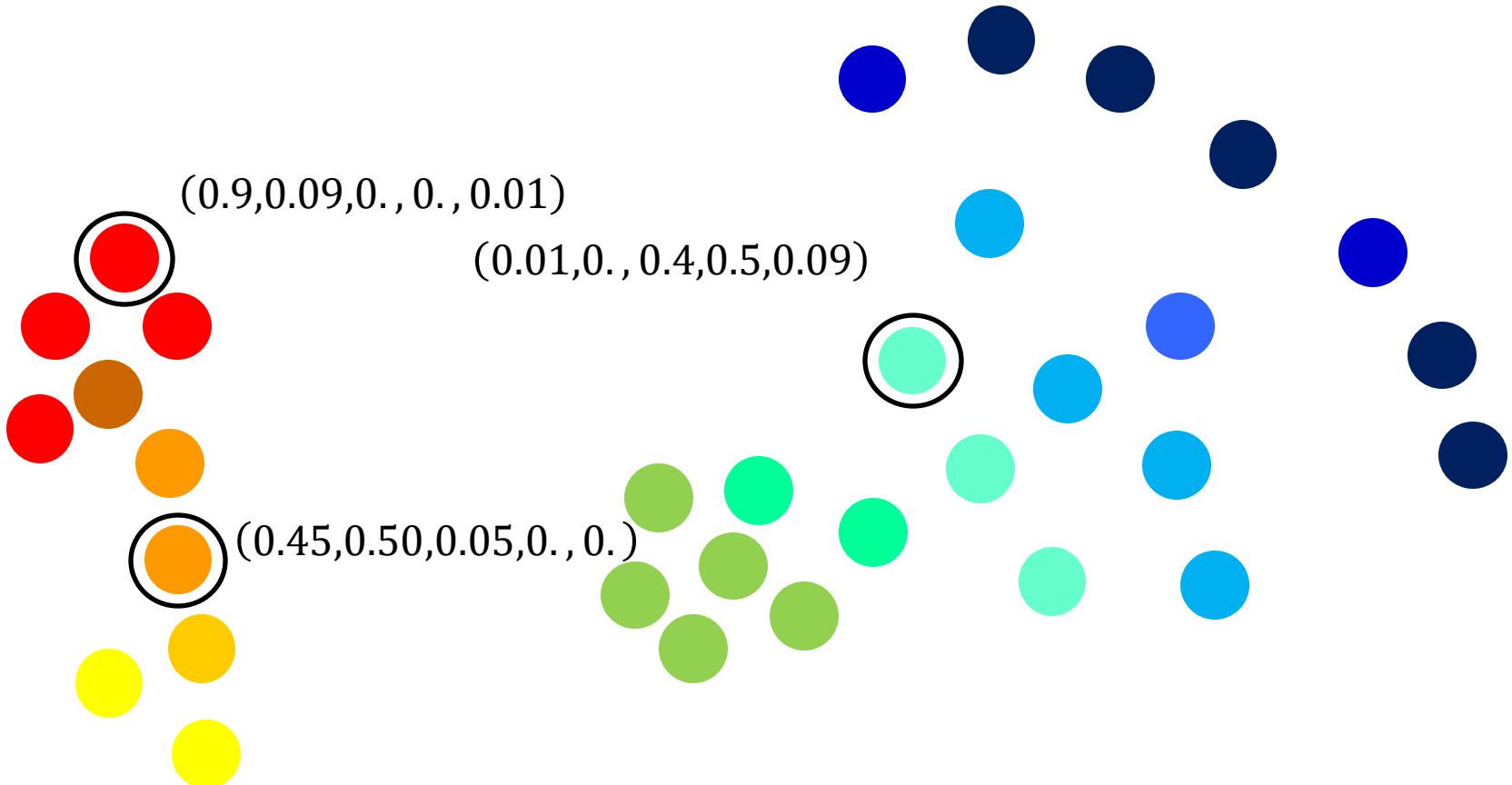
# K-means clustering



# K-means clustering one-hot encoding



# Fuzzy clustering



# Fuzzy c-means

- It can be considered a version of k-means with a soft assignation criteria.
- Just consider the one-hot encoding of the cluster assignation and relax the binary condition.

$$\vec{Cl}(i) = (u_1, u_2, \dots, u_l, \dots, u_k)$$
$$\sum_{l=1}^k u_l = 1$$

- Therefore, the objective function and the optimization algorithm must be adapted to fulfill these conditions.

# Fuzzy c-means algorithm

- Objective function:

$$O(\mathbb{U}) = \sum_{l=1}^k \sum_{i=1}^n (u_{il})^m \|\vec{x}_i - \vec{c}_l\|^2$$

- $\mathbb{U}$  is a  $n \times k$  matrix with the membership of the  $n$  elements in the  $k$  clusters.
- $m$  is the fuzzification parameter. Usually  $m=2$ .
- $\vec{c}_l$  is the vector with the coordinates of the  $l$  cluster center. As in the case of k-means, it is the average value of the cluster coordinates.  $\vec{c}_l = \frac{\sum_{i=1}^n (u_{il})^m \vec{x}_i}{\sum_{i=1}^n (u_{il})^m}$

# Fuzzy c-means algorithm

- Random initialization of  $\mathbb{U}$
- Iterative optimization:
  - Given  $\mathbb{U}$  compute the centers:

$$\vec{c}_l = \frac{\sum_{i=1}^n (u_{il})^m \vec{x}_i}{\sum_{i=1}^n (u_{il})^m}$$

- Given the centers, update  $\mathbb{U}$  :

$$u_{ij} = \frac{1}{\sum_{l=1}^k \left( \frac{\|\vec{x}_i - \vec{c}_j\|}{\|\vec{x}_i - \vec{c}_l\|} \right)^{2/m-1}}$$

- Repeat until the change in  $\mathbb{U}$  is smaller than a given threshold

# Fuzzy c-means algorithm

- Similar problems to k-means:
  - Initialization
  - Number of clusters
  - Sensitive to outliers
  - Spherical clusters

# Hierarchical clustering algorithms

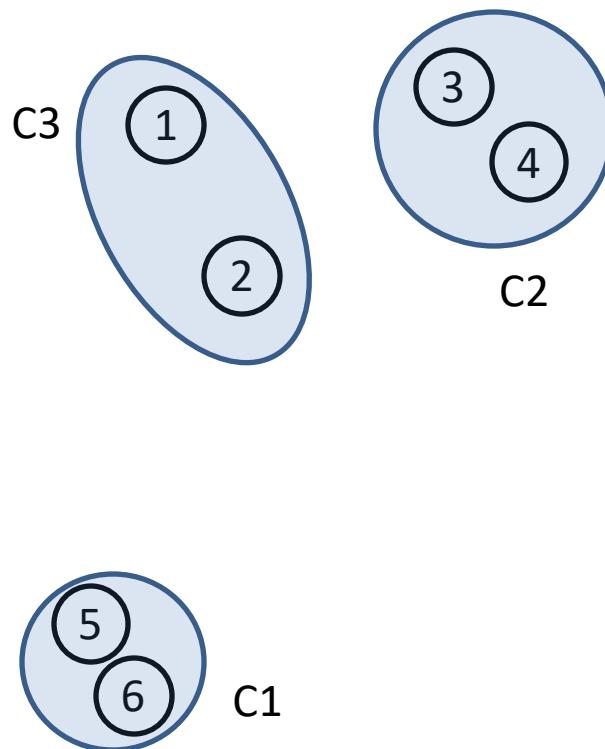
# Hierarchical Clustering

- Two main types of hierarchical clustering
  - **Agglomerative:**
    - Start with the points as individual clusters
    - At each step, merge the closest pair of clusters until only one cluster (or **k** clusters) are left
  - **Divisive:**
    - Start with one, all-inclusive cluster
    - At each step, split a cluster until each cluster contains a point (or there are **k** clusters)
- Traditional hierarchical algorithms use a similarity or distance matrix
  - Merge or split one cluster at a time

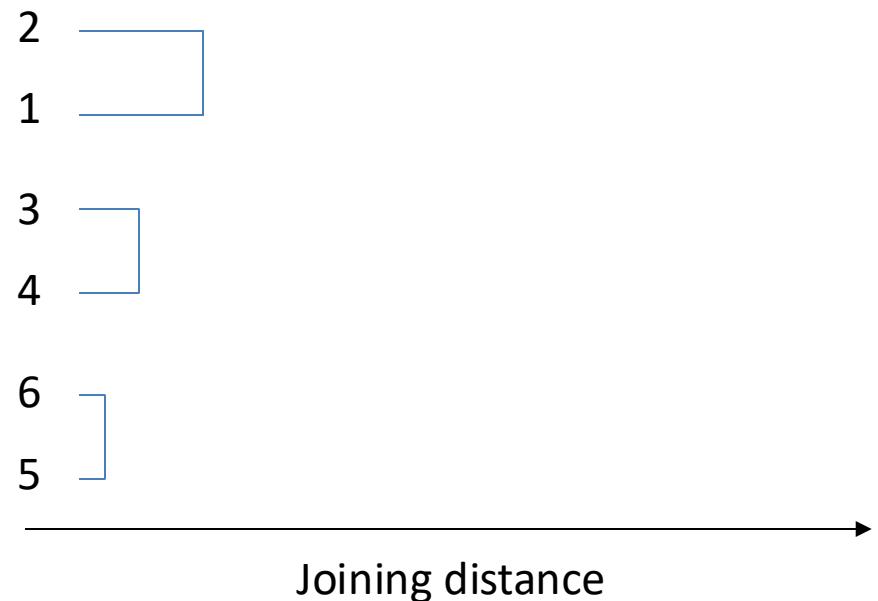
# Agglomerative clustering algorithm

- Most popular hierarchical clustering technique
- Basic algorithm
  1. Compute the distance matrix between the input data points
  2. Let each data point be a cluster
  3. **Repeat**
  4.       Merge the two closest clusters
  5.       Update the distance matrix
  6. **Until** only a single cluster remains

# Hierarchical clustering at glance

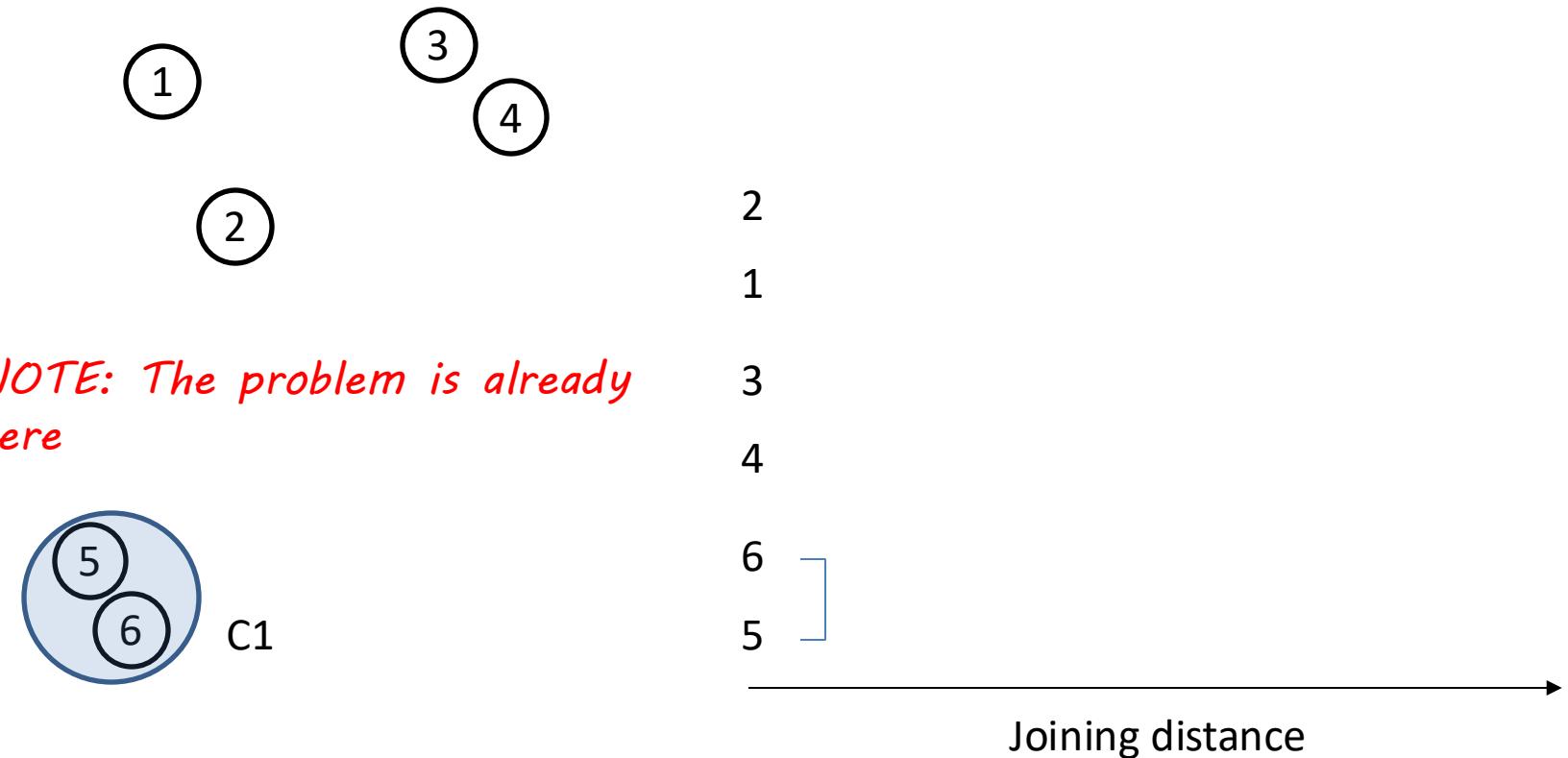


*How do we compute the distance between two clusters?*



**At each step: Pick the minimum distance and merge these elements**

# Hierarchical clustering at glance



**At each step: Pick the minimum distance and merge these elements**

# Agglomerative clustering algorithm

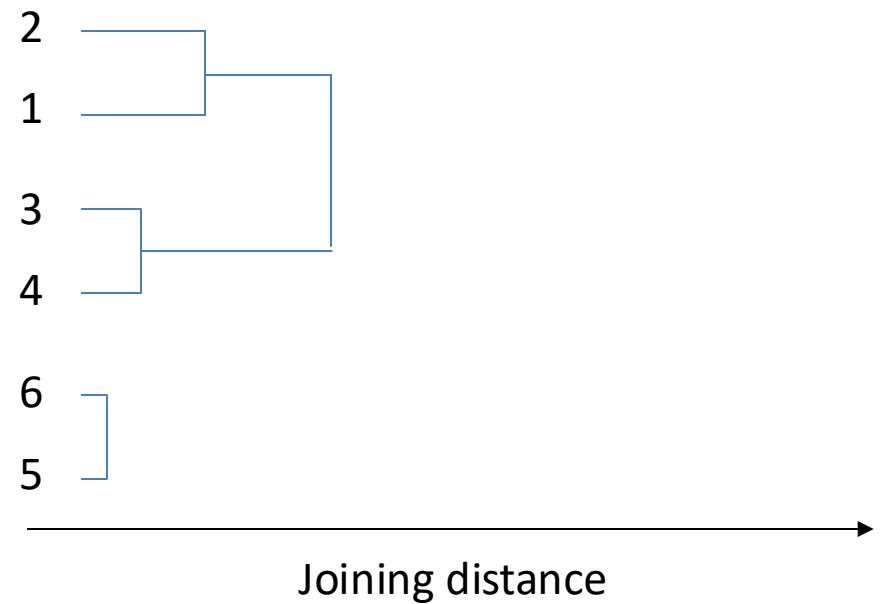
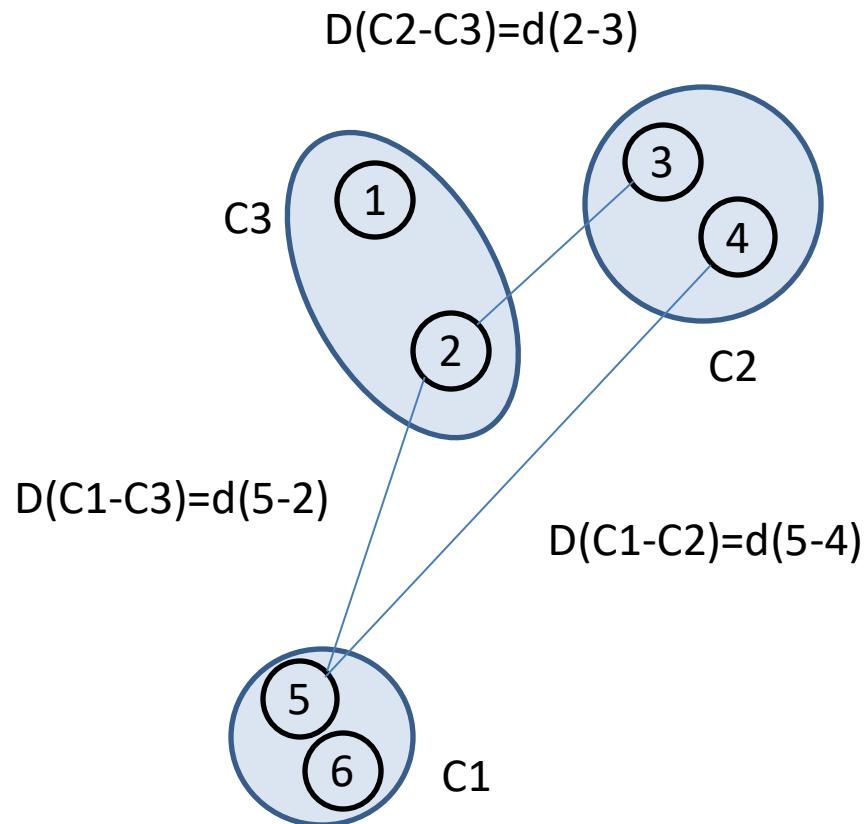
- Most popular hierarchical clustering technique
- Basic algorithm
  1. Compute the distance matrix between the input data points
  2. Let each data point be a cluster
  3. **Repeat**
  4. Merge the two closest clusters
  5. Update the distance matrix
  6. **Until** only a single cluster remains
- Key operation is the computation of the distance between two clusters
  - Different definitions of the distance between clusters lead to different algorithms

# Single-link distance

- **Single-link distance** between clusters  $C_i$  and  $C_j$  is the *minimum distance* between any object in  $C_i$  and any object in  $C_j$
- The distance is **defined by the two most similar (i.e nearest) objects**

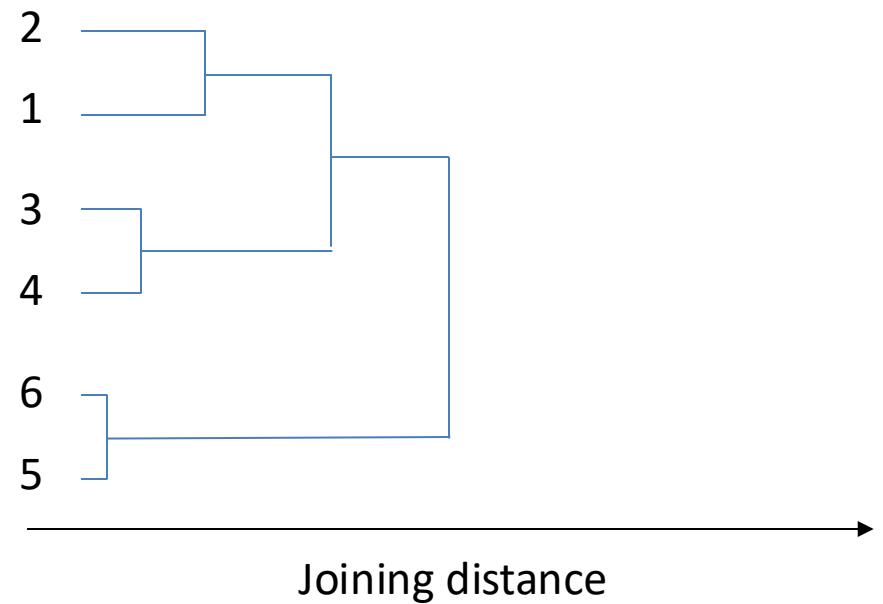
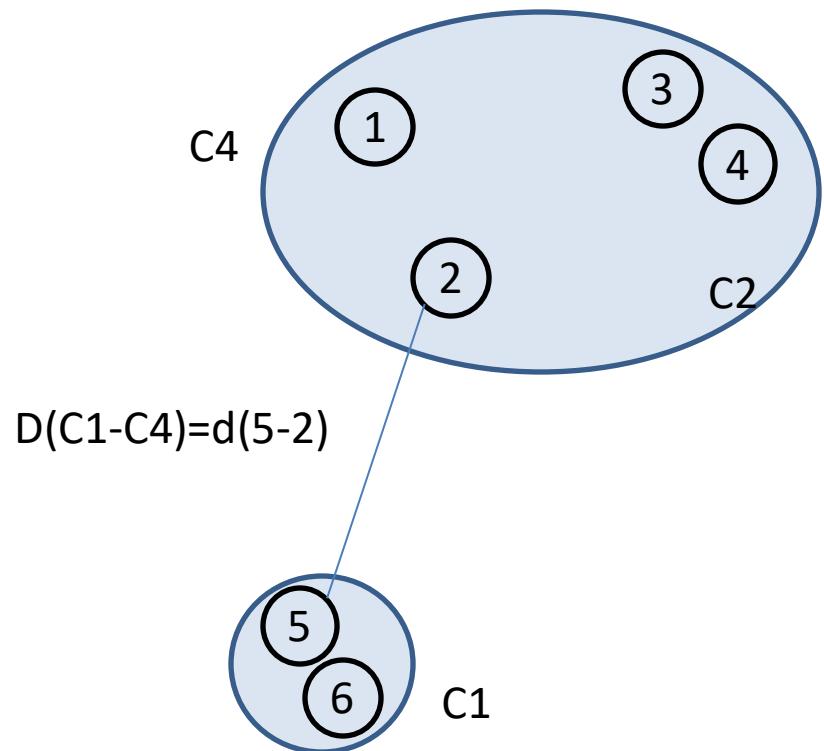
$$D_{sl}(C_i, C_j) = \min_{x,y} \left\{ d(x, y) \mid x \in C_i, y \in C_j \right\}$$

$$D_{sl}(C_i, C_j) = \min_{x,y} \left\{ d(x, y) \mid x \in C_i, y \in C_j \right\}$$



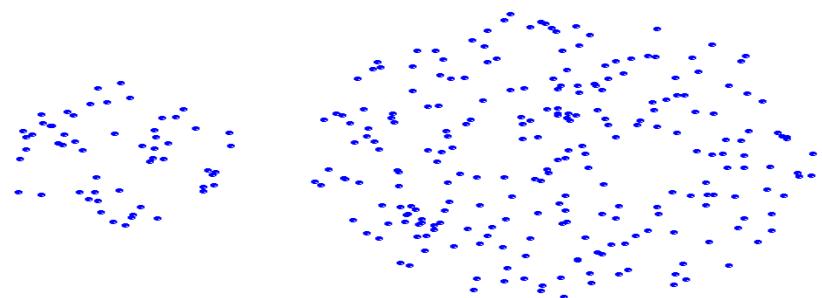
**At each step: Pick the minimum distance and merge these elements**

$$D_{sl}(C_i, C_j) = \min_{x,y} \{d(x, y) \mid x \in C_i, y \in C_j\}$$

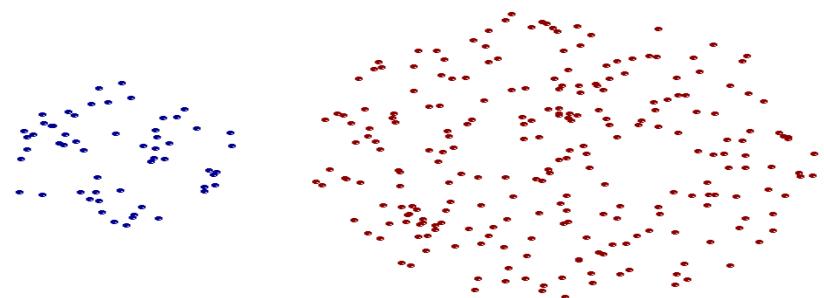


**At each step: Pick the minimum distance and merge these elements**

# Strengths of single-link clustering



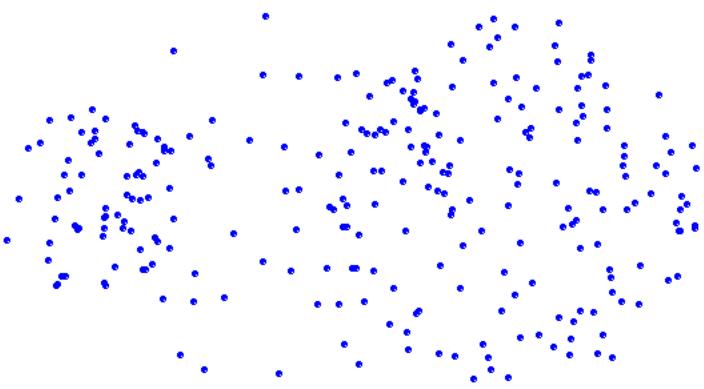
Original Points



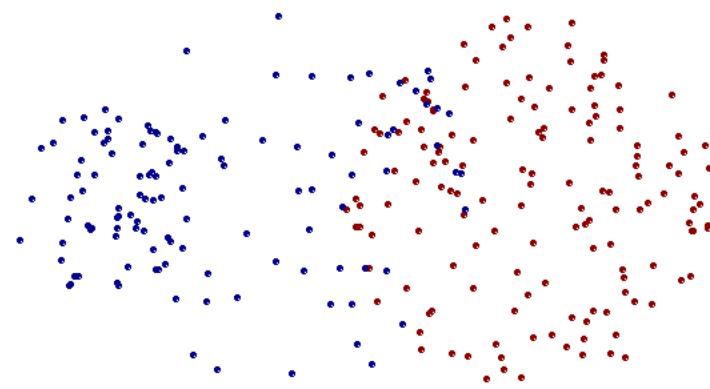
Two Clusters

- Can handle non-elliptical shapes

# Limitations of single-link clustering



Original Points



Two Clusters

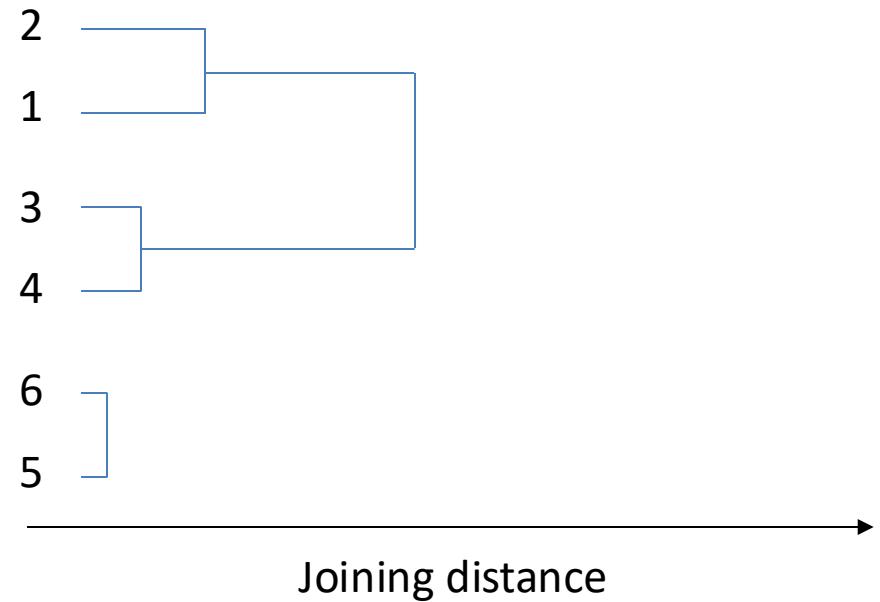
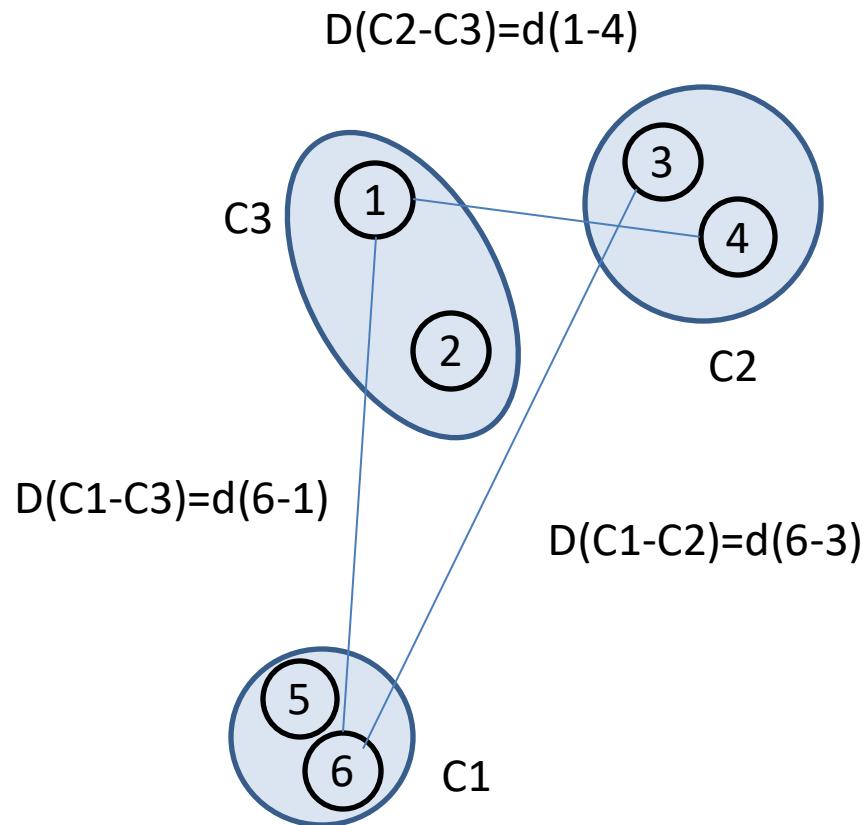
- Sensitive to noise and outliers
- It produces long, elongated clusters

# Complete-link distance

- **Complete-link distance** between clusters  $C_i$  and  $C_j$  is the *maximum distance* between any object in  $C_i$  and any object in  $C_j$
- The distance is **defined by the two most dissimilar (i.e. furthest) objects**

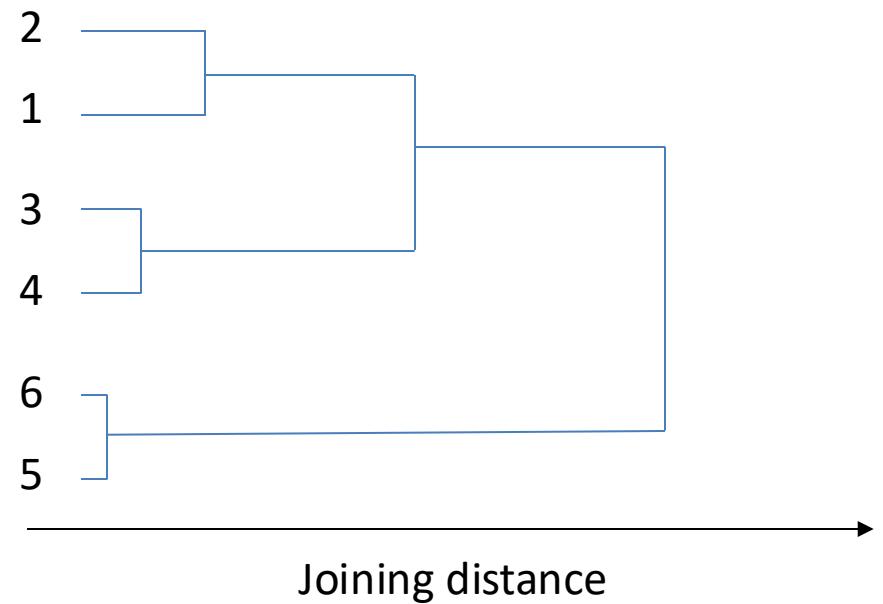
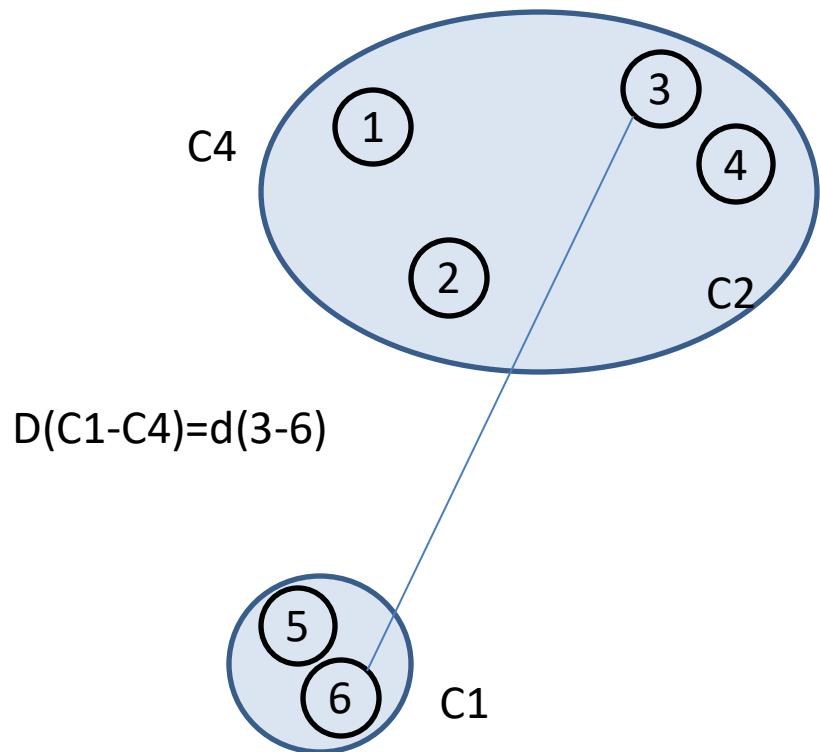
$$D_{cl}(C_i, C_j) = \max_{x,y} \{d(x, y) \mid x \in C_i, y \in C_j\}$$

$$D_{cl}(C_i, C_j) = \max_{x,y} \{d(x, y) \mid x \in C_i, y \in C_j\}$$



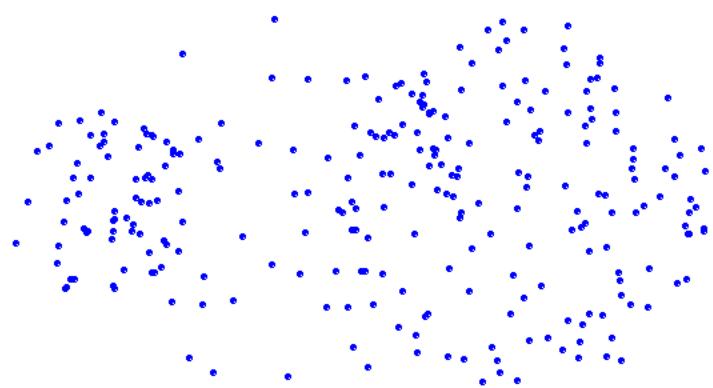
**At each step: Pick the minimum distance and merge these elements**

$$D_{cl}(C_i, C_j) = \max_{x,y} \{d(x, y) | x \in C_i, y \in C_j\}$$

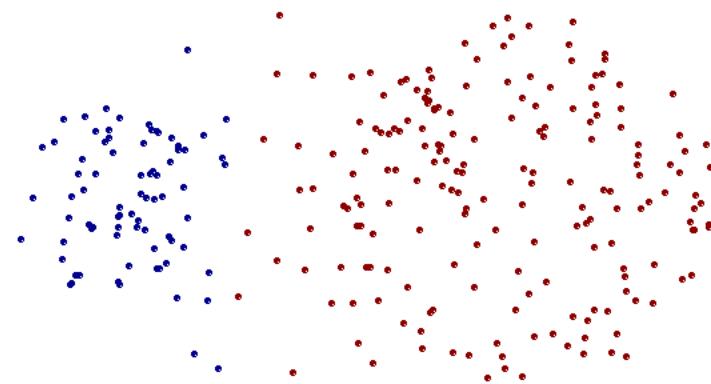


**At each step: Pick the minimum distance and merge these elements**

# Strengths of complete-link clustering



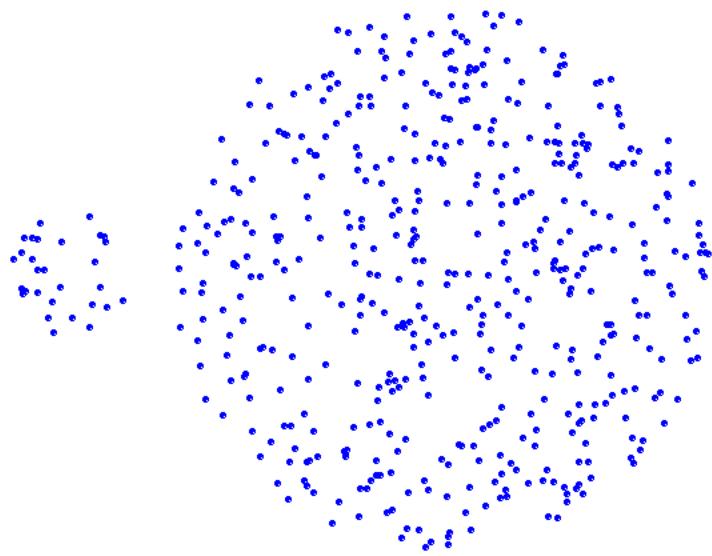
Original Points



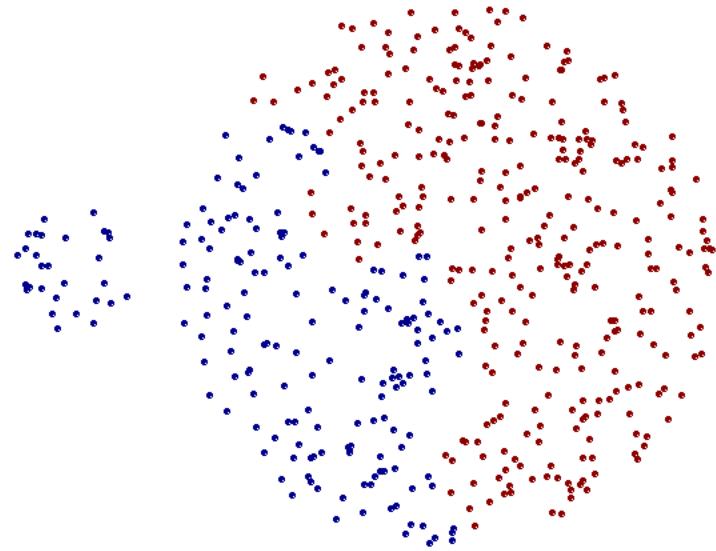
Two Clusters

- More balanced clusters (with equal diameter)
- Less susceptible to noise

# Limitations of complete-link clustering



Original Points



Two Clusters

- Tends to break large clusters
- All clusters tend to have the same diameter – small clusters are merged with larger ones

# Group average distance

- **Group average distance** between clusters  $C_i$  and  $C_j$  is the *average distance* between any object in  $C_i$  and any object in  $C_j$

$$D_{avg}(C_i, C_j) = \frac{1}{|C_i| \times |C_j|} \sum_{x \in C_i, y \in C_j} d(x, y)$$

# Average-link clustering: discussion

- Compromise between Single and Complete Link
- Strengths
  - Less susceptible to noise and outliers
- Limitations
  - Biased towards globular clusters

# Distance between two clusters

- **Centroid distance** between clusters  $C_i$  and  $C_j$  is the distance between the centroid  $r_i$  of  $C_i$  and the centroid  $r_j$  of  $C_j$

$$D_{centroids}(C_i, C_j) = d(r_i, r_j)$$

# Distance between two clusters

- **Ward's distance** between clusters  $C_i$  and  $C_j$  is the *difference* between the *total within cluster sum of squares for the two clusters separately*, and the *within cluster sum of squares resulting from merging the two clusters* in cluster  $C_{ij}$

$$D(C_i, C_j) = \frac{2n_i n_j}{n_i + n_j} D(r_i, r_j)$$

# Updating Ward's distance

- After merging  $C_i$  with  $C_j$ , the distance between the new cluster  $C_{i \cup j}$  and another cluster  $C_k$  is computed with the formula:

$$\begin{aligned} D(C_{i \cup j}, C_k) &= \\ &= \frac{n_i + n_k}{n_i + n_j + n_k} D(C_i, C_k) + \frac{n_j + n_k}{n_i + n_j + n_k} D(C_j, C_k) \\ &\quad - \frac{n_k}{n_i + n_j + n_k} D(C_i, C_j) \end{aligned}$$

# Ward's distance for clusters

- Similar to group average and centroid distance
- Less susceptible to noise and outliers
- Biased towards globular clusters
- Hierarchical analogue of k-means

# Some problems with the discussed algorithms

- K-means and fuzzy c-means are not well suited to deal with arbitrary shape (non-convex) clusters.
- Which intercluster distance shall we use for hierarchical clustering?
- What happens with noisy data sets?
- Which should be the final number of clusters?

# Reviews on clustering algorithms

- There are hundreds of clustering algorithms, each of them with their advantages and withdrawals.
  - Jain, A. K., Murty, M. N., & Flynn, P. J. (1999). Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3), 264-323.
  - Xu, R., & Wunsch, D. (2005). Survey of clustering algorithms. *Neural Networks, IEEE Transactions on*, 16(3), 645-678.
  - Jain, A. K. (2010). Data clustering: 50 years beyond K-means. *Pattern recognition letters*, 31(8), 651-666.
  - Xu, D., & Tian, Y. (2015). A Comprehensive Survey of Clustering Algorithms. *Annals of Data Science*, 2(2), 165-193.
  - Amit Saxena, *et al* (2017). A review of clustering techniques and developments, *Neurocomputing*, 267, 664-681.

# A selected list of clustering algorithms

- K-means
- Fuzzy c-means
- Agglomerative hierarchical
- Gaussian Mixture
- Density Based Algorithms:
  - DBSCAN
  - Density Peaks
  - Mean-Shift clustering
- Affinity Propagation
- Spectral Clustering

# A selected list of clustering algorithms

- K-means
  - Fuzzy c-means
  - Agglomerative hierarchical
- 
- Gaussian Mixture
  - Density Based Algorithms:
    - DBSCAN
    - Density Peaks
    - Mean-Shift clustering
  - Affinity Propagation
  - Spectral Clustering

Classical

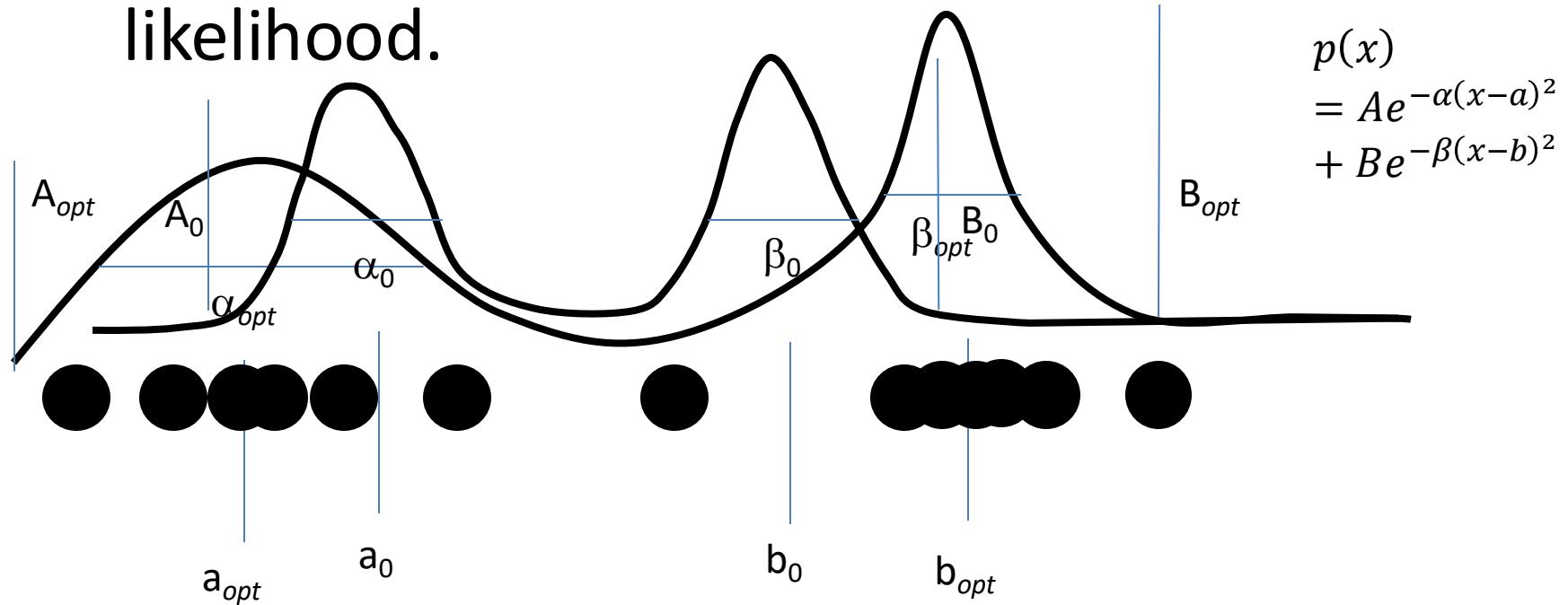
Modern

# A selected list of clustering algorithms

- K-means
- Fuzzy c-means
- Agglomerative hierarchical
- **Gaussian Mixture**
- Density Based Algorithms:
  - DBSCAN
  - Density Peaks
  - Mean-Shift clustering
- Affinity Propagation
- Spectral Clustering

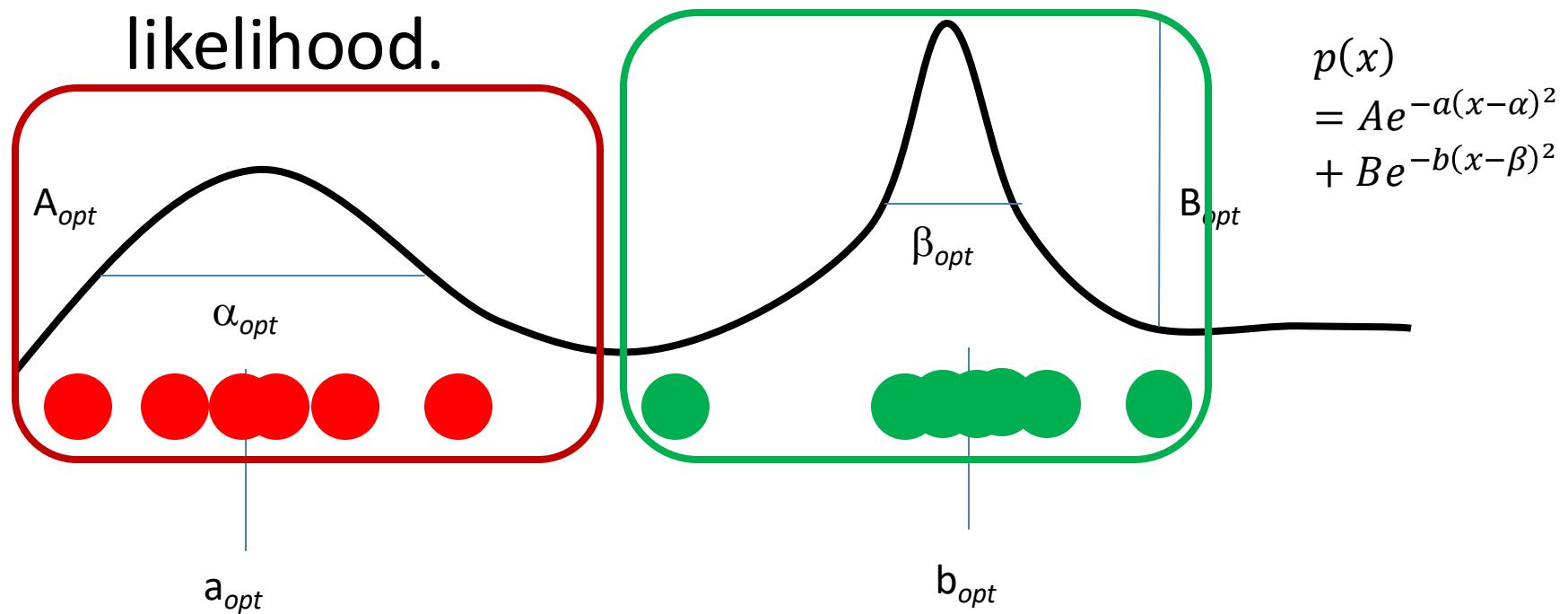
# Model based clustering

- Consider your data as a set of realizations of an underlying probability function  $p(x)$ .
- Assume the functional form of  $p(x)$  and estimate its parameters by maximum likelihood.



# Model based clustering

- Consider your data as a set of realizations of an underlying probability function  $p(x)$ .
- Assume the functional form of  $p(x)$  and estimate its parameters by maximum likelihood.



# Expectation-maximization algorithm

- Iterative procedure to compute the ***Maximum Likelihood (ML)*** estimate – even in the presence of missing or hidden data
- EM consists of two steps:
  - **Expectation step:** the (missing) data are estimated given the observed data and current estimates of model parameters
  - **Maximization step:** The likelihood function is maximized under the assumption that the (missing) data are known

# EM algorithm for mixture of Gaussians

- What is a mixture of  $K$  Gaussians?

$$p(x) = \sum_{k=1}^K \pi_k F(x | \Theta_k)$$

with

$$\sum_{k=1}^K \pi_k = 1$$

and  $F(x | \Theta)$  is the Gaussian distribution with parameters  $\Theta = \{\mu, \Sigma\}$

# EM algorithm for mixture of Gaussians

- If all points  $x \in X$  are mixtures of  $K$  Gaussians then

$$p(X) = \prod_{i=1}^n p(x_i) = \prod_{i=1}^n \sum_{k=1}^K \pi_k F(x_i | \Theta_k)$$

- **Goal:** Find  $\pi_1, \dots, \pi_k$  and  $\Theta_1, \dots, \Theta_k$  such that  $P(X)$  is maximized
- Or,  $\ln(P(X))$  is maximized:

$$L(\Theta) = \sum_{i=1}^n \ln \left\{ \sum_{k=1}^K \pi_k F(x_i | \Theta_k) \right\}$$

# Mixtures of Gaussians -- notes

- Every point  $\mathbf{x}_i$  is *probabilistically* assigned (generated) to (by) the  $k$ -th Gaussian
- Probability that point  $\mathbf{x}_i$  is generated by the  $k$ -th Gaussian is

$$w_{ik} = \frac{\pi_k F(x_i | \Theta_k)}{\sum_{j=1}^K \pi_j F(x_i | \Theta_j)}$$

# Mixtures of Gaussians -- notes

- Every Gaussian (cluster)  $C_k$  has an effective number of points assigned to it  $N_k$

$$N_k = \sum_{i=1}^n w_{ik}$$

- With mean

$$\mu_k = \frac{1}{N_k} \sum_{i=1}^n w_{ik} x_i$$

- And variance

$$\Sigma_k = \frac{1}{N_k} \sum_{i=1}^n w_{ik} (x_i - \mu_k) x_i (x_i - \mu_k)^T$$

# EM for Gaussian Mixtures

- Initialize the means  $\mu_k$ , variances  $\Sigma_k$  ( $\Theta_k = (\mu_k, \Sigma_k)$ ) and mixing coefficients  $\pi_k$ , and evaluate the initial value of the loglikelihood
- **Expectation step:** Evaluate weights

$$w_{ik} = \frac{\pi_k F(x_i | \Theta_k)}{\sum_{j=1}^K \pi_j F(x_i | \Theta_j)}$$

# EM for Gaussian Mixtures

- **Maximization step:** Re-evaluate parameters

$$\mu_k^{new} = \frac{1}{N_k} \sum_{i=1}^n w_{ik} x_i$$

$$\Sigma_k^{new} = \frac{1}{N_k} \sum_{i=1}^n w_{ik} (x_i - \mu_k^{new}) x_i (x_i - \mu_k^{new})^T$$

$$\pi_k^{new} = \frac{N_k}{N}$$

- Evaluate  $L(\Theta^{new})$  and stop if converged

# EM characteristics

- Notice the similarity between EM for Normal mixtures and K-means: The expectation step is the assignment, while the maximization step is the update of the centers.
- If the model ( $k$ , functional form) is not realistic, neither will the results.
- It is not guaranteed to reach the global optimum.
- $k$  can be derived with the use of Bayesian algorithms (Considering the generation of new clusters as a Dirichlet Process)

# A selected list of clustering algorithms

- K-means
- Fuzzy c-means
- Agglomerative hierarchical
- Gaussian Mixture
- Density Based Algorithms:
  - **DBSCAN**
  - Density Peaks
  - Mean-Shift clustering
- Affinity Propagation
- Spectral Clustering

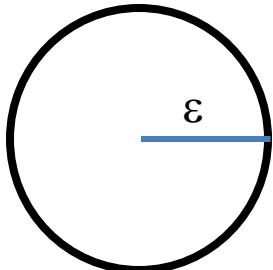
# DBSCAN

Density-Based Spatial Clustering of Applications with Noise  
(DBSCAN):

Density-based Clustering locates regions of high density that are separated from one another by regions of low density.

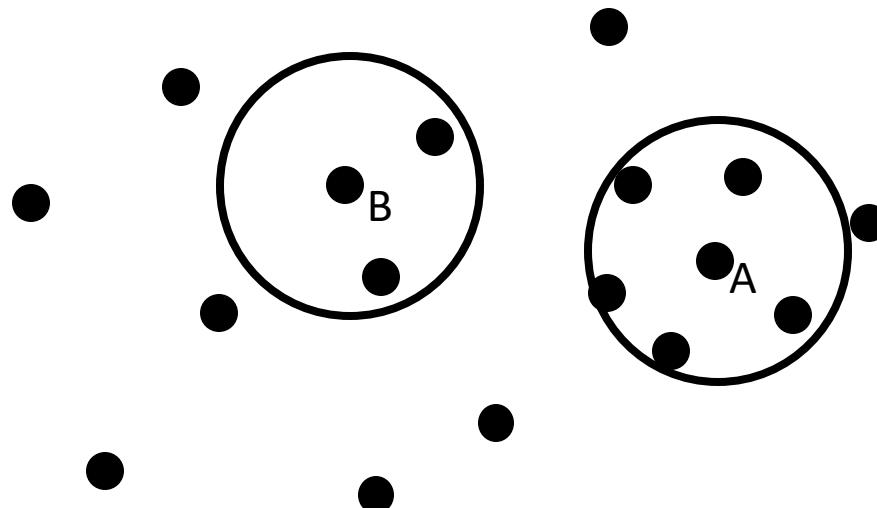
- Density = number of points within a specified radius (Eps)

# A route for this explanation



$\varepsilon$  / Eps is a method parameter.

$$\begin{aligned}\rho_A &= 6 \\ \rho_B &= 3\end{aligned}$$



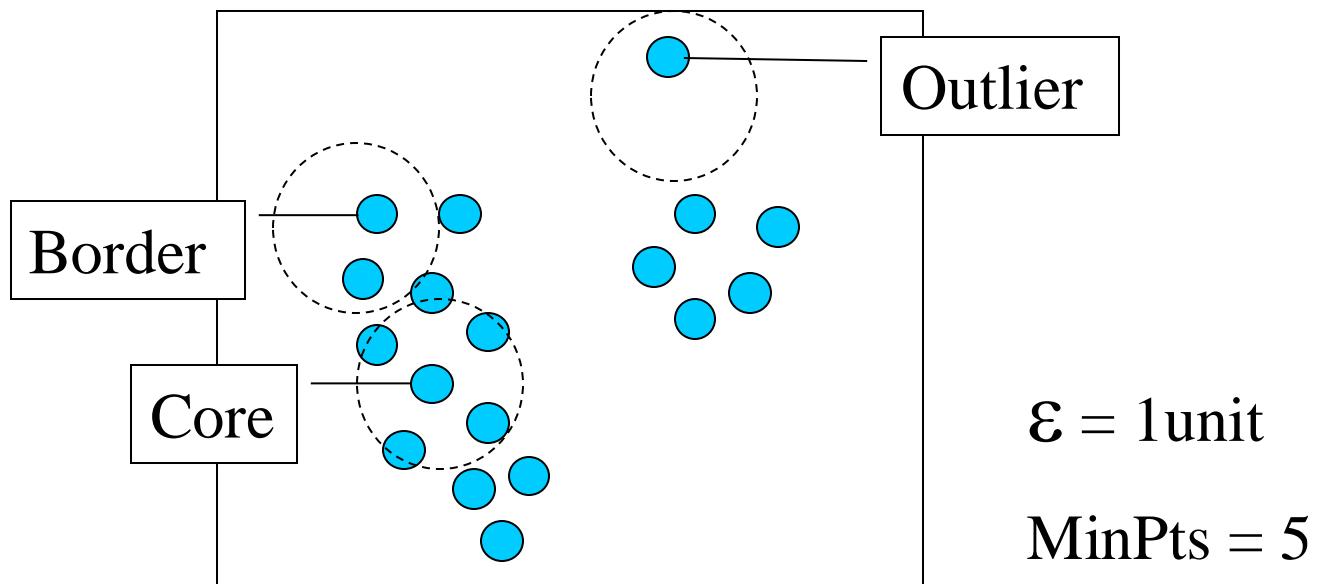
- We will classify the points according with their density.
- We will define a set of rules that exploit this classification in order to define the clusters.

## DBSCAN: Point classes

- A point is a **core point** if it has more than a specified number of points (MinPts) within Eps. These are points that are at the interior of a cluster
- A **border point** has fewer than MinPts within Eps, but is in the neighborhood of a core point
- A **noise point** is any point that is not a core point or a border point.

Note that we are adding a new method parameter: MinPts

# Border & Core

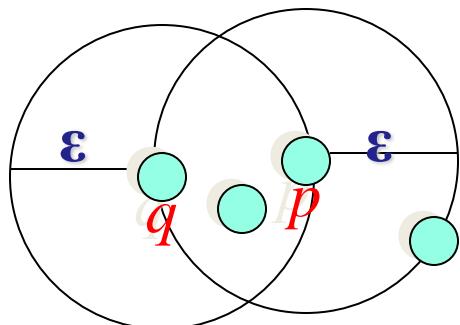


# DBSCAN: Ideas behind the rules

- Any two core points are close enough – within a distance  $Eps$  of one another – are put in the same cluster
- Any border point that is close enough to a core point is put in the same cluster as the core point
- Noise points are discarded

# Concepts: $\varepsilon$ -Neighborhood

- **$\varepsilon$ -Neighborhood** - Objects within a radius of  $\varepsilon$  from an object. (epsilon-neighborhood)
- **Core objects** -  $\varepsilon$ -Neighborhood of an object contains at least **MinPts** of objects

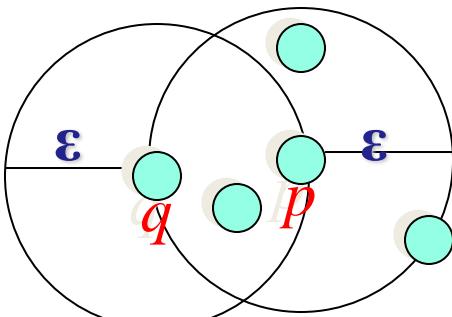


$\varepsilon$ -Neighborhood of  $p$   
 $\varepsilon$ -Neighborhood of  $q$

$p$  is a core object ( $\text{MinPts} = 4$ )

$q$  is not a core object

# Concepts: Reachability

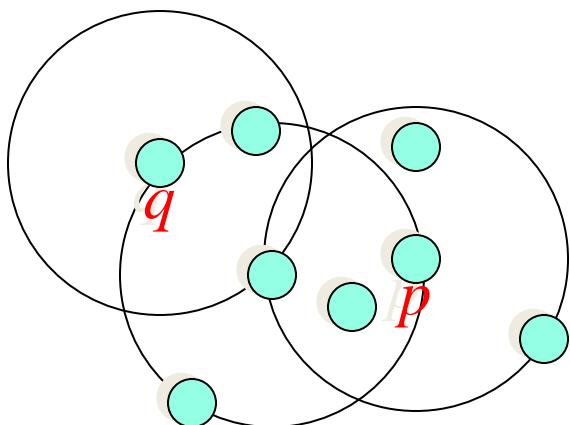
- **Directly density-reachable**
    - An object q is directly density-reachable from object p if q is within the  $\varepsilon$ -Neighborhood of p and p is a core object.
    - q is directly density-reachable from p
    - p is not directly density-reachable from q?
- 

# Concepts: Reachability

- **Density-reachable:**

- An object  $p$  is density-reachable from  $q$  w.r.t  $\epsilon$  and  $MinPts$  if there is a chain of objects  $p_1, \dots, p_n$ , with  $p_1=q$ ,  $p_n=p$  such that  $p_{i+1}$  is directly density-reachable from  $p_i$  w.r.t  $\epsilon$  and  $MinPts$  for all  $1 \leq i \leq n$

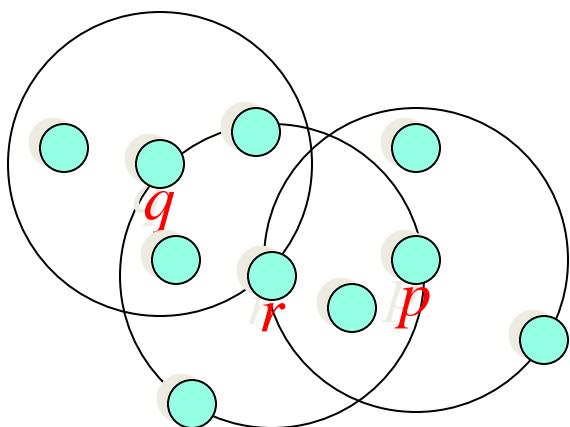
- $q$  is density-reachable from  $p$
    - $p$  is not density-reachable from  $q$ ?
    - Transitive closure of direct density-Reachability, asymmetric



# Concepts: Connectivity

- **Density-connectivity**

- Object  $p$  is density-connected to object  $q$  w.r.t  $\epsilon$  and  $MinPts$  if there is an object  $o$  such that both  $p$  and  $q$  are density-reachable from  $o$  w.r.t  $\epsilon$  and  $MinPts$

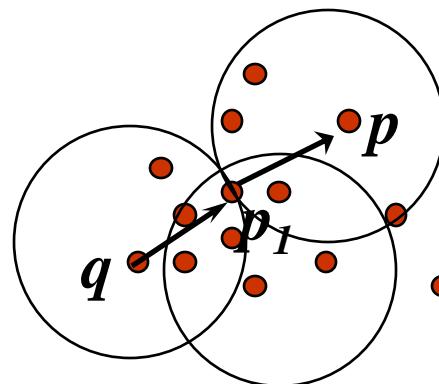


- $P$  and  $q$  are density-connected to each other by  $r$
- Density-connectivity is symmetric

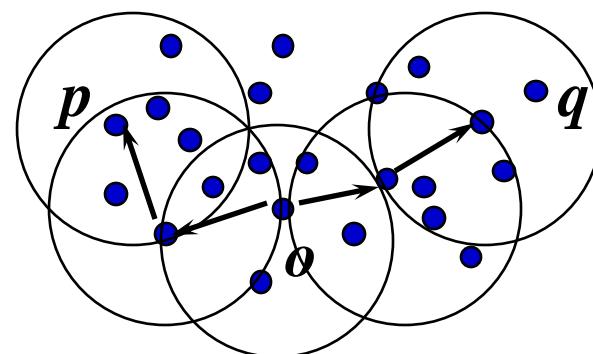
# Concepts: cluster & noise

- **Cluster:** a cluster  $\mathbf{C}$  in a set of objects  $\mathbf{D}$  w.r.t  $\varepsilon$  and  $MinPts$  is a non empty subset of  $\mathbf{D}$  satisfying
  - Maximality: For all  $p, q$  if  $p \in \mathbf{C}$  and if  $q$  is density-reachable from  $p$  w.r.t  $\varepsilon$  and  $MinPts$ , then also  $q \in \mathbf{C}$ .
  - Connectivity: for all  $p, q \in \mathbf{C}$ ,  $p$  is density-connected to  $q$  w.r.t  $\varepsilon$  and  $MinPts$  in  $\mathbf{D}$ .
  - **Note:** cluster contains *core objects* as well as *border objects*
- **Noise:** objects which are not directly density-reachable from at least one core object.

# (Indirectly) Density-reachable:



Density-connected



# DBSCAN: The Algorithm

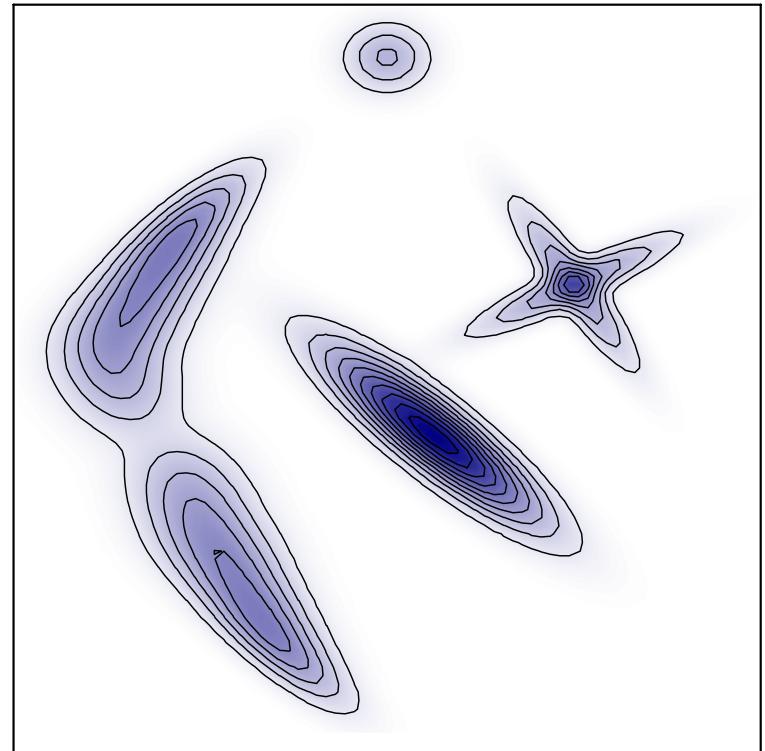
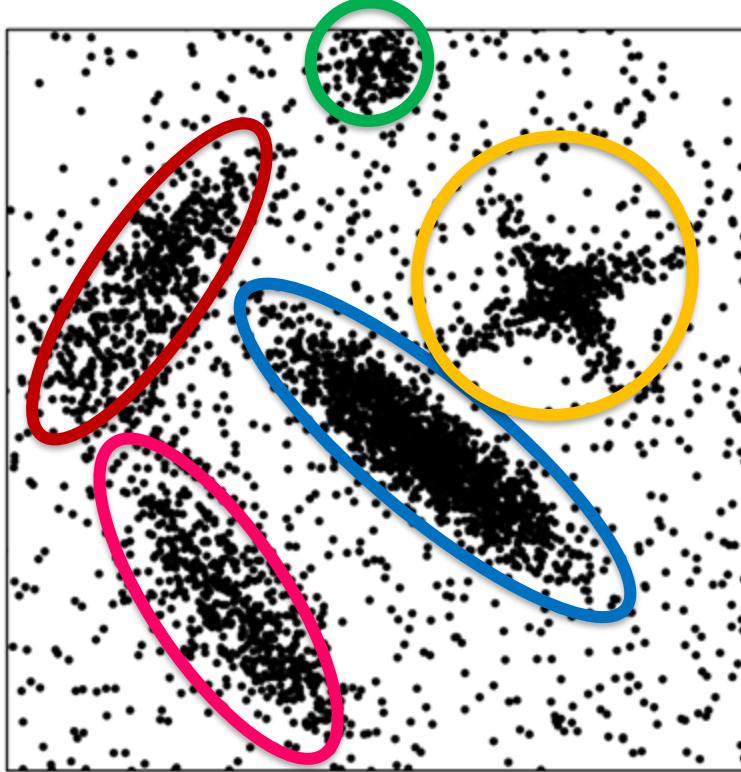
- select a point  $p$
- Retrieve all points density-reachable from  $p$  wrt  $\varepsilon$  and  $MinPts$ .
- If  $p$  is a core point, a cluster is formed.
- If  $p$  is a border point, no points are density-reachable from  $p$  and DBSCAN visits the next point of the database.
- Continue the process until all of the points have been processed.

Result is independent of the order of processing the points

# A selected list of clustering algorithms

- K-means
- Fuzzy c-means
- Agglomerative hierarchical
- Gaussian Mixture
- Density Based Algorithms:
  - DBSCAN
  - **Density Peaks**
  - Mean-Shift clustering
- Affinity Propagation
- Spectral Clustering

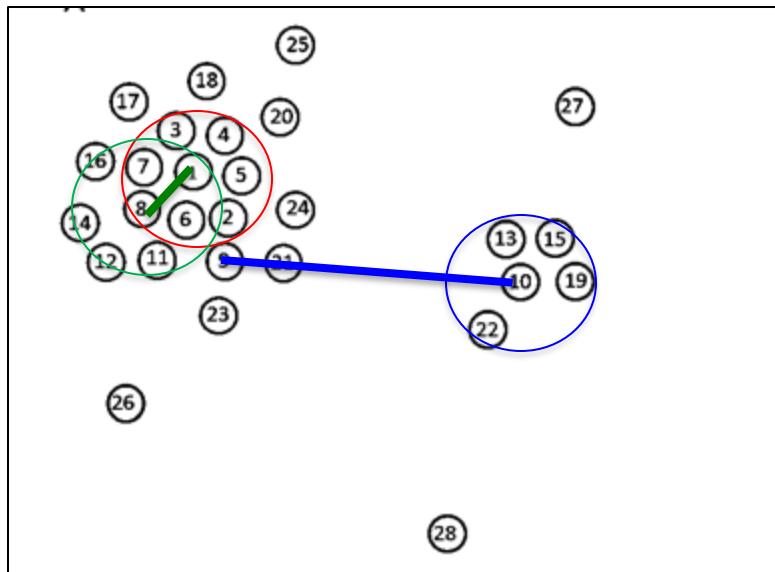
# Density Peaks clustering



Clusters = peaks in the density of points\*  
= peaks in the “mother” probability distribution

# Density Peaks: $\delta$ concept

Clustering in a 2-dimensional space



- 1) Compute the local density ( $\rho$ ) around each point

$$\rho(1)=7$$

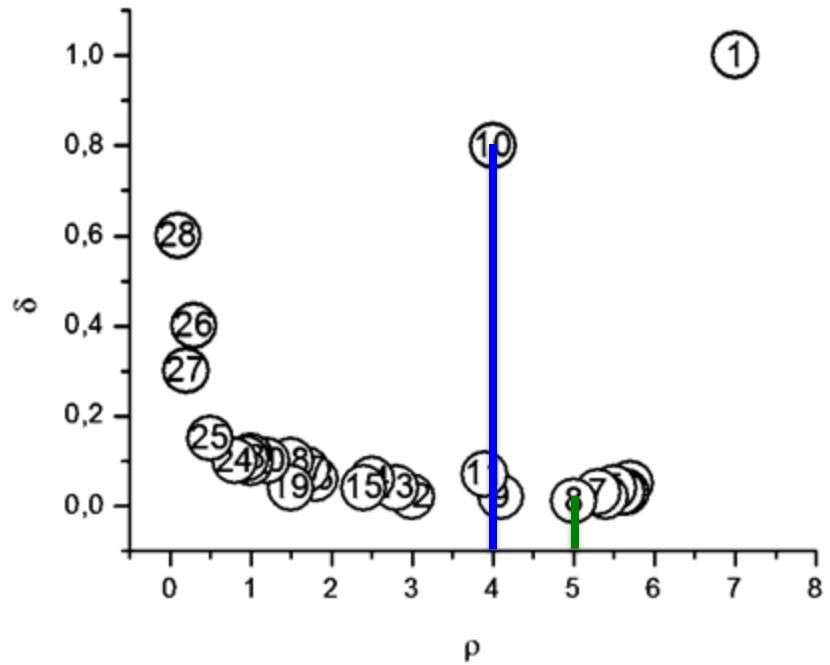
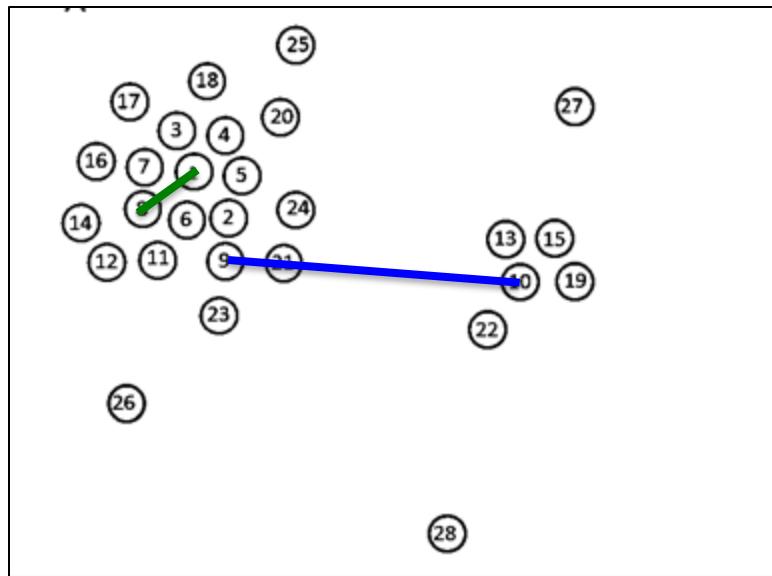
$$\rho(8)=5$$

$$\rho(10)=4$$

- 2) For each point compute the distance from all the points with higher density. Take the minimum value.

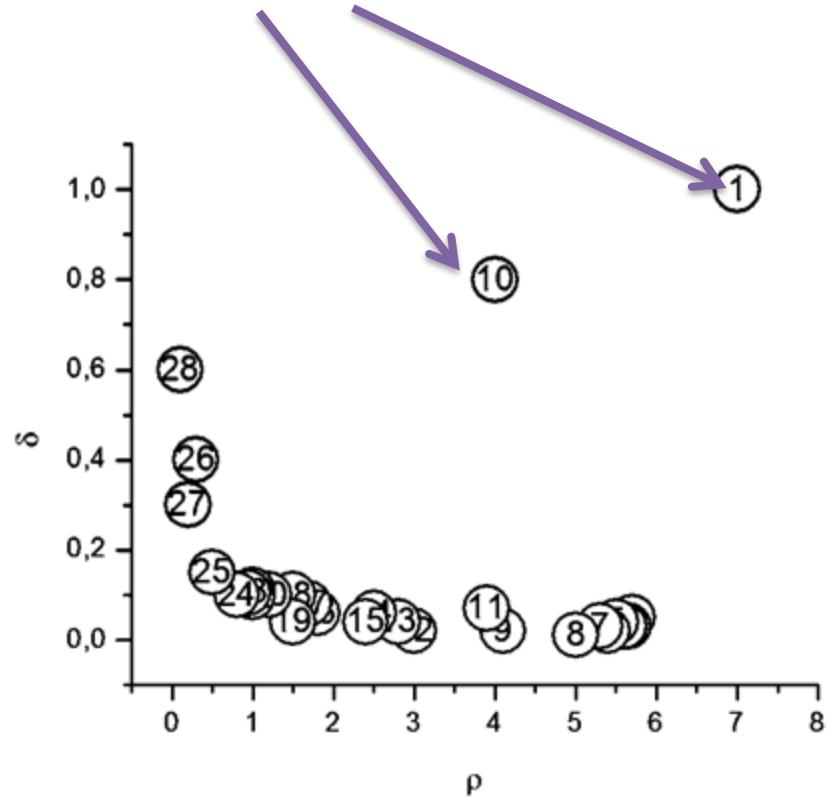
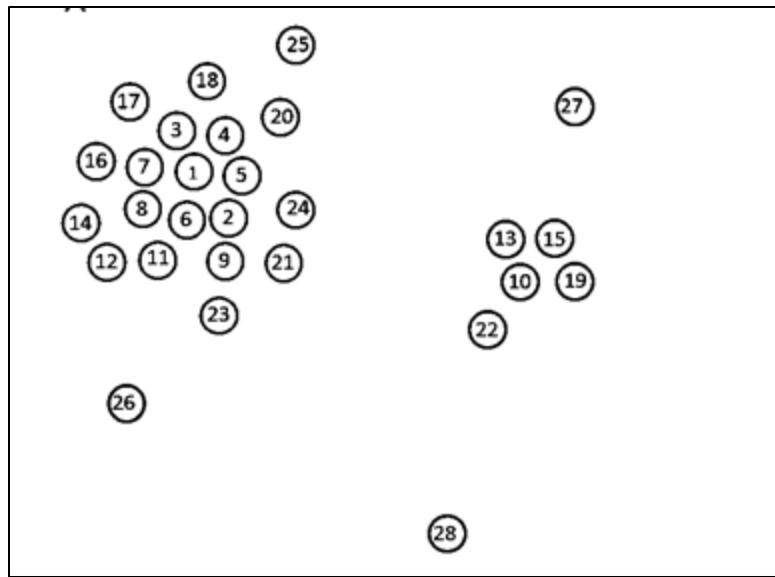
# Density Peaks decision graph

- 3) For each point, plot the minimum distance as a function of the density.



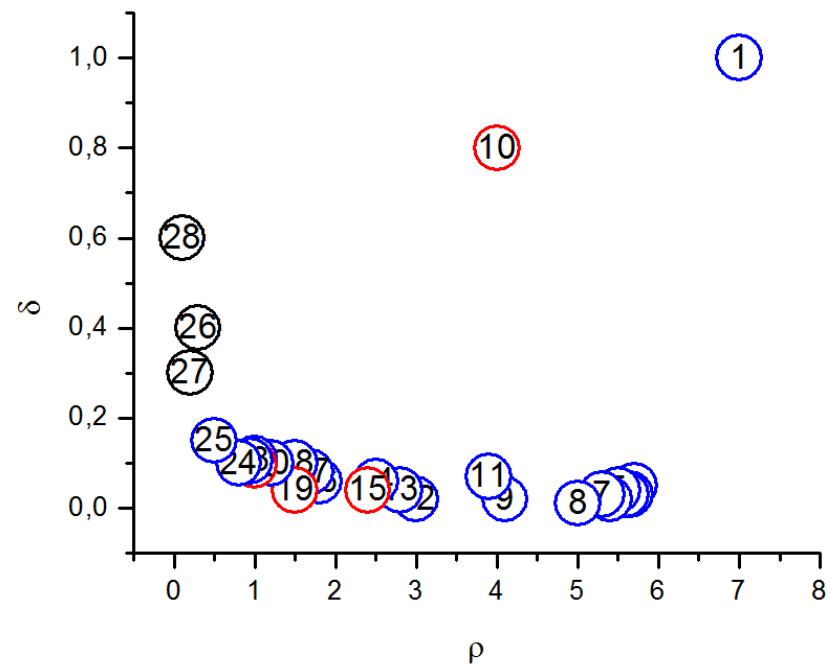
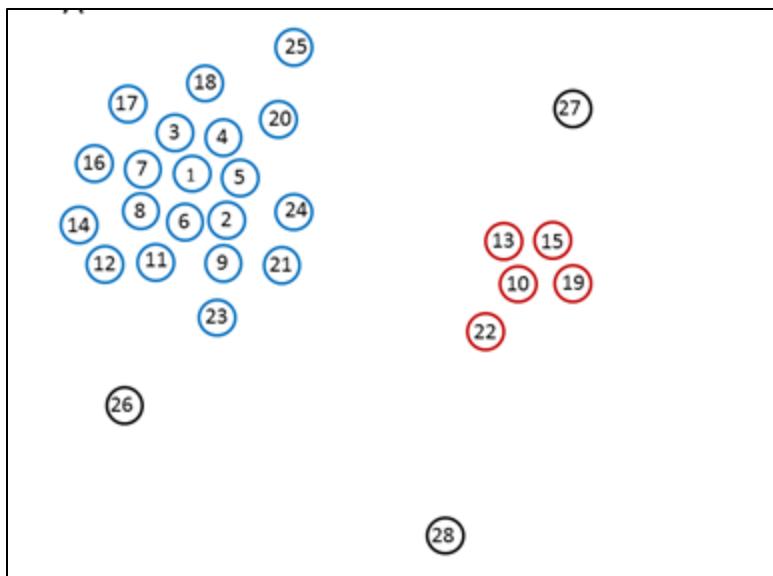
# Density Peaks decision graph

4) the “outliers” in this graph are the cluster centers

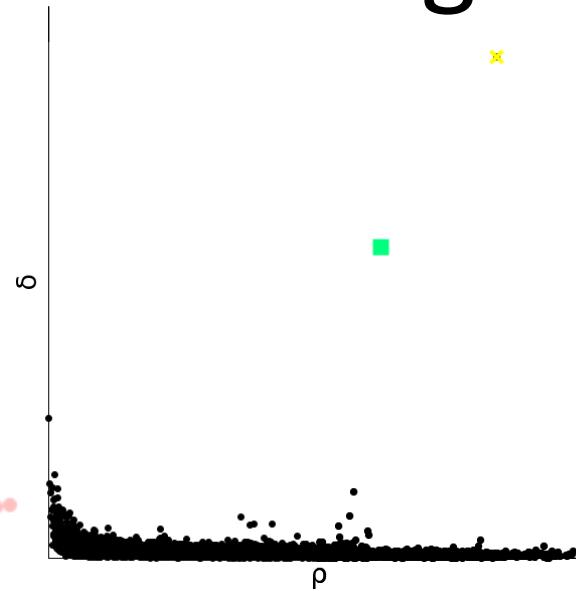


# Density Peaks decision graph

- 4) ) the “outliers” in this graph are the cluster centers
- 5) Assign each point to the same cluster of its nearest neighbor of higher density



# Density peaks clustering



- Compute the density ( $\rho$ ).
- Compute the distance from the nearest point with higher density ( $\delta$ ).
- Pick centers.
- Assign points.

# Density Peaks algorithm

Given a distance matrix  $d_{ij}$ , for each data point  $i$  compute:

$$\rho_i = \sum_j \chi(d_{ij} - d_c) \sim \sum_j e^{-\left(\frac{d_{ij}}{d_c}\right)^2} \quad (\text{number of data points within a distance } d_c \text{ or density estimation})$$

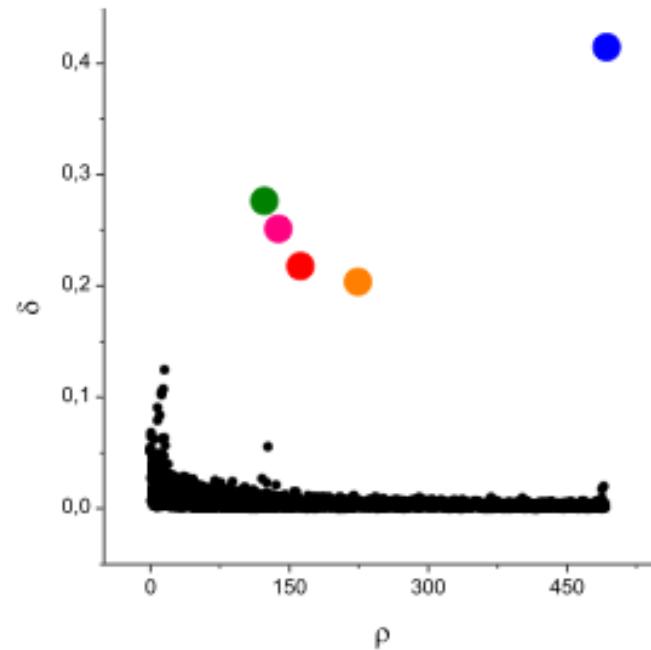
$$\delta_i = \min_{\rho_i < \rho_j} (d_{ij}) \quad (\text{distance of the closest data point of higher density})$$

Make the Decision graph:

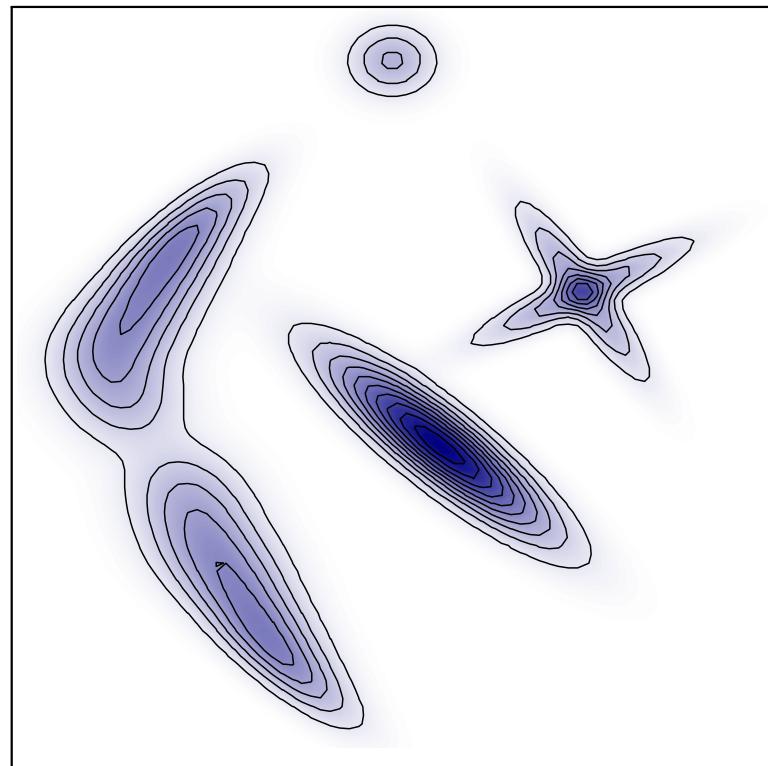
Plot  $\delta$  as a function of  $\rho$

One free parameter: the cutoff distance  $d_c$

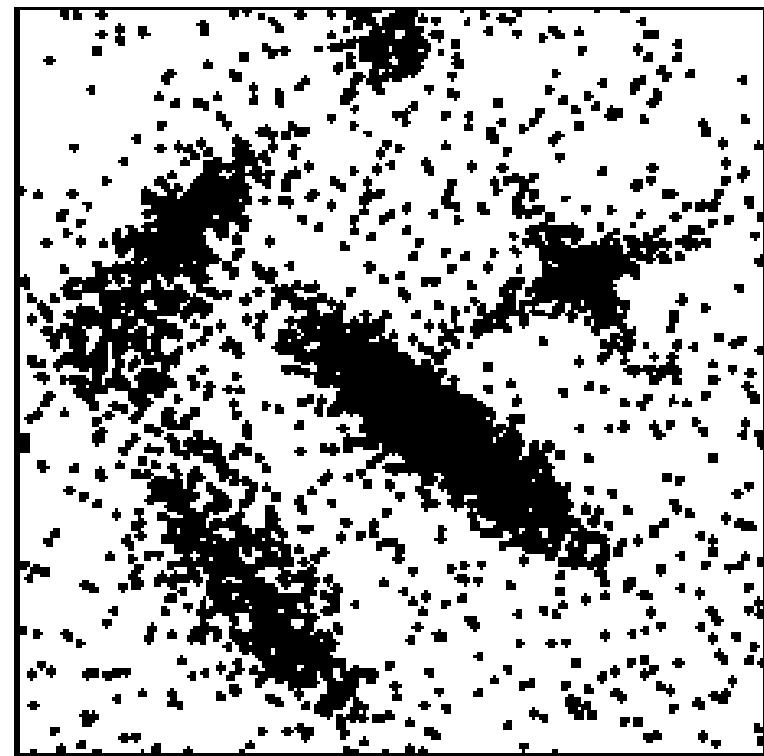
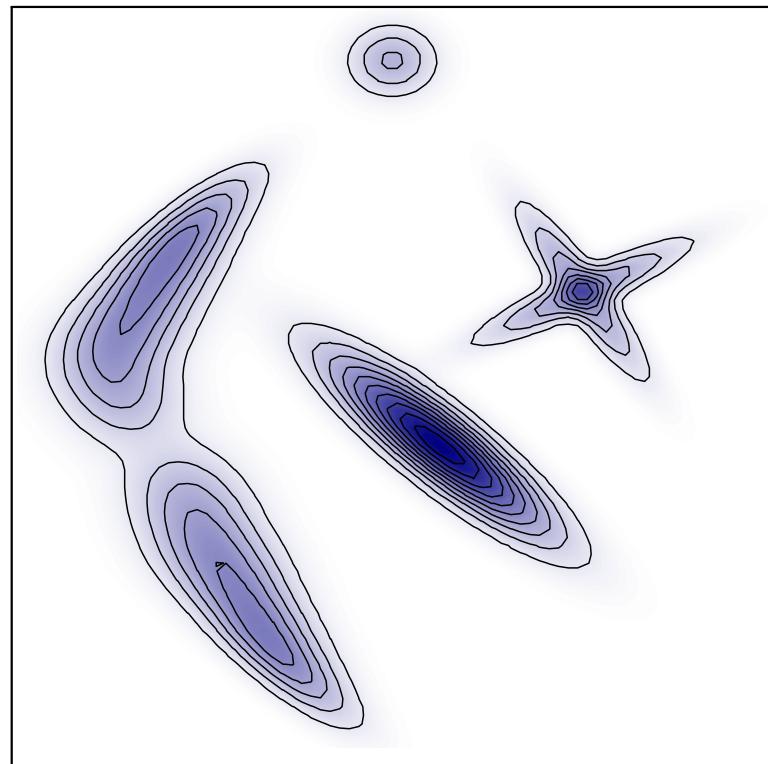
However: the method is only sensitive to the relative density of two data points, not to the absolute value of  $\rho$



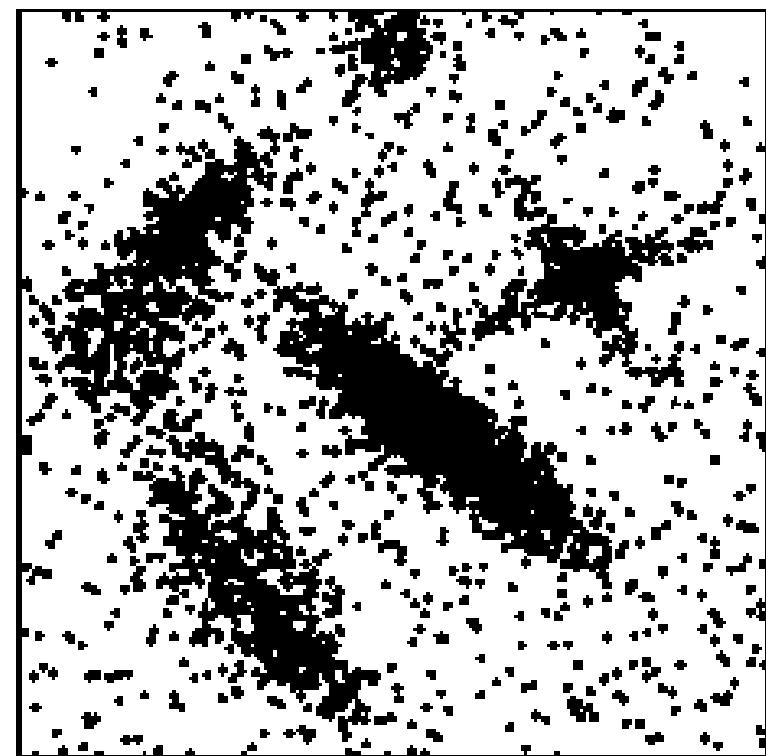
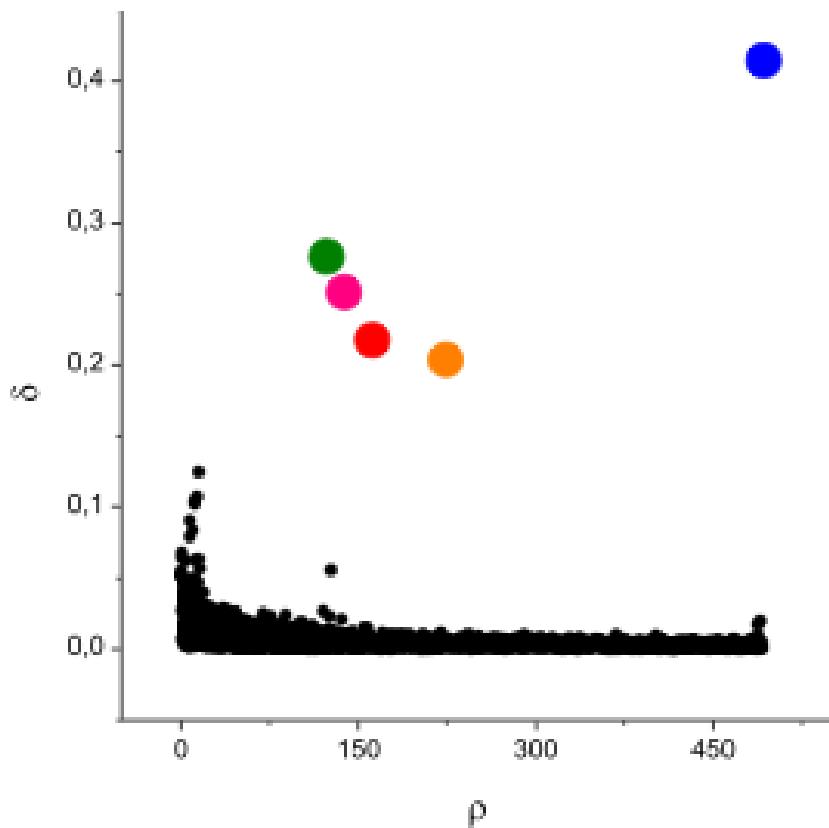
# Density Peaks clustering example



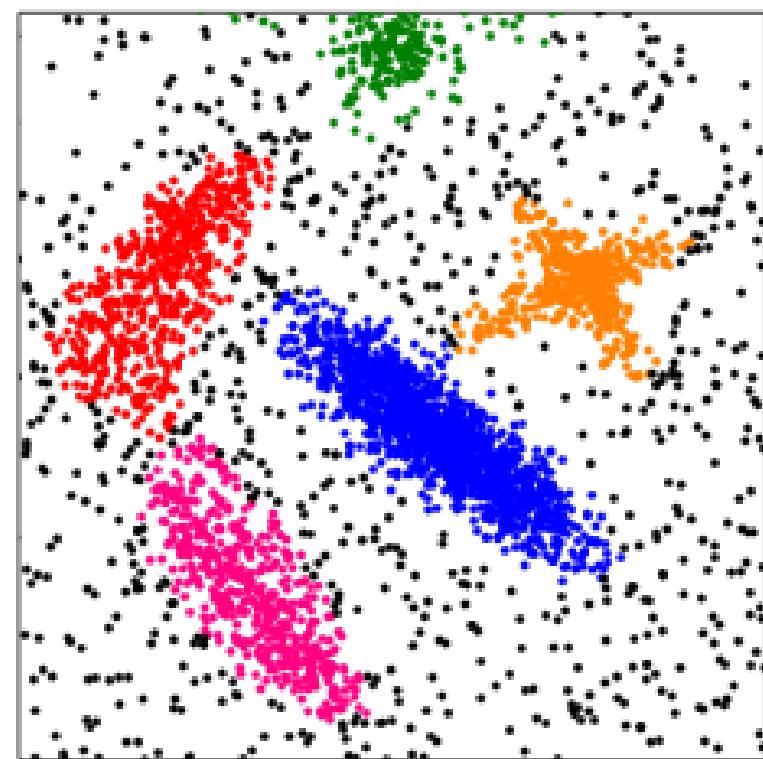
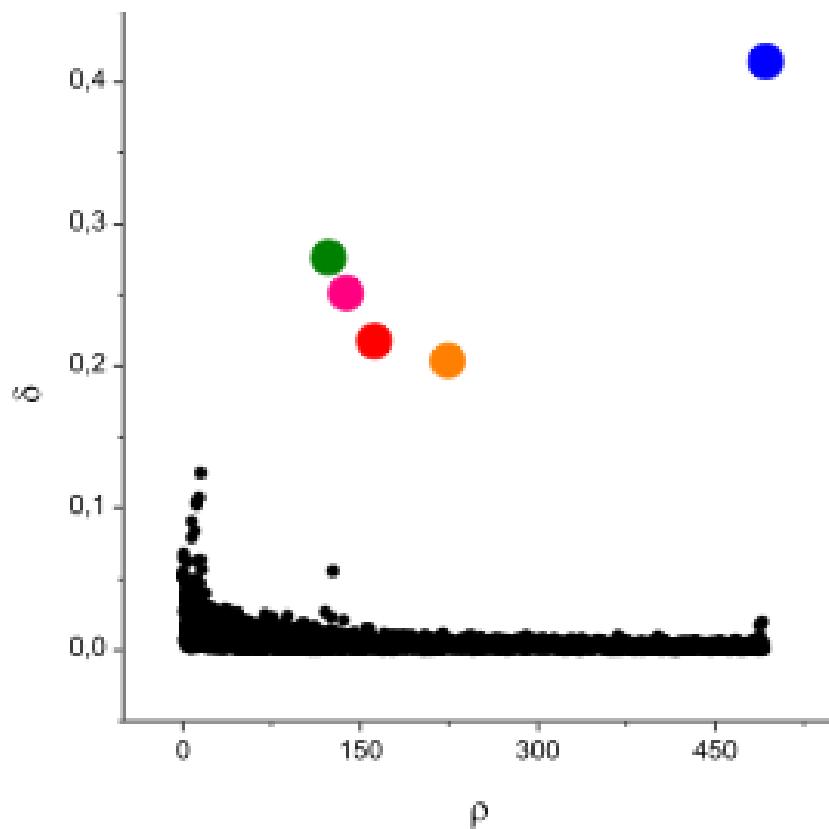
# Density Peaks clustering example



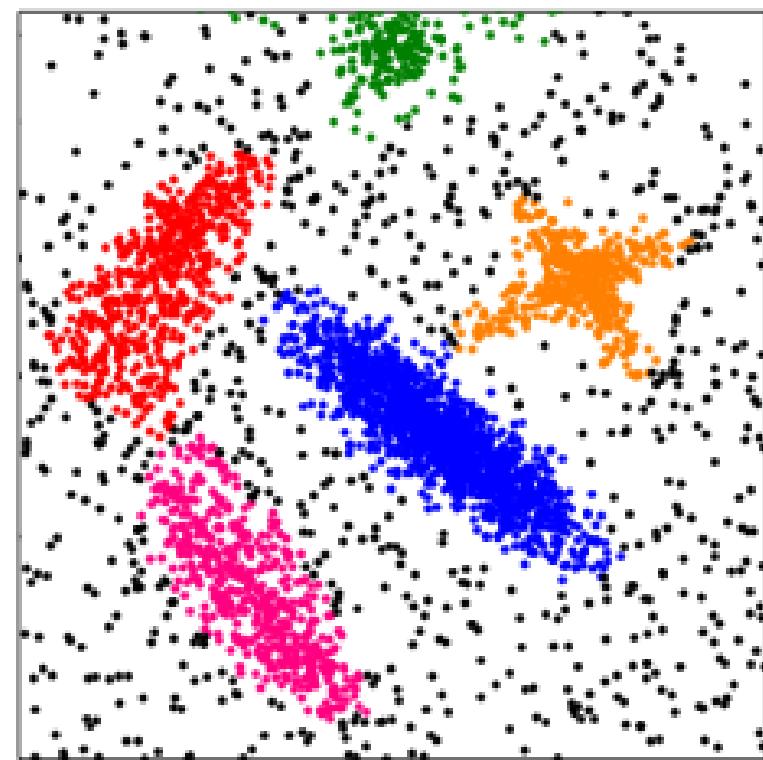
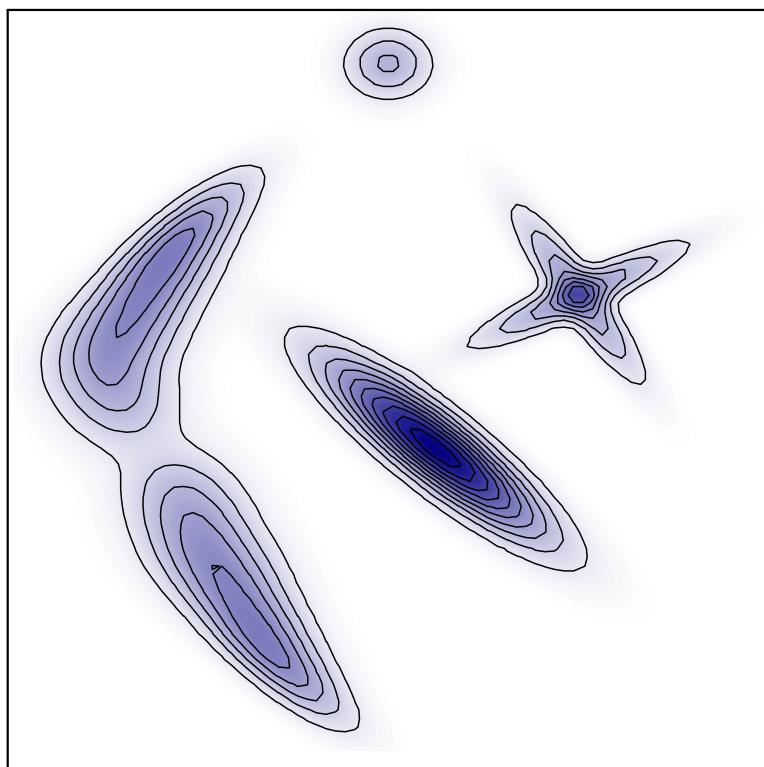
# Density Peaks clustering example



# Density Peaks clustering example

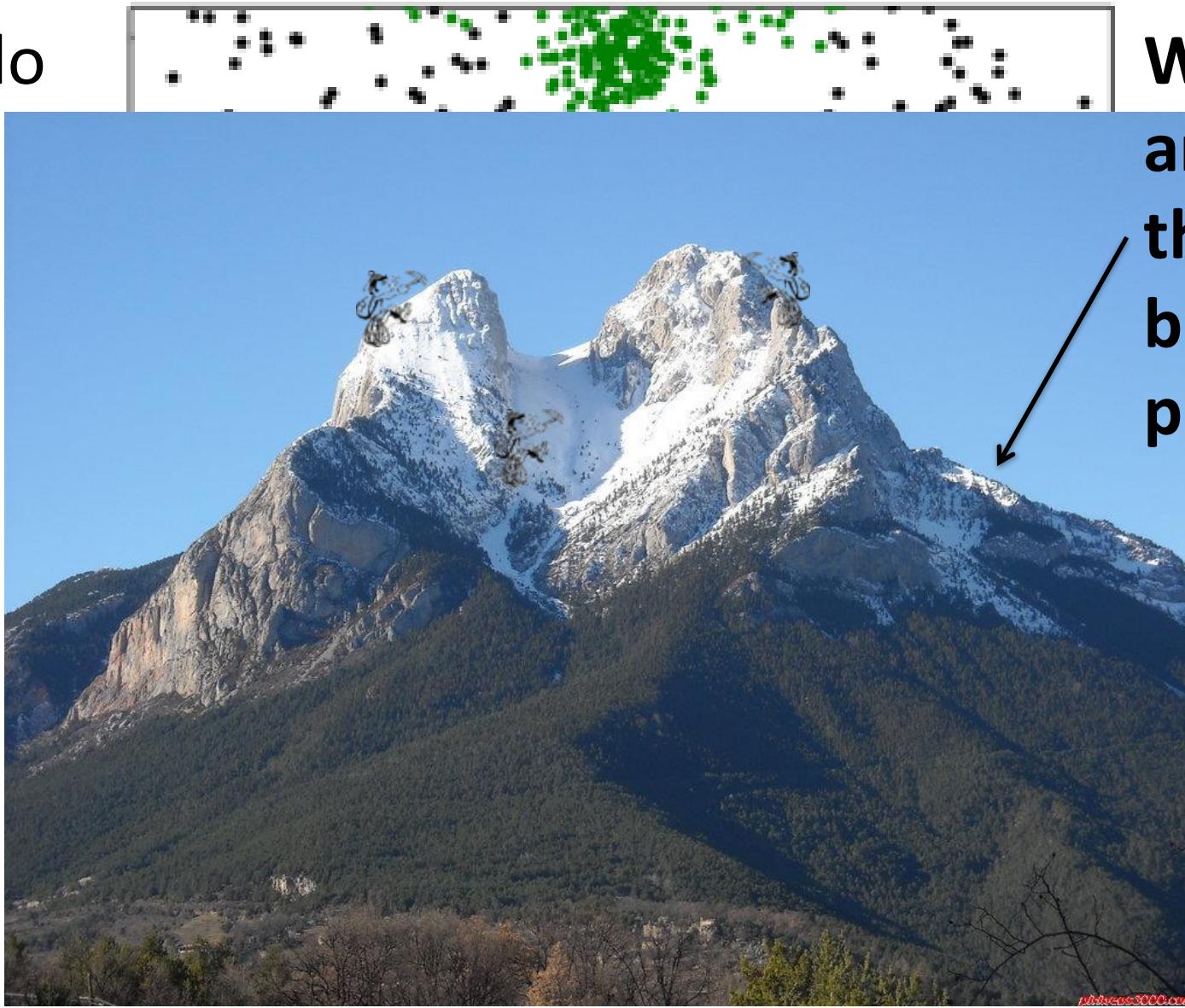


# Density Peaks clustering example



# Density Peaks: Halo

Halo

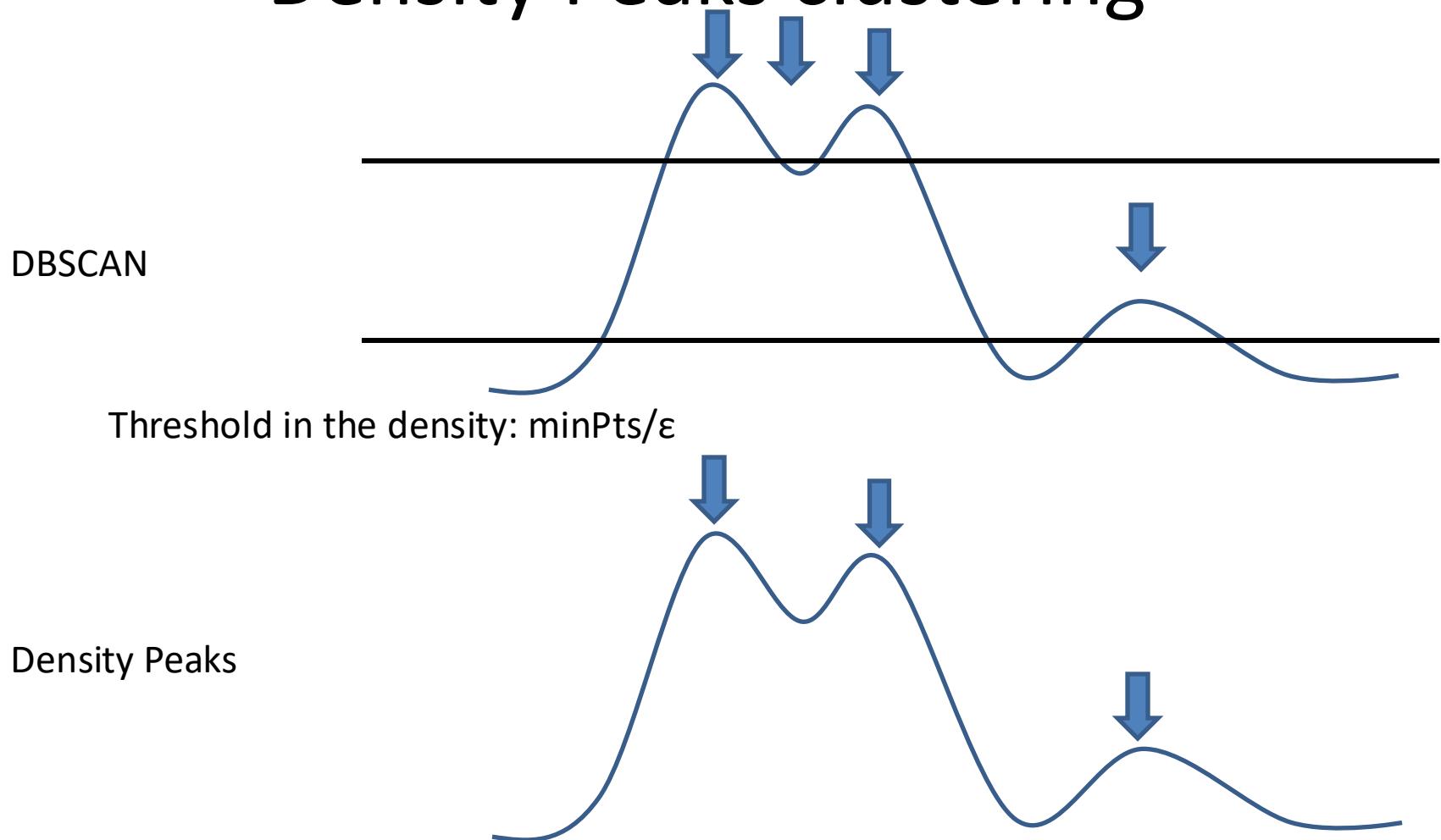


What  
are  
these  
black  
points?

# Border definition

- A point  $i$  of cluster A is border point if it has, within  $d_c$ , a point  $j$  belonging to another cluster.
- The border density of A is, then, the higher density among all the border points belonging to A.

# Differences between DBSCAN and Density Peaks clustering



Directly find the peaks (But depends on the decision graph)

# A selected list of clustering algorithms

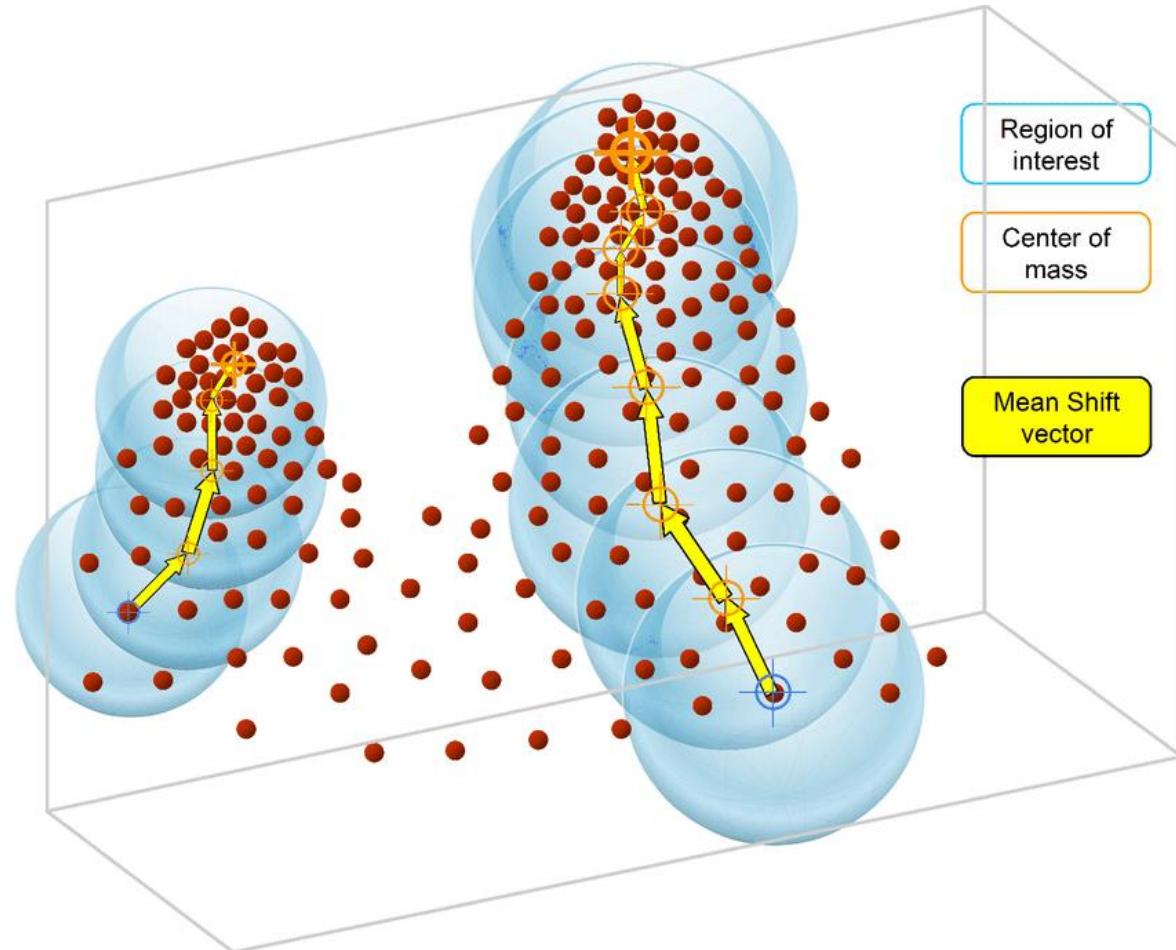
- K-means
- Fuzzy c-means
- Agglomerative hierarchical
- Gaussian Mixture
- Density Based Algorithms:
  - DBSCAN
  - Density Peaks
  - **Mean-Shift clustering**
- **Affinity Propagation**
- **Spectral Clustering**

# Other important clustering algorithms: **Mean-Shift**

- The idea is similar to Density-Peaks: Find the density maxima and assign the data points to its correspondent maxima.
- The density is estimated with kernel density estimation.
- For each point, shifts its coordinates towards the weighted mean of the density in a region of interest until it converges.

$$m(x) = \frac{\sum_{x_i \in N} x_i K(x_i - x)}{\sum_{x_i \in N} K(x_i - x)}$$

# Other important clustering algorithms: Mean-Shift



# Other important clustering algorithms:

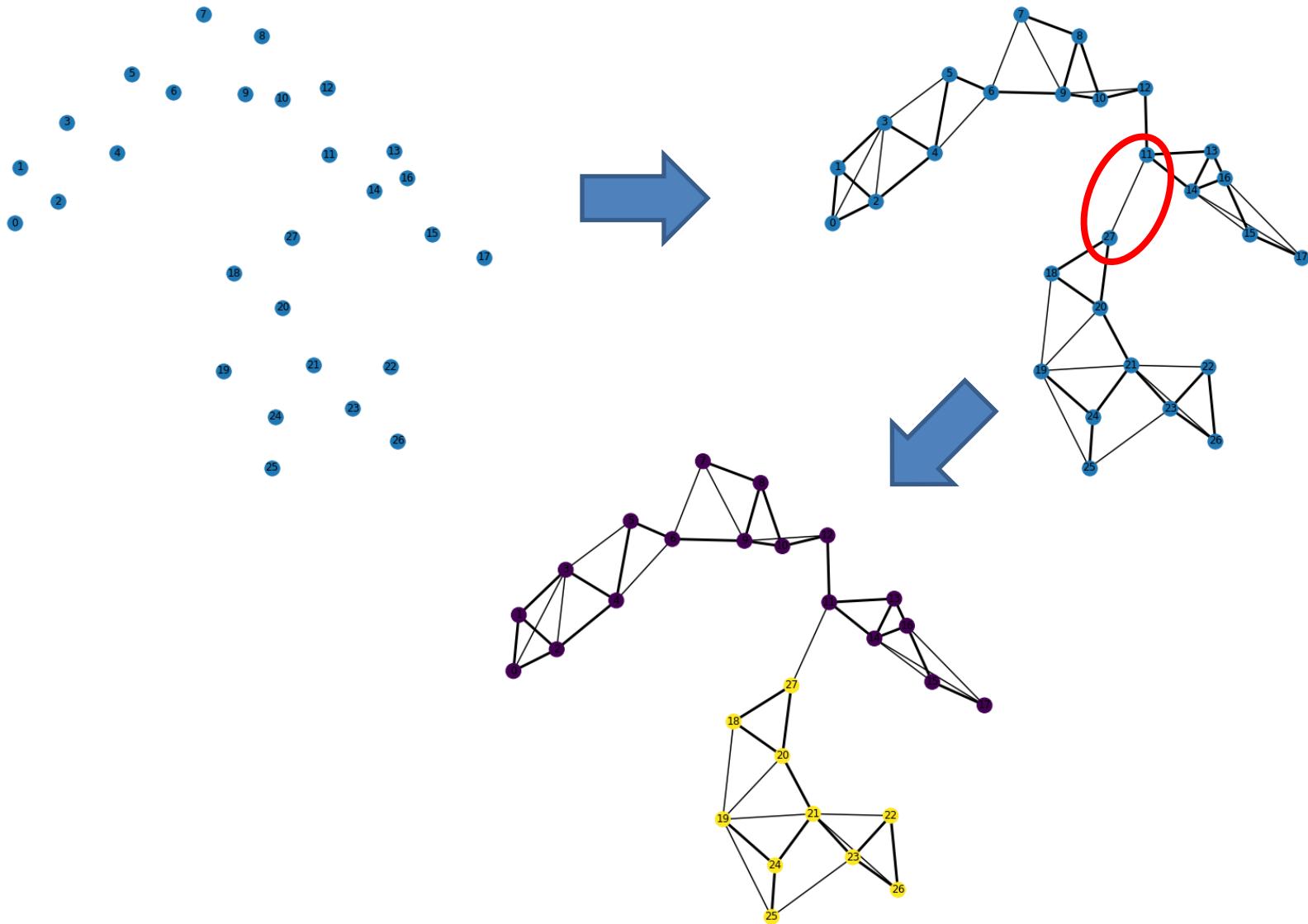
## Affinity Propagation. Concepts

- AffinityPropagation creates clusters by sending messages between pairs of samples until convergence.
- Two categories of messages:
  - **Responsibility**  $r(i, k)$ , which is the accumulated evidence that sample  $k$  should be the exemplar for sample  $i$ .  
$$r(i, k) \leftarrow S_{ik} - \max_{k' \neq k} [S_{ik'} + a(i, k')]$$
  - **Availability**  $a(i, k)$  which is the accumulated evidence that sample  $i$  should choose sample  $k$  to be its exemplar, and considers the values for all other samples  
$$a(i, k) \leftarrow \min[0, r(k, k) + \sum_{i' \neq \{k, i\}} r(i', k)]$$

# Other important clustering algorithms: **Affinity Propagation**. The algorithm

- $S_{ik}$  is the similarity, usually is taken  $S_{ik} = -\|x_i - x_k\|^2$
- $S_{kk}$ , known as preference, plays an important role. A value close to the minimum possible similarity produces fewer classes, while a value close to or larger than the maximum possible similarity produces many classes. It is typically initialized to the median similarity of all pairs of inputs.
- The algorithm starts by setting all the values of  $r$  and  $a$  to zero. And iterates by first updating responsibilities and then availabilities until a number of iterations is reached.
- The exemplars are extracted from the final matrices as those whose 'responsibility + availability' for themselves is positive.

# Other important clustering algorithms: **Spectral clustering**. From data points to similarity graphs



Other important clustering algorithms: **Spectral clustering**. Obtaining the (similarity) graph

- **$\varepsilon$ -ball graph:** Join vertices within a radius  $\varepsilon$ .
- **$k$ -NN graph:** Join the  $k$ -Nearest Neighbor vertices.
- **Fully connected graphs:** Join all the vertices,

$$\text{for instance: } S_{ij} = e^{-\frac{d_{ij}^2}{2\sigma^2}}$$

# Other important clustering algorithms: **Spectral clustering.** Concepts

- We need to find the cuts that minimize the weight of the cut edges but taking into account the size of the remaining graphs (otherwise it cuts only graph leaves).
- Several flavors of spectral clustering depend on the way that the graph is build and how we consider the size of the graph.
- An important concept is the Laplacian of the graph, defined as:  $\mathbb{L} = \mathbb{D} - \mathbb{S}$ . Where  $\mathbb{S}$  is the similarity matrix and  $\mathbb{D}$  is the *degree* matrix. A diagonal matrix with  $D_{ii} = \sum_j S_{ij}$

# Other important clustering algorithms: **Spectral clustering.** Algorithm

1. Construct the similarity graph
2. Compute the Laplacian
3. Compute the first  $k$  eigenvectors
4. Use the coordinates as input for the  $k$ -means algorithm and obtain the clusters

# A selected list of clustering algorithms

- **K-means\***
- **Fuzzy c-means \***
- **Agglomerative hierarchical \***
- **Gaussian Mixture \***
- **Density Based Algorithms:**
  - DBSCAN \*
  - Density Peaks \*
  - Mean-Shift clustering
- Affinity Propagation
- Spectral Clustering

# Some ideas about Cluster Validation

# Cluster validation

Supervised classification:

- Class labels known for ground truth
- Accuracy, precision, recall

Cluster analysis

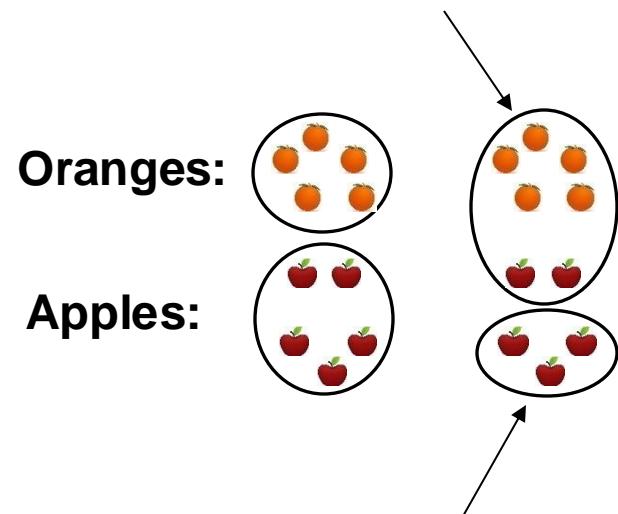
- No class labels

Validation need to:

- Compare clustering algorithms
- Solve number of clusters
- Avoid finding patterns in noise

$$\text{Precision} = 5/5 = 100\%$$

$$\text{Recall} = 5/7 = 71\%$$



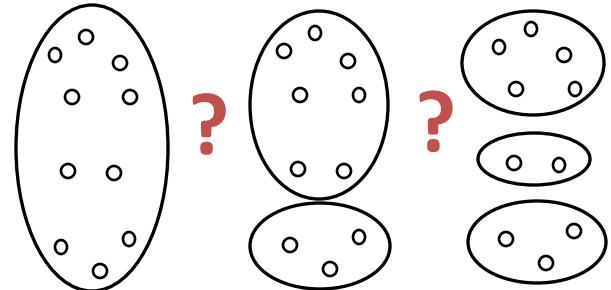
$$\text{Precision} = 3/5 = 60\%$$

$$\text{Recall} = 3/3 = 100\%$$

# Measuring clustering validity

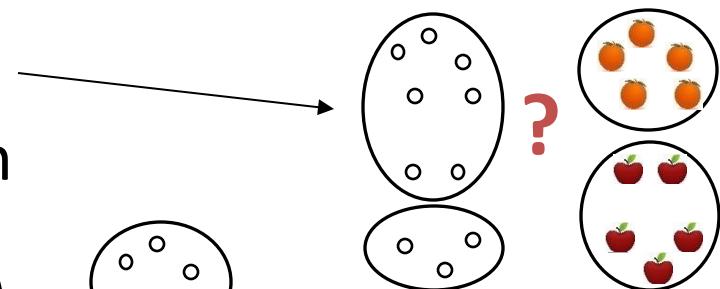
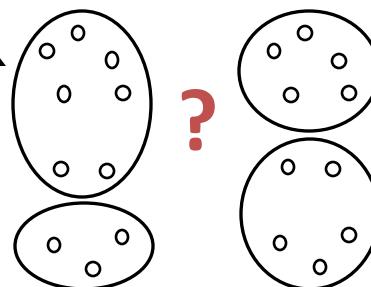
## Internal Index:

- Validate *without* external info
- With different number of clusters
- Solve the number of clusters

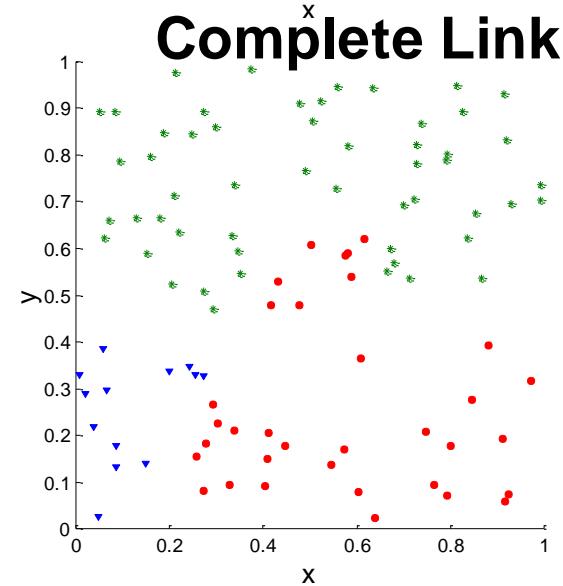
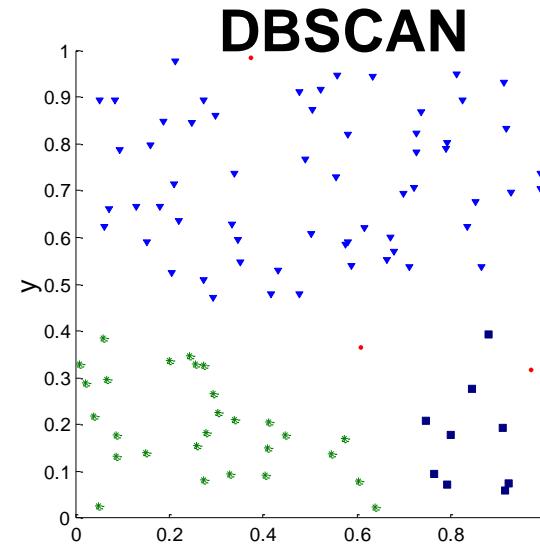
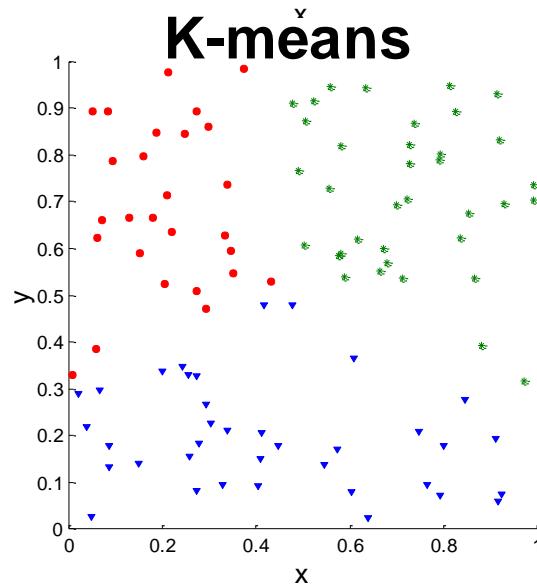
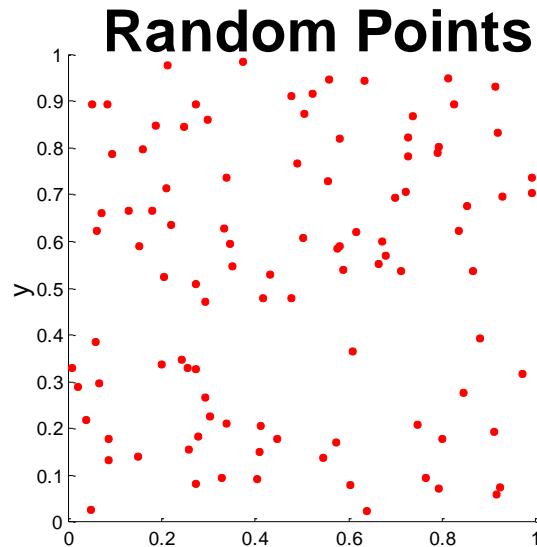


## External Index

- Validate against ground truth
- Compare two clusters:  
(how similar)



# Clustering of random data

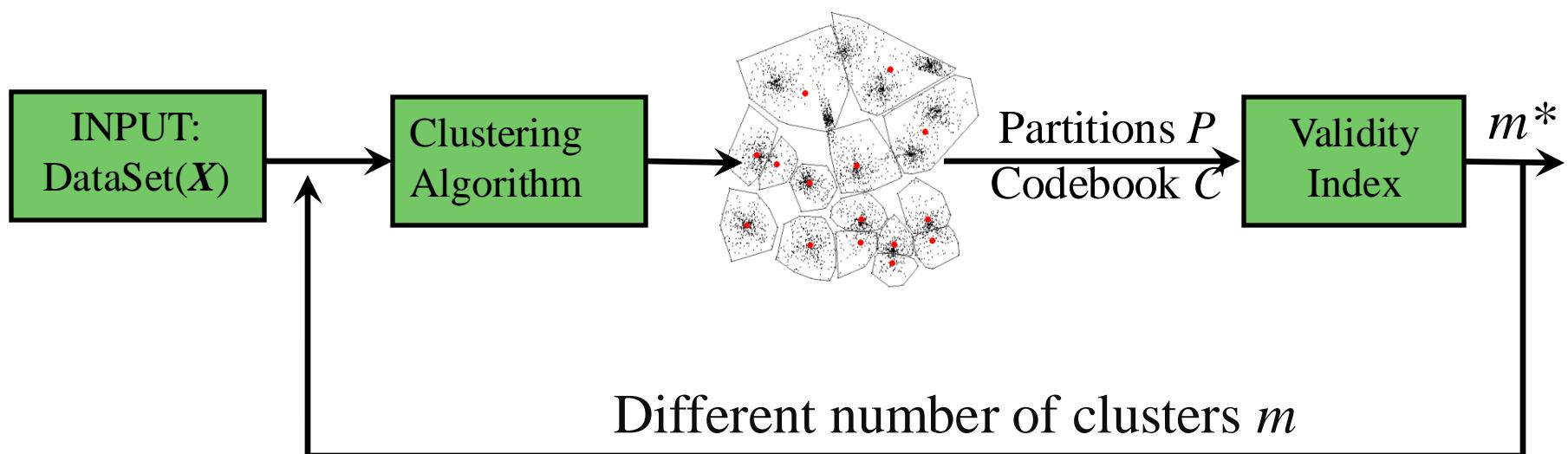


# Cluster validation process

1. Distinguishing whether non-random structure actually exists in the data (one cluster).
2. Comparing the results of a cluster analysis to externally known results, e.g., to externally given class labels.
3. Evaluating how well the results of a cluster analysis fit the data *without* reference to external information.
4. Comparing the results of two different sets of cluster analyses to determine which is better.
5. Determining the number of clusters.

# Cluster validation process

- **Cluster validation** refers to procedures that evaluate the results of clustering in a **quantitative** and **objective** fashion. [Jain & Dubes, 1988]
  - How to be “quantitative”: To employ the measures.
  - How to be “objective”: To validate the measures!

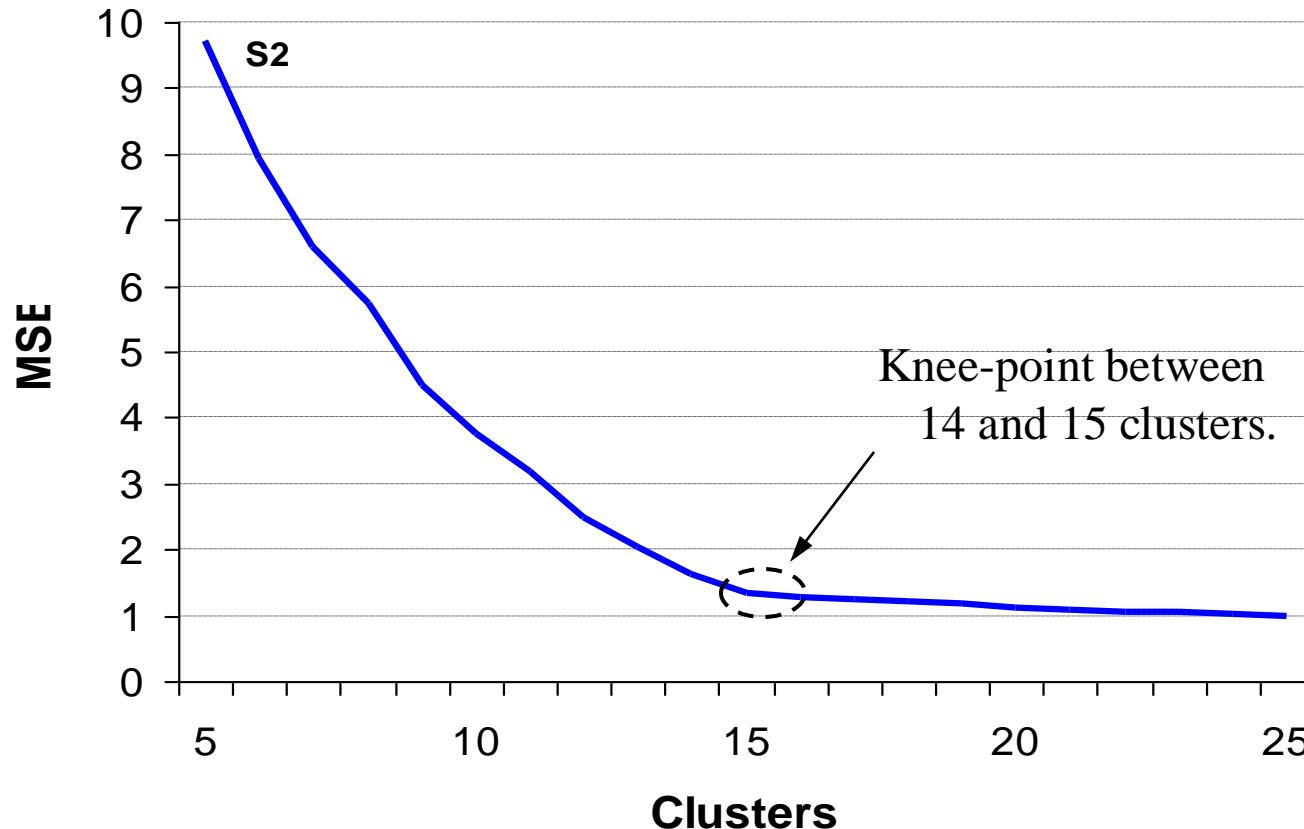


# Internal indexes

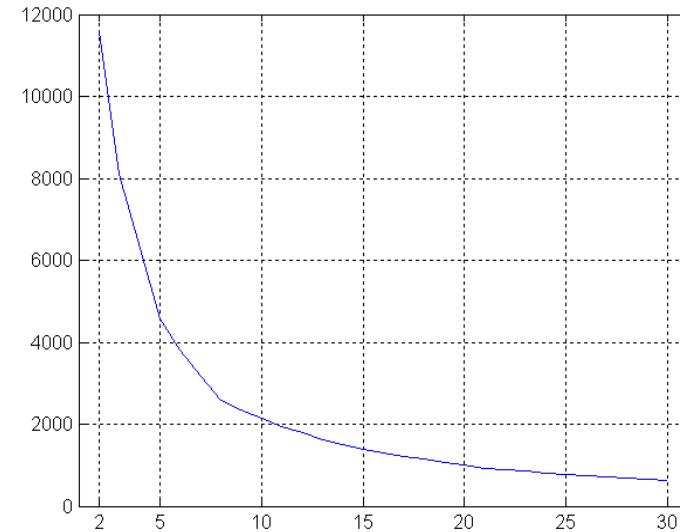
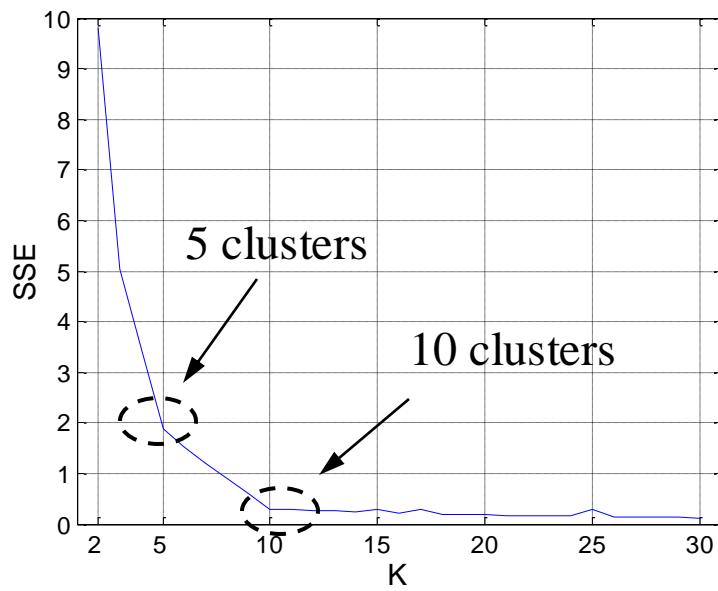
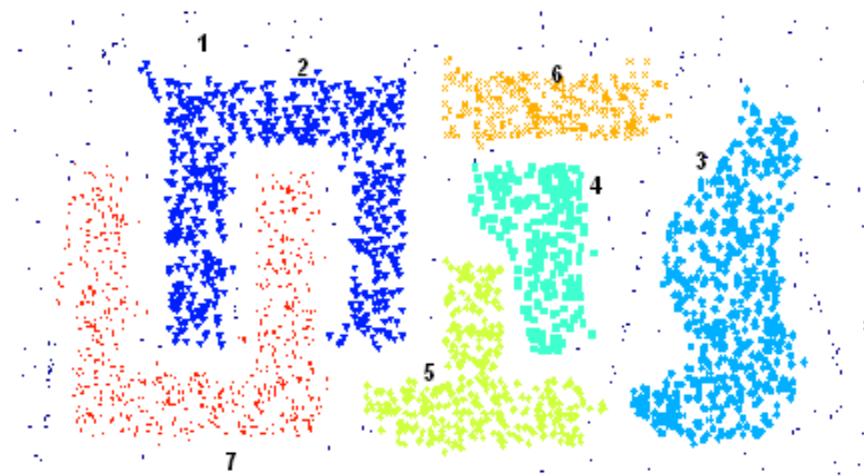
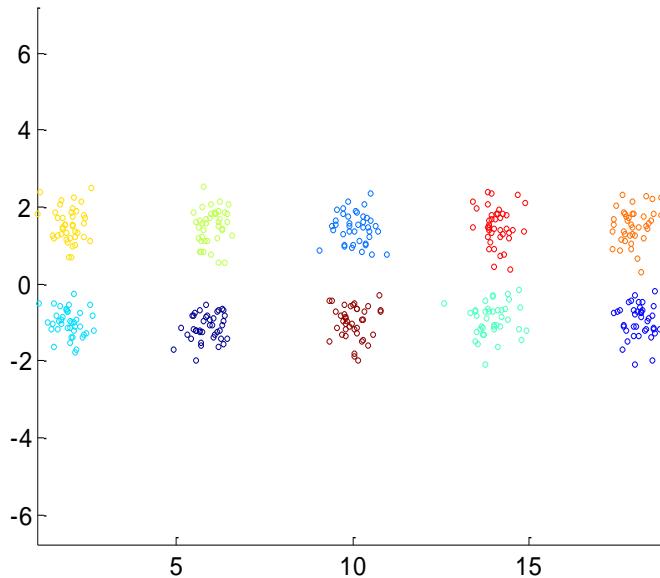
- Ground truth is rarely available but unsupervised validation must be done.
- Minimizes (or maximizes) internal index:
  - Variances of within cluster and between clusters
  - Rate-distortion method
  - F-ratio
  - Davies-Bouldin index (DBI)
  - Bayesian Information Criterion (BIC)
  - Silhouette Coefficient
  - Minimum description principle (MDL)
  - Stochastic complexity (SC)

# Mean square error (MSE)

- The more clusters the smaller the MSE.
- Small knee-point near the correct value.
- But how to detect?



# Mean square error (MSE)

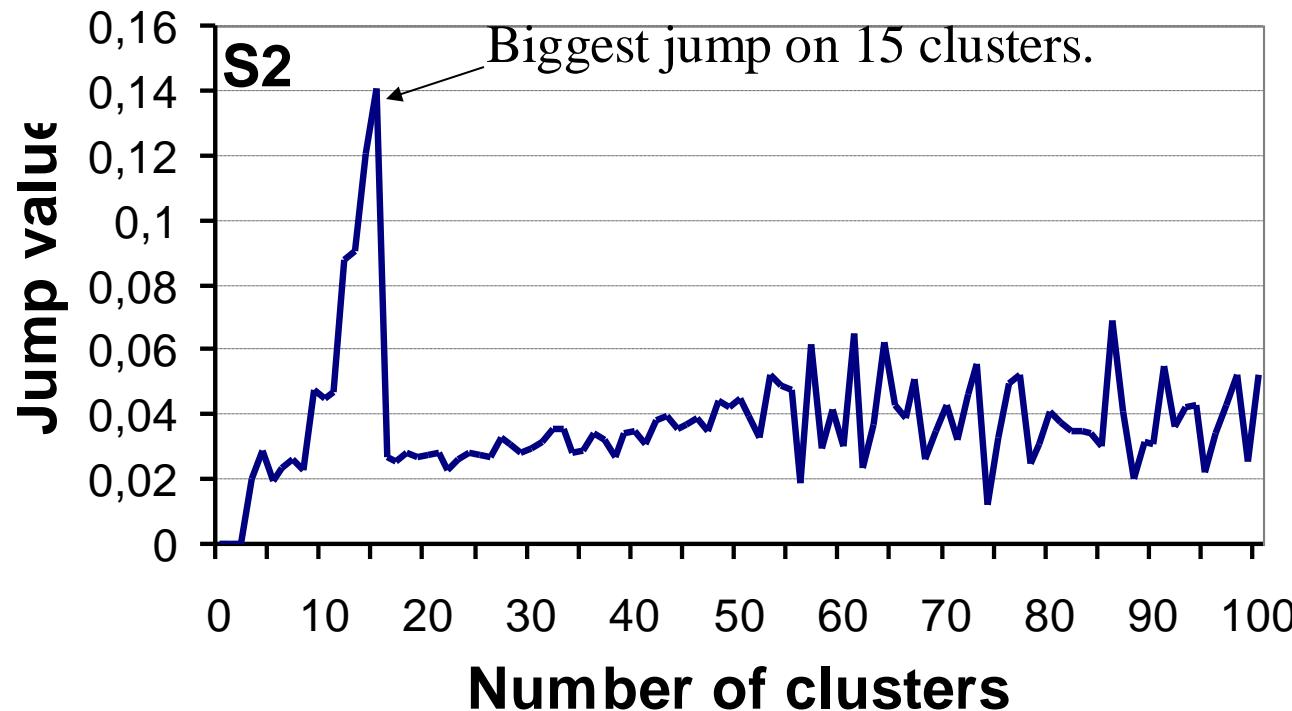


# Jump point of MSE

(rate-distortion approach)

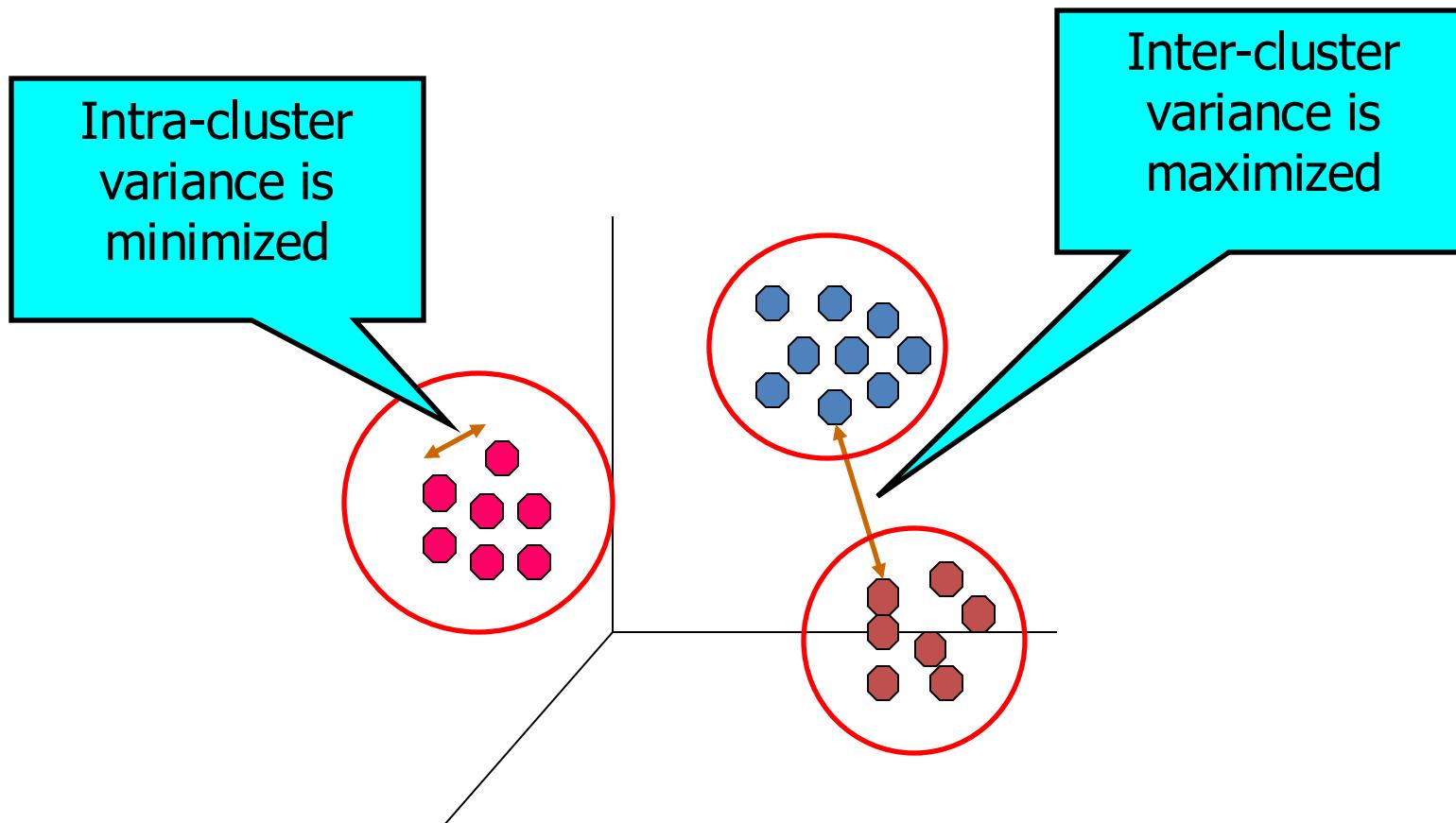
First derivative of powered MSE values:

$$J(k) = \text{MSE}(k)^{-d/2} - \text{MSE}(k-1)^{-d/2}$$



# From MSE to cluster validity

- Minimize within cluster variance (MSE)
- Maximize between cluster variance



# Variances

Within cluster:

$$SSW(C, k) = \sum_{i=1}^N \|x_i - c_{p(i)}\|^2$$

Between clusters:

$$SSB(C, k) = \sum_{j=1}^k n_j \|c_j - \bar{x}\|^2$$

Total Variance of data set:

$$\sigma(X) = \sum_{i=1}^N \|x_i - c_{p(i)}\|^2 + \sum_{j=1}^k n_j \|c_j - \bar{x}\|^2$$

SSW      SSB

# F-ratio variance test

- Variance-ratio F-test
- Measures ratio of between-groups variance against the within-groups variance (original f-test)
- F-ratio (WB-index):

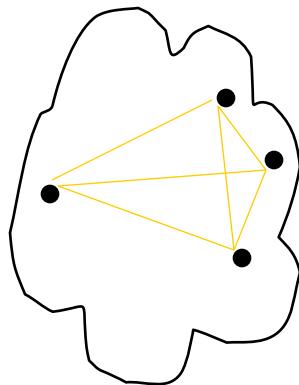
$$F = \frac{k \cdot \sum_{i=1}^N \|x_i - c_{p(i)}\|^2}{\sum_{j=1}^k n_j \|c_j - \bar{x}\|^2} = \frac{k \cdot SSW}{\sigma(X) - SSW}$$

SSB

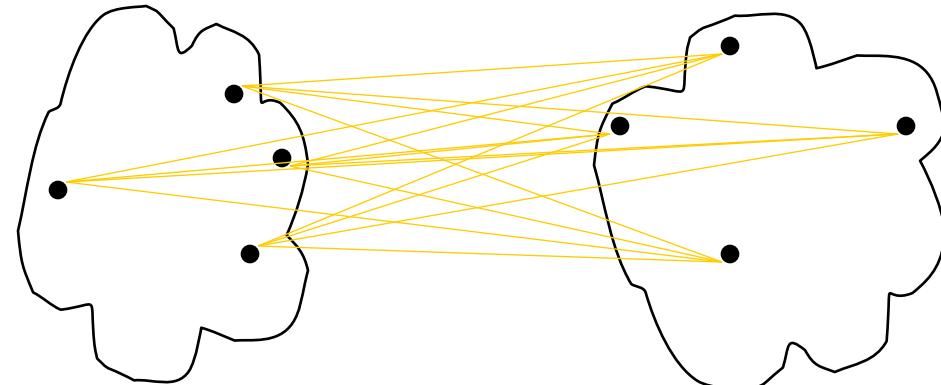
# Silhouette coefficient

[Kaufman&Rousseeuw, 1990]

- Cohesion: measures how closely related are objects in a cluster
- Separation: measure how distinct or well-separated a cluster is from other clusters



cohesion



separation

# Silhouette coefficient

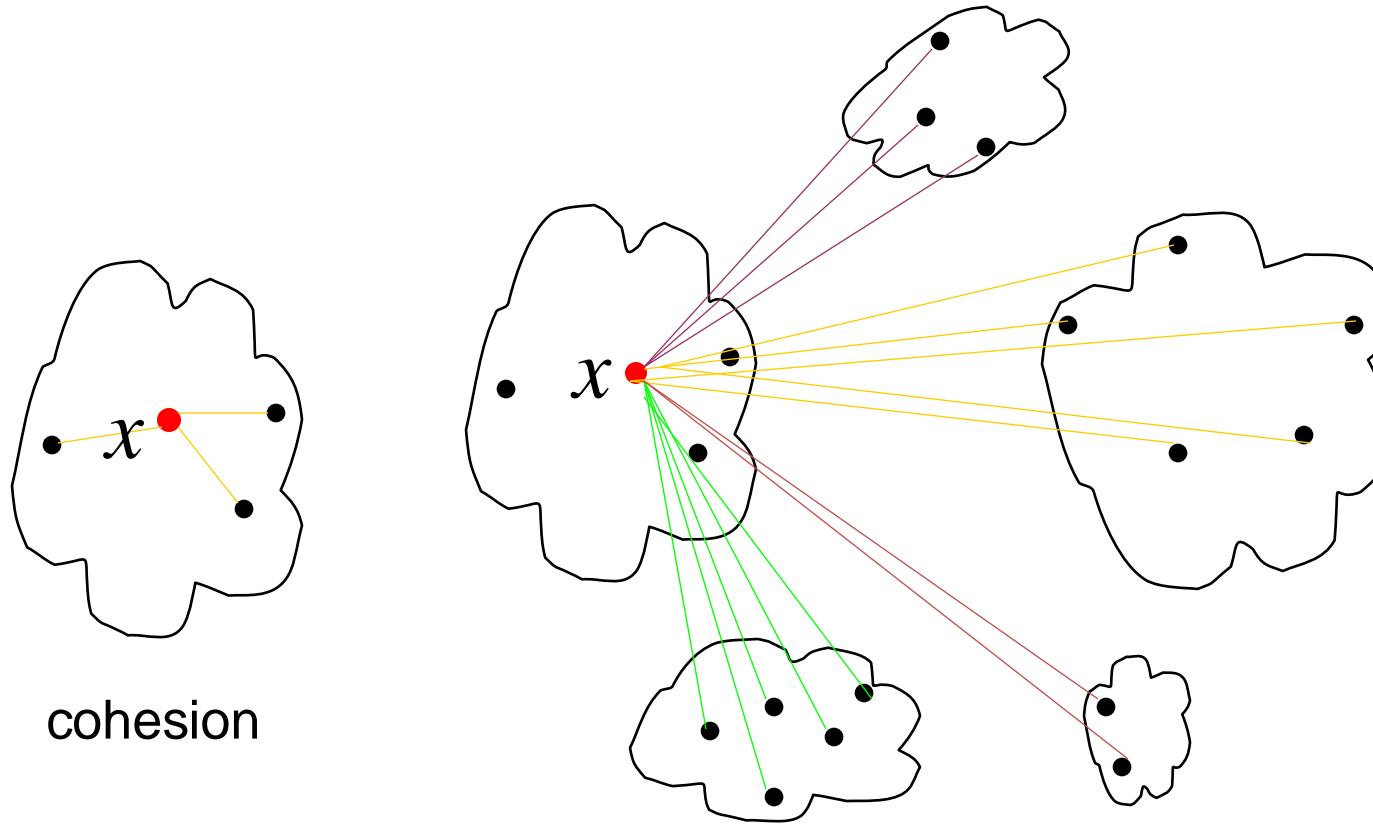
- *Cohesion*  $a(x)$ : average distance of  $x$  to all other vectors in the same cluster.
- *Separation*  $b(x)$ : average distance of  $x$  to the vectors in other clusters. Find the minimum among the clusters.
- *silhouette*  $s(x)$ :

$$s(x) = \frac{b(x) - a(x)}{\max\{a(x), b(x)\}}$$

- $s(x) = [-1, +1]$ : -1=bad, 0=indifferent, 1=good
- Silhouette coefficient (SC):

$$SC = \frac{1}{N} \sum_{i=1}^N s(x)$$

# Silhouette coefficient



$a(x)$ : average distance  
in the cluster

$b(x)$ : average distances to  
others clusters, find minimal