Data Analysis - Lecture 4

Introduction to unsupervised learning: clustering

Dr. Jérémie Sublime

LISITE Laboratory - DaSSIP Team - ISEP
LIPN - CNRS UMR 7030

jeremie.sublime@isep.fr

Plan

Outline

What is unsupervised learning ?

Unsupervised learning *is a Machine Learning task the aim of which is to find hidden structures and patterns in* unlabeled data.
There are several tasks linked to unsupervised learning:

- Data partitioning (or clustering)
- Anomalies detection
- Unsupervised neural-networks
- Latent variables model learning

What is unsupervised learning ?

Unsupervised learning *is a Machine Learning task the aim of which is to find hidden structures and patterns in* unlabeled data.
There are several tasks linked to unsupervised learning:

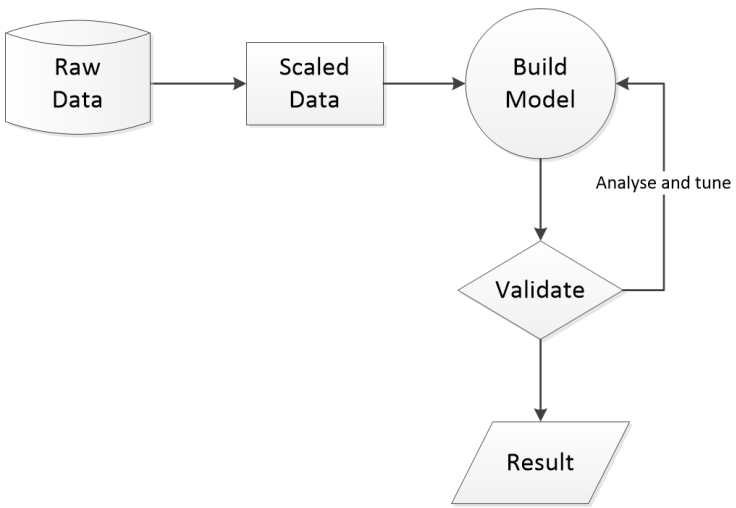
- Data partitioning (or clustering)
- Anomalies detection
- Unsupervised neural-networks
- Latent variables model learning

Remark

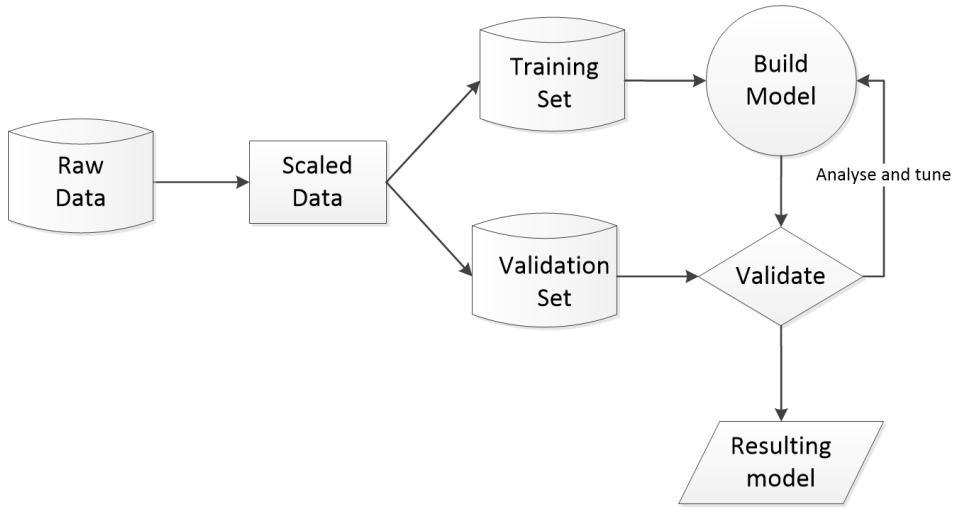
Unlike statistical analysis methods that we have seen so far and whose focus was the relations between the features of a data set, unsupervised learning focuses on links and similarities between the data themselves.

What is unsupervised learning ?

Unsupervised learning is said to be "*unsupervised*" because it finds structures and build a model from completely unlabeled data. It therefore differs from **supervised learning** (Cf. Lecture 5) which builds a model given already labeled data.



(a) Unsupervised learning flow



(b) Supervised learning flow

Data partitioning : Definition

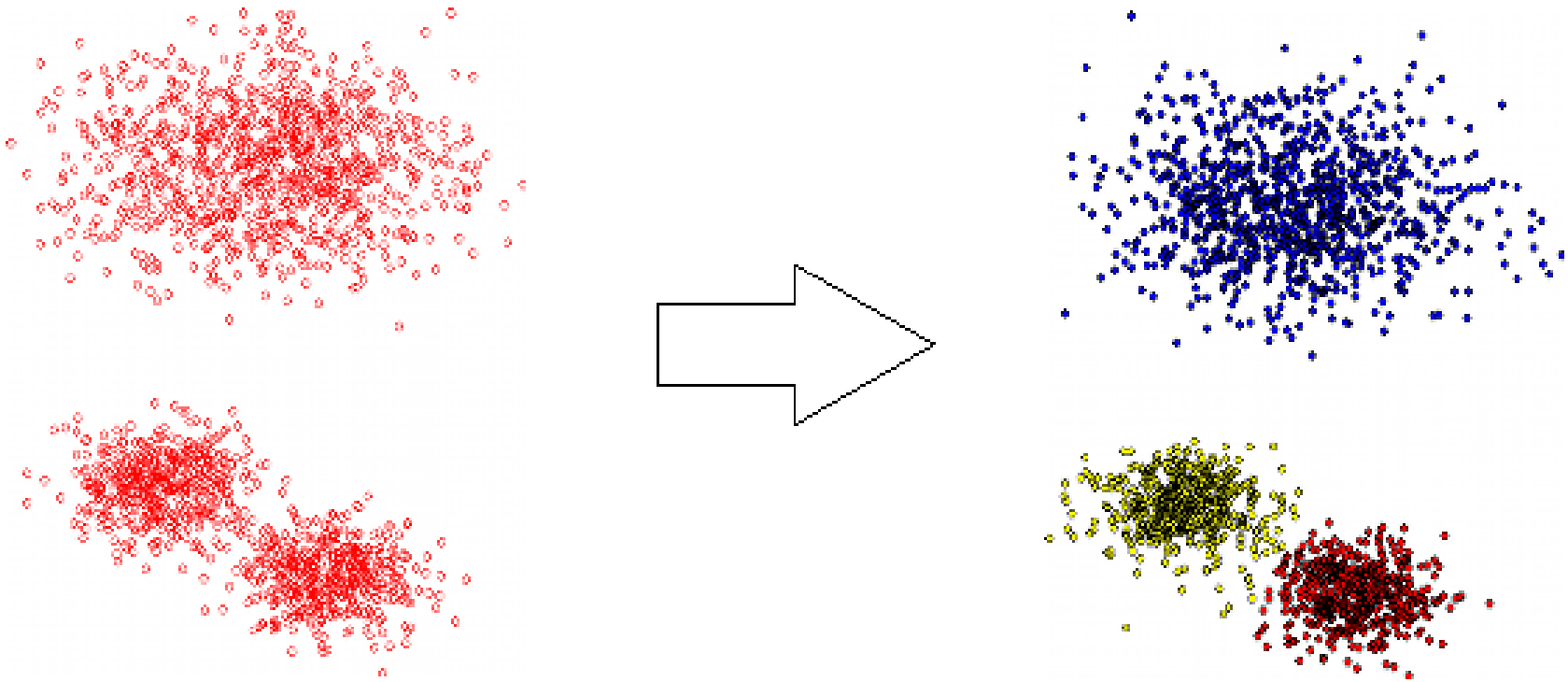
Data partitioning *-or* clustering- *is a Machine Learning task of exploratory data mining the aim of which is to split a data set made of several data (also called objects, data objects, or observations) into several subsets containing homogeneous and similar data. The groups created via this process are called* clusters*.*

Applications of data partitioning

Pattern recognition, web mining, business intelligence, biology, security applications, customer analysis, etc.

Data partitioning : Examples

We want to automatically discover groups of similar data:



Data partitioning : Examples

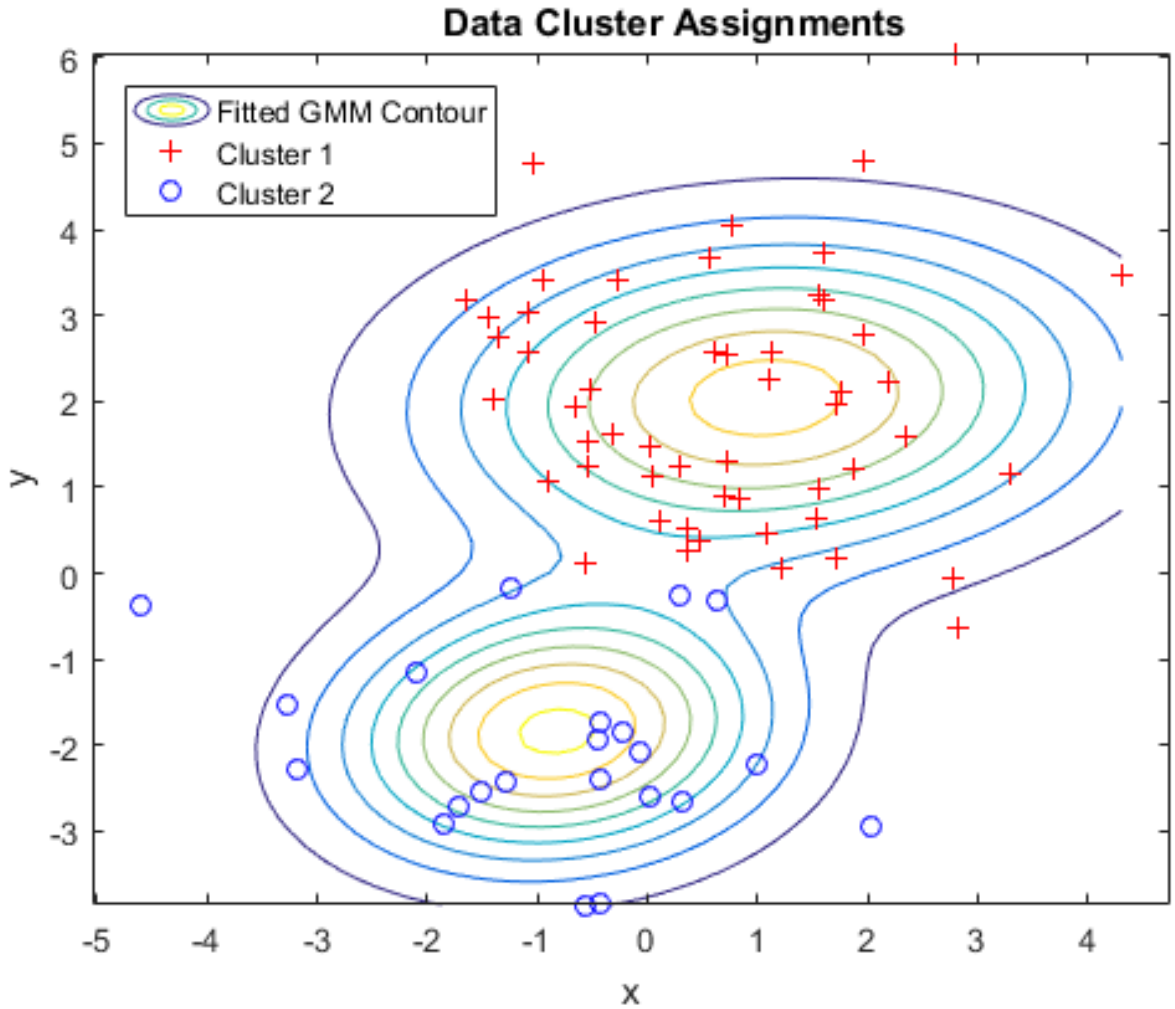


Figure: Example of 2 clusters with Gaussian distributions

Data partitioning : Examples

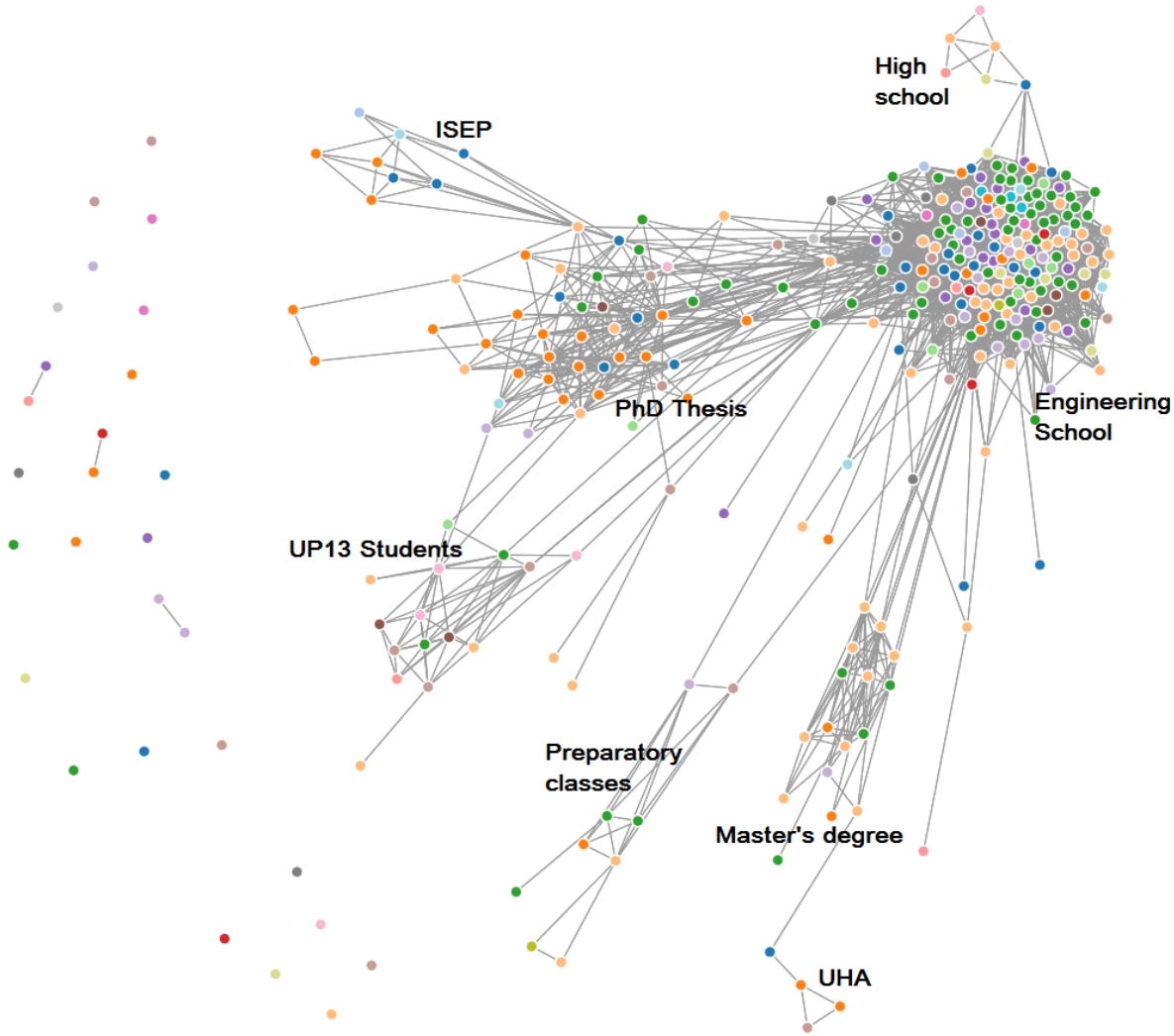


Figure: Community detection in social network graphs

Outline

Cluster : Definition

Definition

A cluster is a group of relatively homogeneous data that share more common characteristics between each other than with the data belonging to other clusters.

The notion of cluster therefore relies on the notion of *similarity* and *dissimilarity* between the data.

Clusters and similarity

What is similarity ?

- The quality or state of being similar, the likeness, the resemblance, the a similarity of features.
- How to define the notion of similarity between complex objects described by several attributes ?

Similarity is hard to define, but ...

Clusters and similarity

What is similarity ?

- The quality or state of being similar, the likeness, the resemblance, the a similarity of features.
- How to define the notion of similarity between complex objects described by several attributes ?

Similarity is hard to define, but ...

You know it when you see it !

Clusters, similarity and distance

- The similarity or dissimilarity between two observations is most often computed using a distance function.
- Depending on the data and the application, some distances may prove more appropriate than others (e.g. Hamming distance for text data).

Euclidian distance	$\|\|a-b\|\|_2 = \sqrt{\sum_i (a_i - b_i)^2}$
Squared Euclidian distance	$\|\|a-b\|\|_2^2 = \sum_i (a_i - b_i)^2$
Manhattan distance	$\|\|a-b\|\|_1 = \sum_i \|a_i - b_i\|$
Maximum distance	$\|\|a-b\|\|_\infty = \max_i \|a_i - b_i\|$
Mahalanobis distance	$\sqrt{(a-b)^\top S^{-1}(a-b)}$ where S is the covariance matrix
Hamming distance	$Hamming(a,b) = \sum_i (1 - \delta_{a_i,b_i})$

Table: Examples of common distances

- Creating custom distance functions is sometimes required.

Clustering partitions

Let us denote $X = \{x_1, \cdots, x_N\}, x_n \in \mathbb{R}^d$ a data data set containing N observations described by d real features, and $C = \{c_1, ..., c_K\}$ the set of possible clusters.

Clustering partitions

The result of a clustering algorithm is called a clustering partition*, or just partition.* Depending on whether or not a data can belong to several clusters, the partition can be **hard**, **soft**, or **fuzzy**.

(a) Hard clustering

	c_1	c_2	c_3
x_1	1	0	0
x_2	0	1	0
x_3	0	0	1
x_4	0	0	1

(b) Soft clustering

	c_1	c_2	c_3
x_1	1	1	0
x_2	0	1	1
x_3	0	0	1
x_4	0	0	1

(c) Fuzzy clustering

	c_1	c_2	c_3
x_1	0.9	0.1	0
x_2	0	0.8	0.2
x_3	0	0.3	0.7
x_4	0	0	1.0

Table: Example of several objects' degree of belonging to 3 clusters

Clustering partitions

(a) Hard clustering

	c_1	c_2	c_3
x_1	1	0	0
x_2	0	1	0
x_3	0	0	1
x_4	0	0	1

(b) Soft clustering

	c_1	c_2	c_3
x_1	1	1	0
x_2	0	1	1
x_3	0	0	1
x_4	0	0	1

(c) Fuzzy clustering

	c_1	c_2	c_3
x_1	0.9	0.1	0
x_2	0	0.8	0.2
x_3	0	0.3	0.7
x_4	0	0	1.0

Let us note S the clustering partition:

- In hard clustering, S is a vector $S = \{s_1, \cdots, s_N\}, s_n \in [1..K]$
- In soft clustering, S is a matrix $S = \{s_{n,k}\}_{(N \times K)}, s_{n,k} \in (0, 1)$
- In fuzzy clustering, S is a matrix too:
  $S = \{s_{n,k}\}_{(N \times K)}, s_{n,k} \in [0, 1] \quad \forall n, \sum_k s_{n,k} = 1$

Types of clustering algorithms

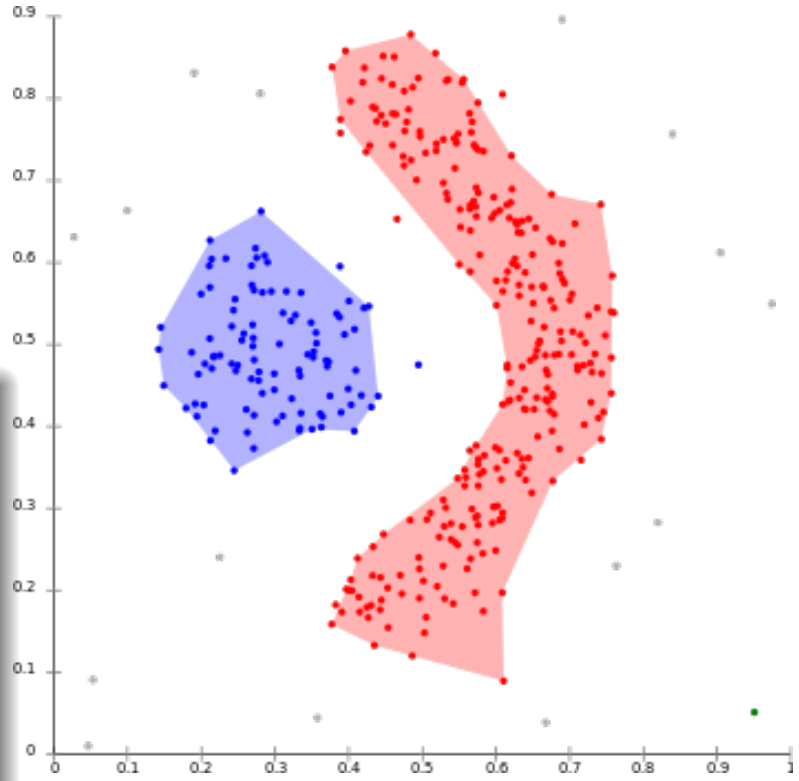
Now that we know what we want to do, the question is: Given a data set, a distance function and a desired type of partition, how do we find the clusters ?

There are several "families" of clustering algorithms that favor different methods to extract the clusters from the data:

- Density-based clustering algorithms
- Hierarchical clustering algorithms
- Prototype-based clustering algorithms
- Distribution-bsed clustering methods
- Spectral clustering methods

Density-based clustering algorithms

- Inspired from the human vision: the clusters are defined as high density spaces and are separated by low density areas.
- Examples of algorithms: DBSCAN, OPTICS

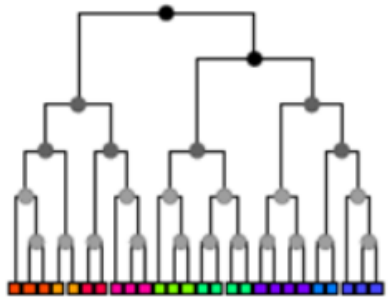


Properties

- Pros: Does not presume the shape of the clusters or their number, detect outliers.
- Cons: High complexity $O(N^2)$, difficult to parametrize

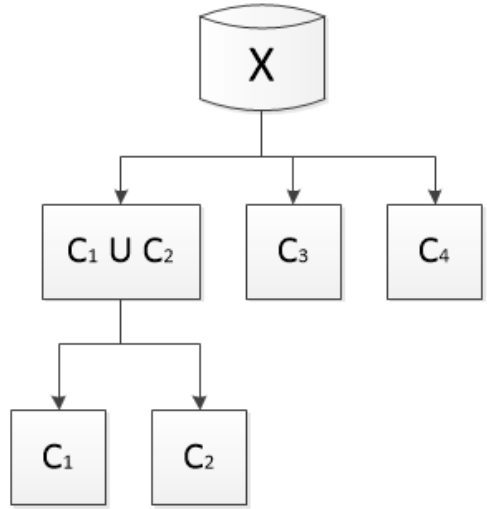
Hierarchical clustering algorithms

- Inspired from phylogenetic trees in biology.
- Approaches: agglomerative, divisive
- Examples of algorithms: HCA, CURE, CLINK



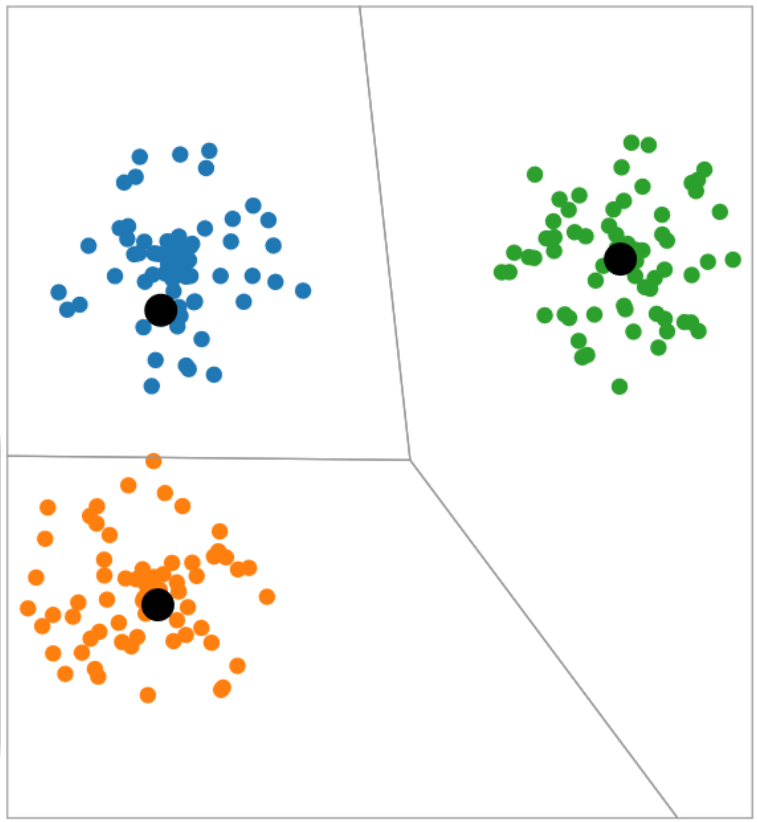
Properties

- Pros: Highlight hierarchical cluster structures, comprehensive visualization of the clusters
- Cons: High complexity $O(N^2 \log N) to O(N^3)$, choosing where to cut the dendrogram can be cumbersome.



Prototype-based clustering algorithms

- Uses vector quantization to sum up the data using representatives (e.g. mean values)
- Examples of algorithms: K-Means, Fuzzy C-Means



Properties

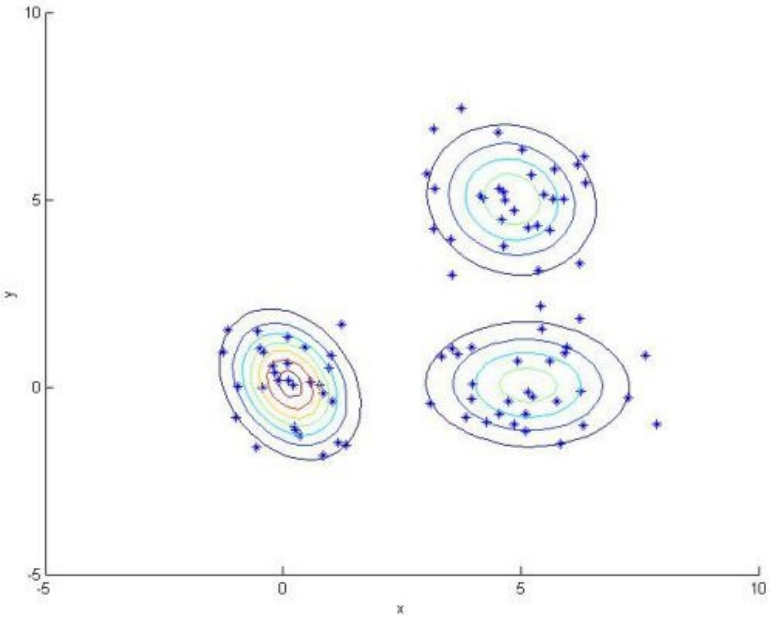
- Pros: Lower complexity $O(N)$
- Cons: Requires to give the number of clusters and to presume of their shapes.

Distribution-based clustering algorithms

- Uses strong mathematical models to describe the clusters (e.g. mixture of gaussian distributions)
- Examples of algorithms: EM

Properties

- Pros: Lower complexity $O(N)$, strong models, convergence proofs
- Cons: Requires to give the number of clusters and to assume that the clusters will follow a given distribution.



Spectral clustering algorithms

- Sees clustering as a graph partitioning problem using the similarity matrix between the different objects.
- Examples of algorithms: Shi-Malik algorithm (normalized cuts), Meila-Shi algorithms.
- Mainly use for image analysis

Properties

- Pros: Good results, does not assume any shape for the clusters, fast with sparse data.
- Cons: High complexity $O(N^3)$, does not scale well, not very intuitive.

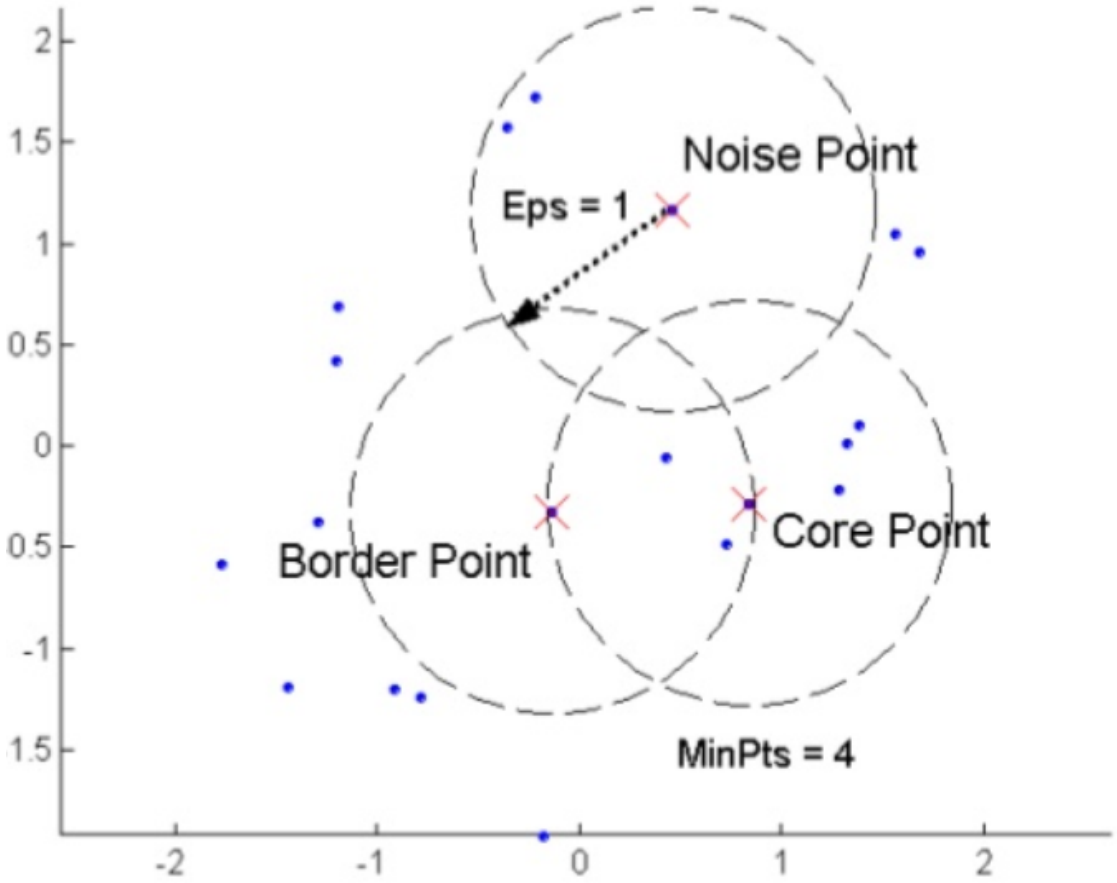
Outline

DBSCAN: Introduction

DBSCAN is a density based algorithm:

- The "Density" is based on the number of data within the radius "**Eps**" of a given element.
- A point is a **core point** if it has more data in its neighboring radius than a specified number of points "**MinPts**".
  - Core points are the main structures for the clusters.
- A **border point** has fewer points in its radius **Eps** than the number **MinPts** but it is within range of a core point.
- A **noise point** has fewer points in its radius **Eps** than the number **MinPts** and is not within range of a core point.

DBSCAN: Introduction



DBSCAN: Algorithm

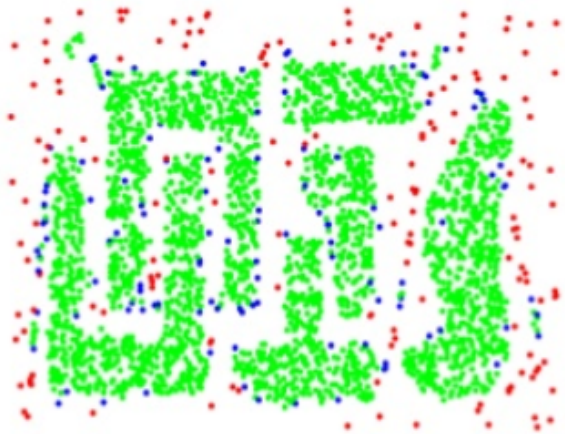
Function *DBSCAN*(X,ϵ,*MinPts*)
- C=0
- **forall** $x_n \in X$ **do**
  - **if** x_n has not been visited mark x_n as *visited*
  - V_n= regionQuery(x_n,ϵ)
  - **if** *sizeof*(V_n)$\geq$*MinPts* **then**
    - expandCluster(x_n,V_n,C++,ϵ,MinPts)
  - **else**
    - mark x_n as noise
  - **end**
- **end**

Function *expandCluster*(x_n,V_n,C,ϵ,*MinPts*)
- Add x_n to cluster C
- **forall** $x_i \in V_n$ **do**
  - **if** x_i *has not been visited* **then**
    - Mark x_i as visited
    - V_i = regionQuery(x_i,ϵ)
    - **if** *sizeof*(V_i) $>$ *MinPts* **then**
      - $V_n += V_i$
    - **end**
  - **end**
  - **if** x_i does not belong to any cluster yet: Add x_i to cluster C
- **end**

Function *regionQuery*(x_n,ϵ)
- **forall** $x_i \in X$ **do**
  - **if** $i \neq n$ and $d(x_n, x_i) \leq \epsilon$: list.add($\{x_i\}$)
- **end**
- **return** list

DBSCAN: Example

Original Points



Point types: core, border and noise

Eps = 10, MinPts = 4

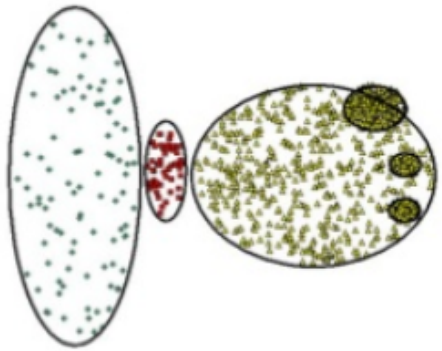
DBSCAN: Strengths

Original Points

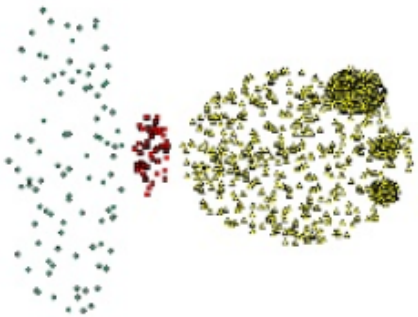
Clusters

- Resistant to noise
- Can handle clusters of different shapes and sizes

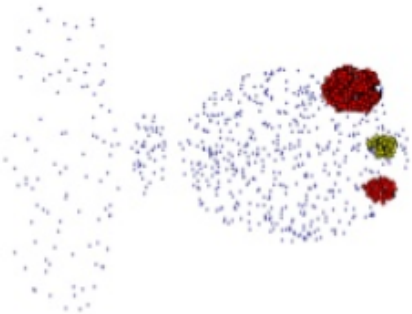
DBSCAN: Limits



Original Points



(MinPts=4, Eps=9.92)



(MinPts=4, Eps=9.75)

- Difficult to parametrize
- High Computational cost
- Does not handle well varying densities
  - OPTICS is a good evolution of DBSCAN that handles varying densities a lot better.

HCA: Introduction

Principle

Create a partition a each step by aggregating together the 2 closest clusters.

- Each data starts out as a single cluster.
- During the process a cluster can be a single data, or a group of data.
- The algorithm return a hierarchy of partitions in the form of a tree containing the history of the different aggregations.

HCA: Algorithm

1. Repeat until there is only one cluster left:
   - 1.a Compute the distance matrix between all existing clusters
   - 1.b Merge the two closest clusters
   - 1.c Update the dendrogram (hierarchical tree)
2. Cut the tree according to a criterion of your choice to get the final partition.

- This algorithm requires distance criterion between clusters: the **linkage criterion**.

HCA: Linkage criteria

Name	Formula	Comments
Single-linkage	$D_s(c_1, c_2) = \min_{x \in c_1, y \in c_2} d(x, y)$	Distance between the two closest elements
Complete-linkage	$D_c(c_1, c_2) = \max_{x \in c_1, y \in c_2} d(x, y)$	Distance between the two farthest element
Average-linkage	$D_a(c_1, c_2) = \frac{1}{\lvert c_1 \rvert \lvert c_2 \rvert} \sum_{x \in c_1} \sum_{y \in c_2} d(x, y)$	Average pairwise distance
Centroid-linkage	$D_\mu(c_1, c_2) = \lvert\lvert \mu_1 - \mu_2 \rvert\rvert$	Distance between the centroids

HCA: Single-Linkage criterion

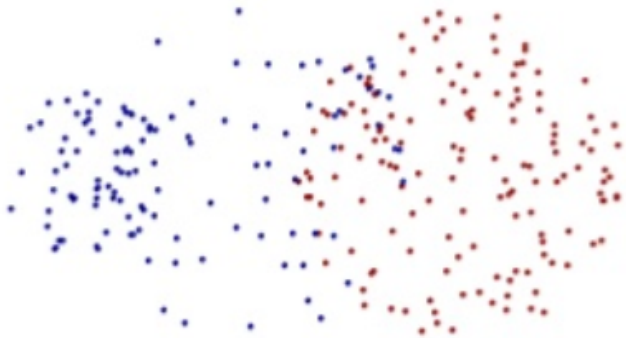
Original Points

Two Clusters

- Can handle clusters that are not necessarily spherical

HCA: Single-Linkage criterion

Original Points

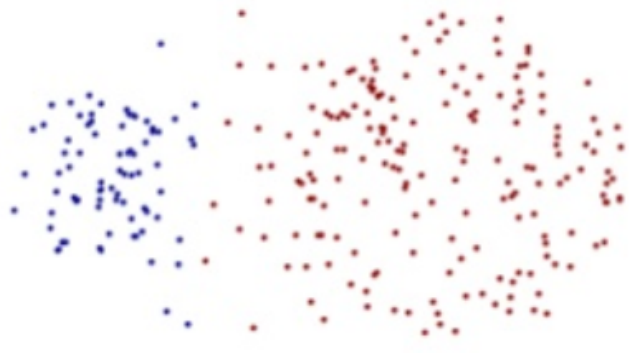


Two Clusters

- Sensitive to noise and outliers
- Cannot handle clusters that are in contact

HCA: Complete-Linkage criterion

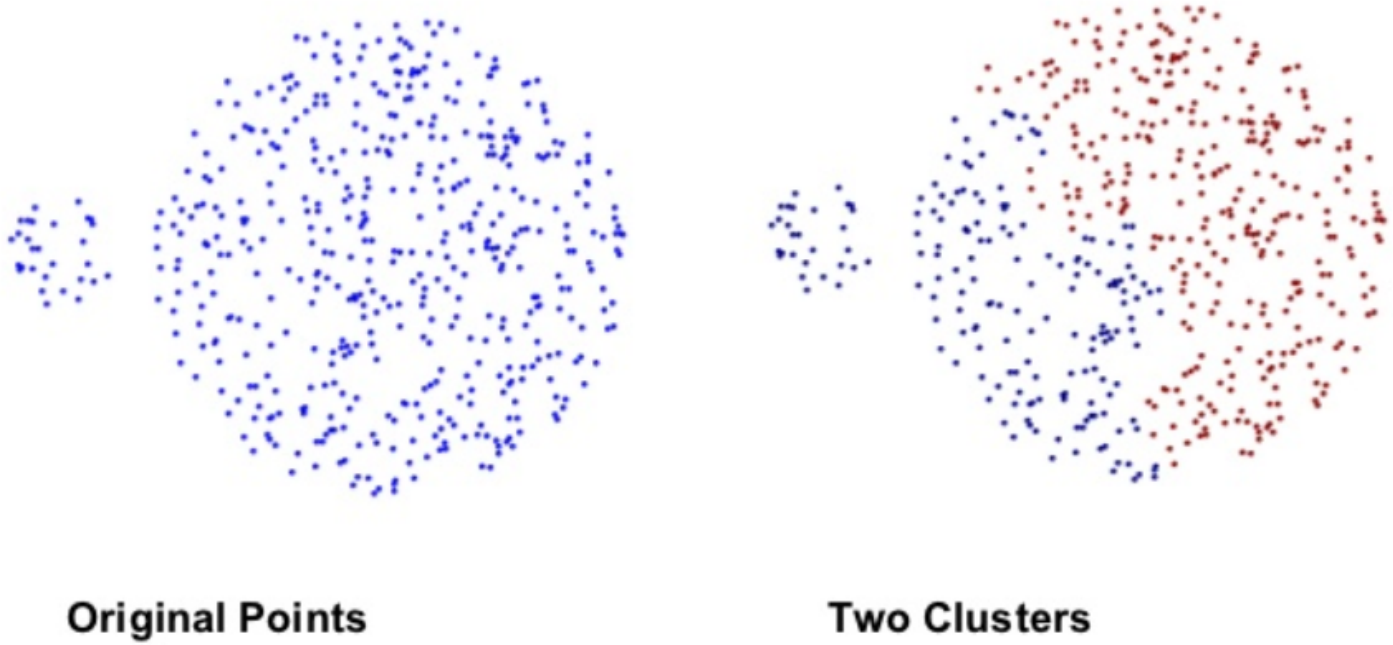
Original Points



Two Clusters

- Less sensitive to noise and outliers

HCA: Complete-Linkage criterion



- Biased toward spherical clusters
- Tends to break large clusters

HCA: Other linkages

Average-linkage

- Less noise sensitive, tends to favor hyper-spherical clusters
- High computational cost for roughly the same results as the centroid-linkage.

HCA: Other linkages

Average-linkage

- Less noise sensitive, tends to favor hyper-spherical clusters
- High computational cost for roughly the same results as the centroid-linkage.

Centroid-linkage

- A good compromise between single and complete link.
- Less sensitive to noise and outliers.
- tends to favor elliptical clusters.

HCA: Other linkages

Average-linkage

- Less noise sensitive, tends to favor hyper-spherical clusters
- High computational cost for roughly the same results as the centroid-linkage.

Centroid-linkage

- A good compromise between single and complete link.
- Less sensitive to noise and outliers.
- tends to favor elliptical clusters.

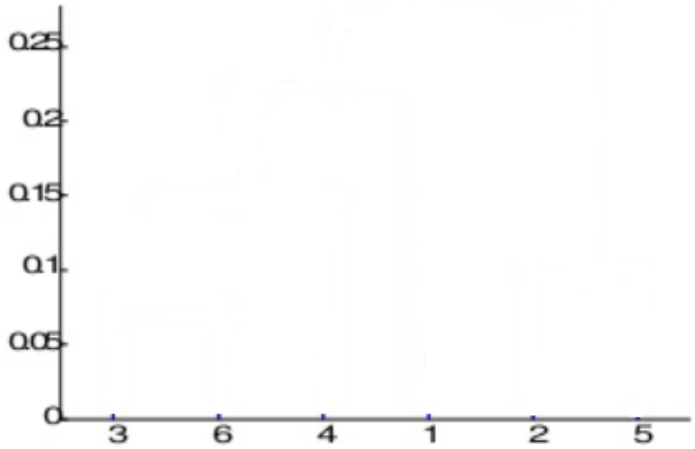
Ward's method

This similarity is based on the increase on the mean squared distance to the centroid when 2 clusters are merged.

- Less sensitive to noise and outliers.
- tends to favor elliptical clusters.

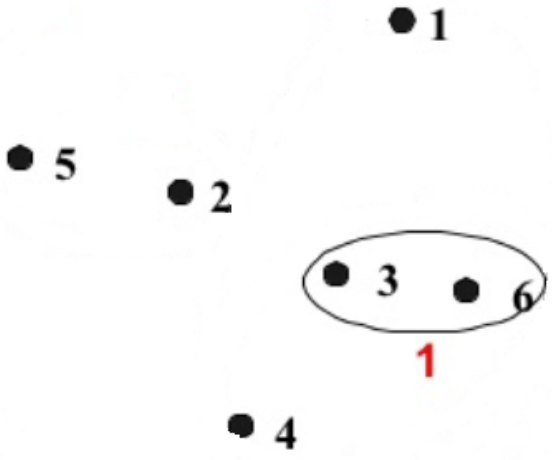
HCA: Example

Nested Clusters

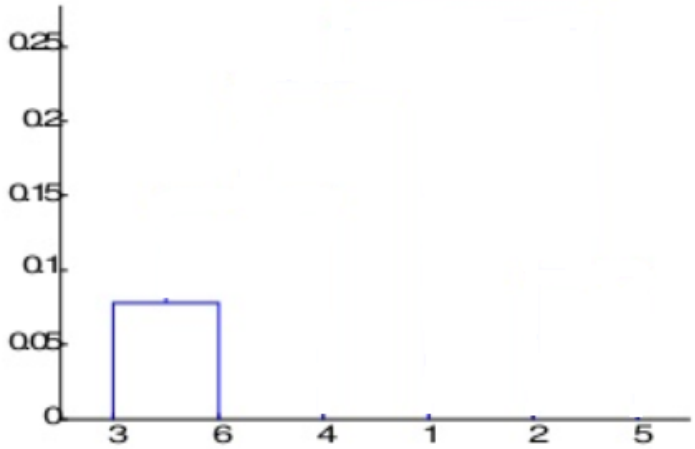


Dendrogram

HCA: Example

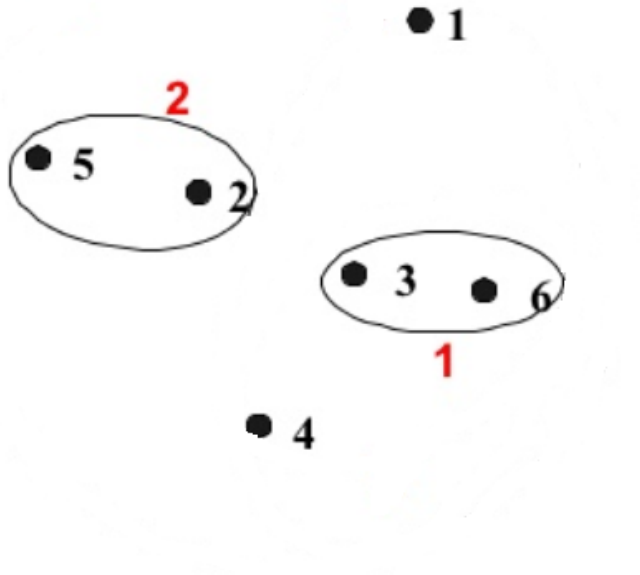


Nested Clusters

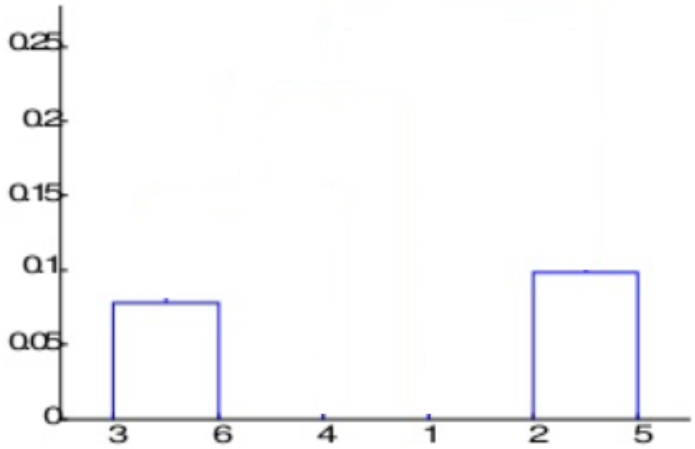


Dendrogram

HCA: Example

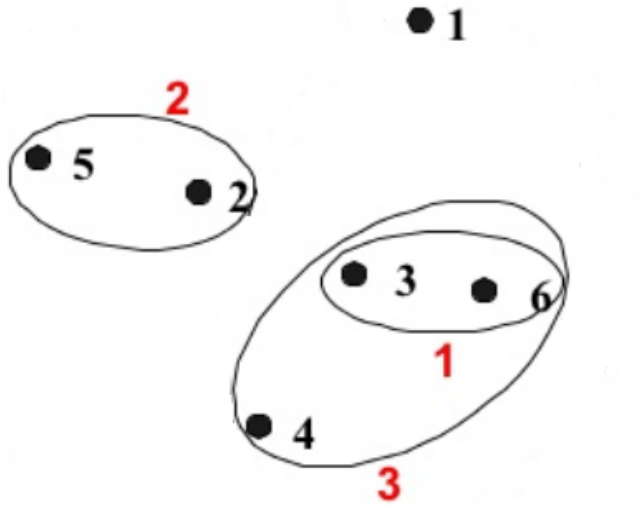


Nested Clusters

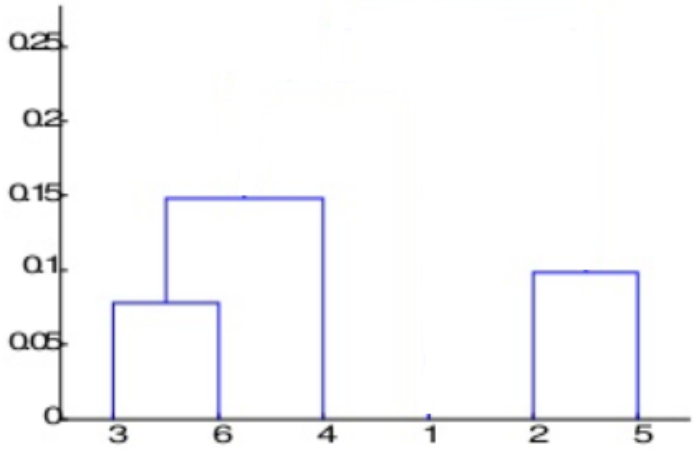


Dendrogram

HCA: Example

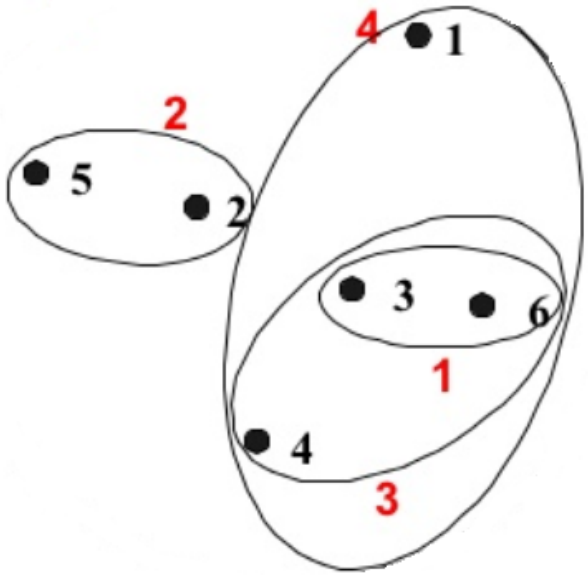


Nested Clusters

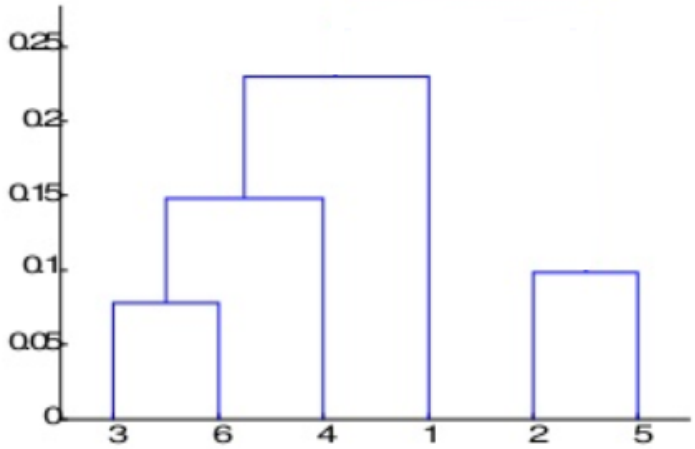


Dendrogram

HCA: Example

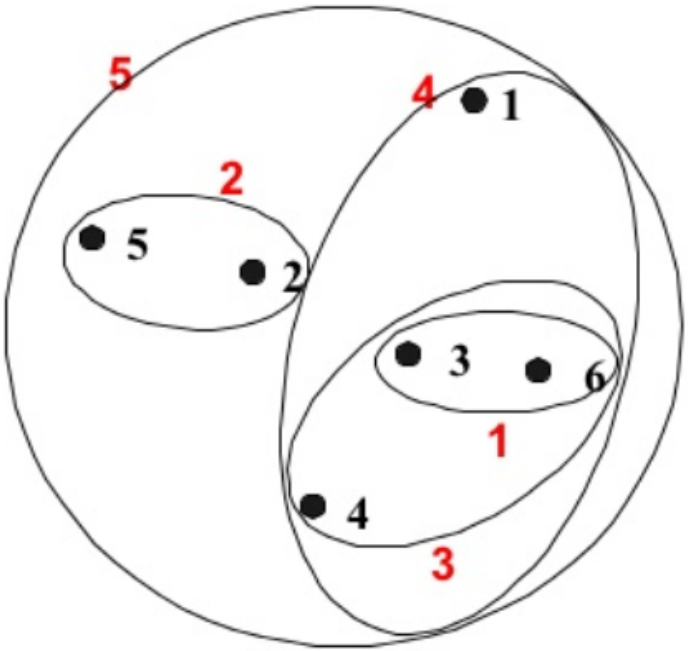


Nested Clusters

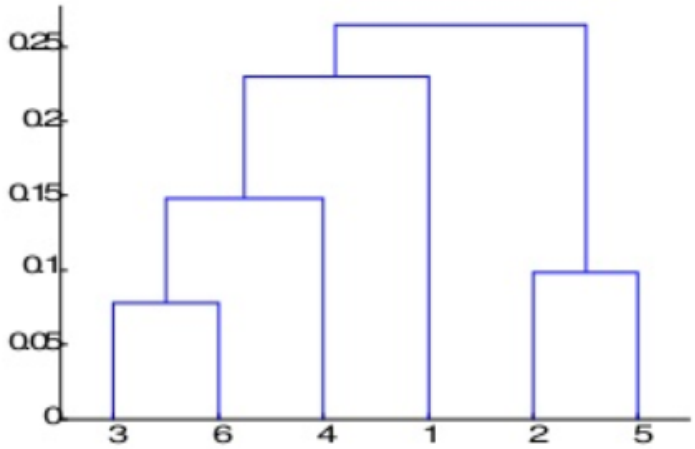


Dendrogram

HCA: Example

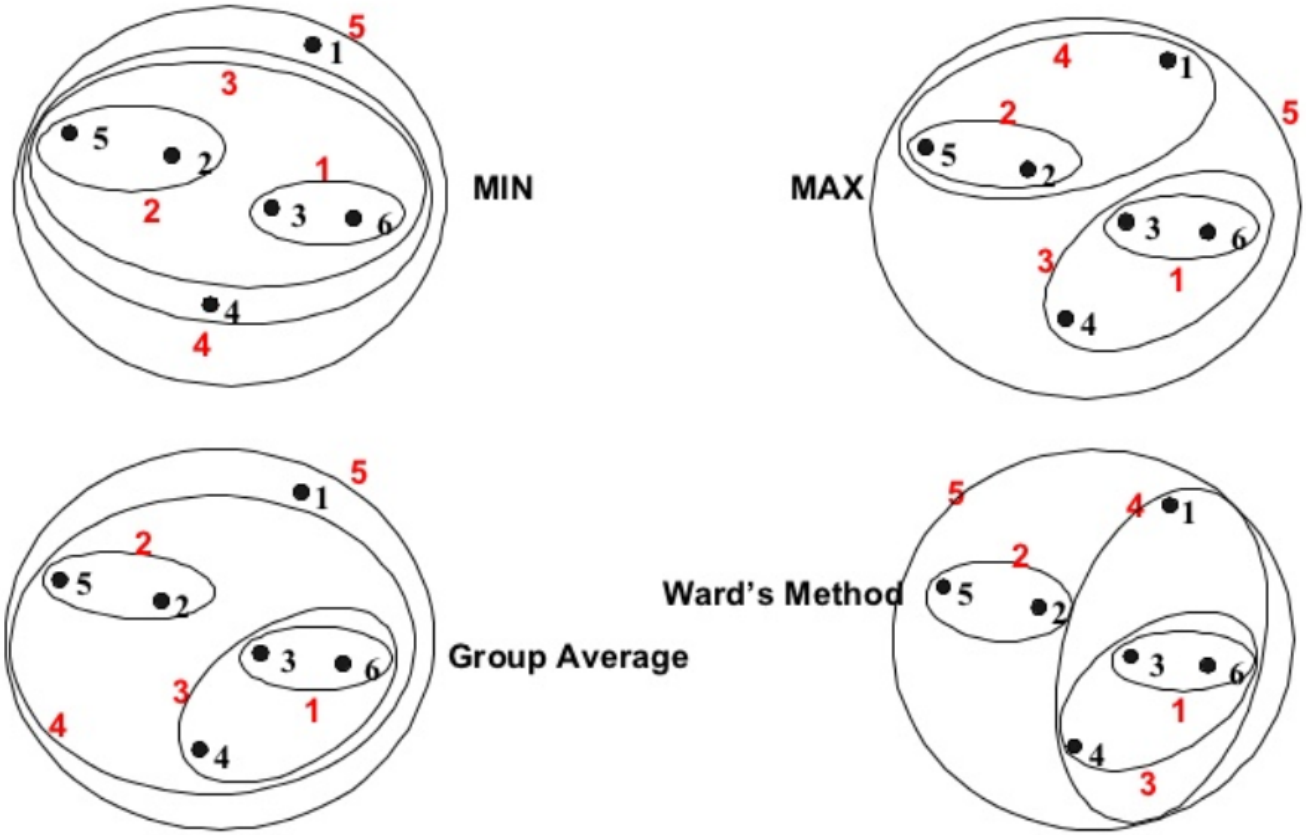


Nested Clusters



Dendrogram

HCA: Examples comparisons



K-Means: Introduction

The method of the K Means, or K-Means algorithm, is a special case of the mobile centers methods. Its principle consists in using a certain number of representatives (centroids, or prototypes) in the data space. Each of these representative will represent a cluster.

- In the end, each data is associated to its closest prototype in order to obtain the clustering partition.
- The groups formed with this process are homogeneous and well separated.

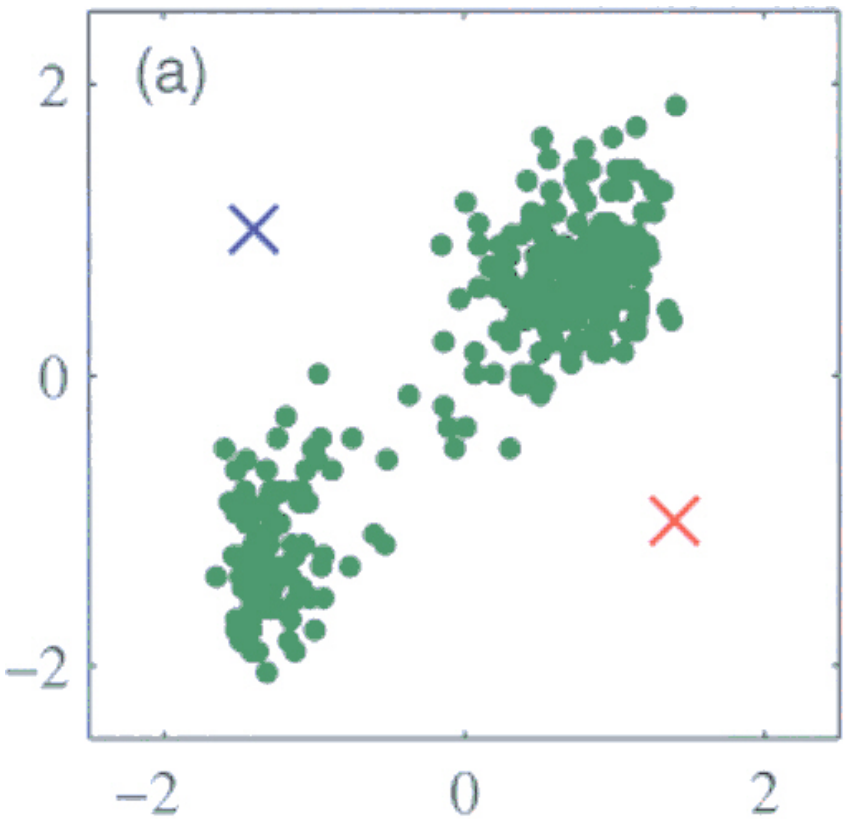
K-Means: Algorithm

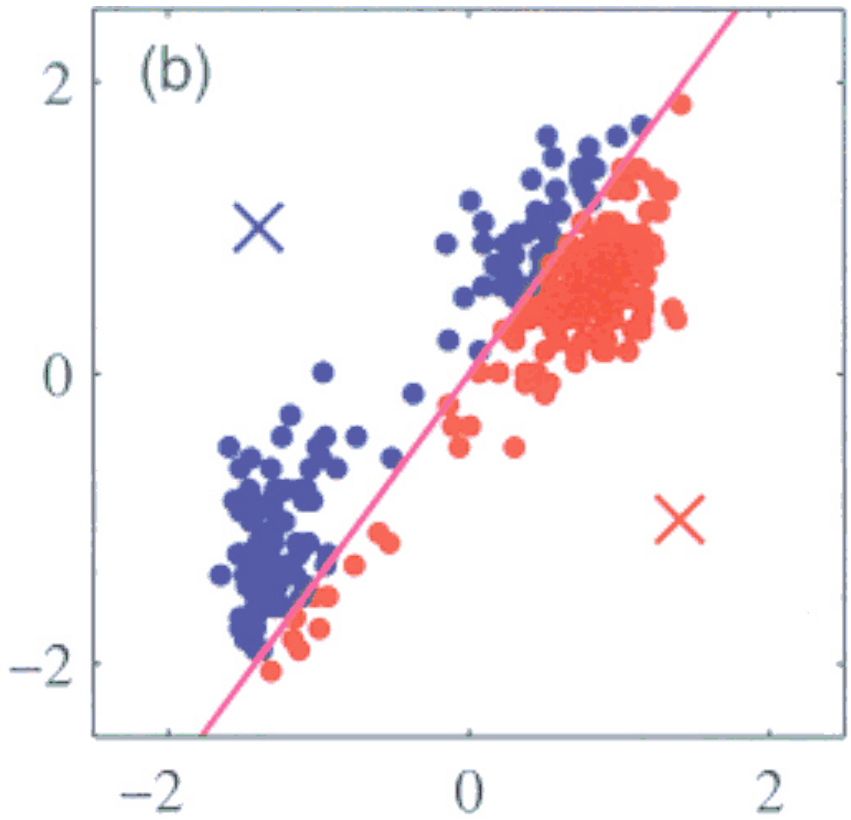
1. Randomly select K initial center.
2. Assign each data x to its closest center μ_i.
3. If the partition doesn't change: **stop**
4. Else, update the centers, each μ_i must be the gravity center of its cluster: $\mu_i = \frac{1}{|c_i|} \sum_{x \in c_i} x$
5. Go to 2

Like in most clustering algorithms, step 2 is dependent on a chosen distance function. For the K-Means algorithm, the Euclidian distance is usually prefered:

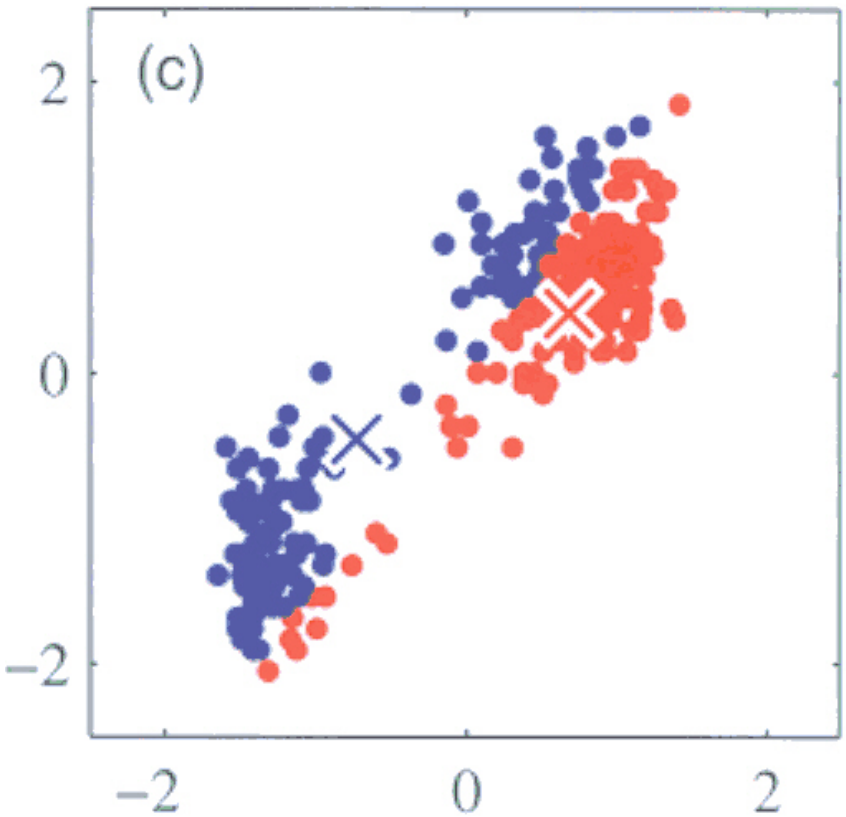
$$d(x, y) = \sqrt{\sum_{i=1}^{d} (x_i - y_i)^2}$$

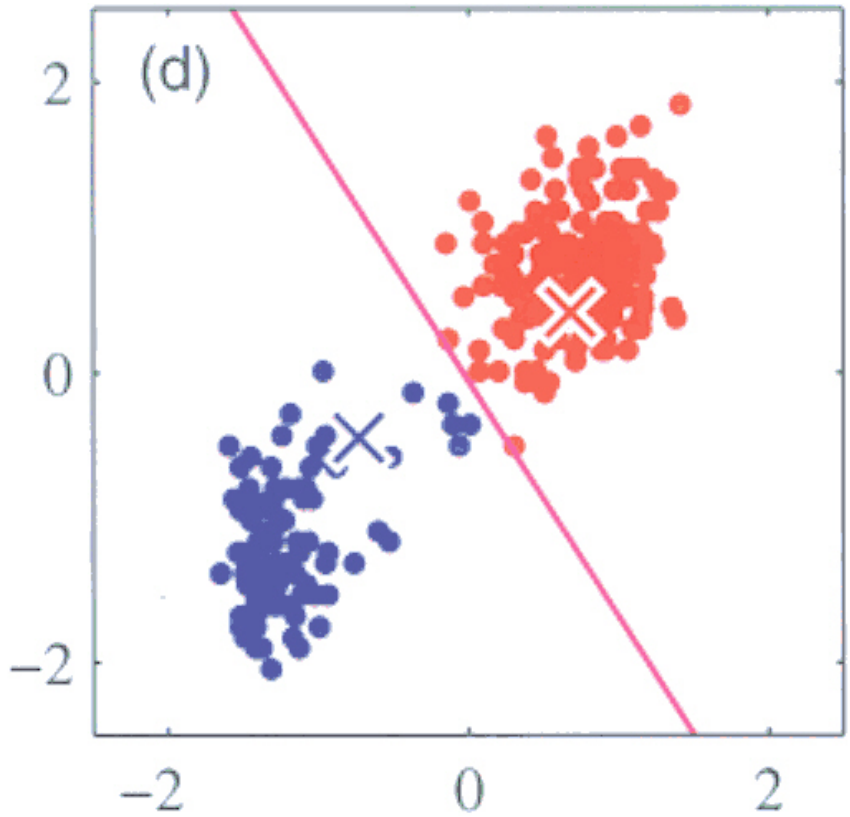
K-Means: Example



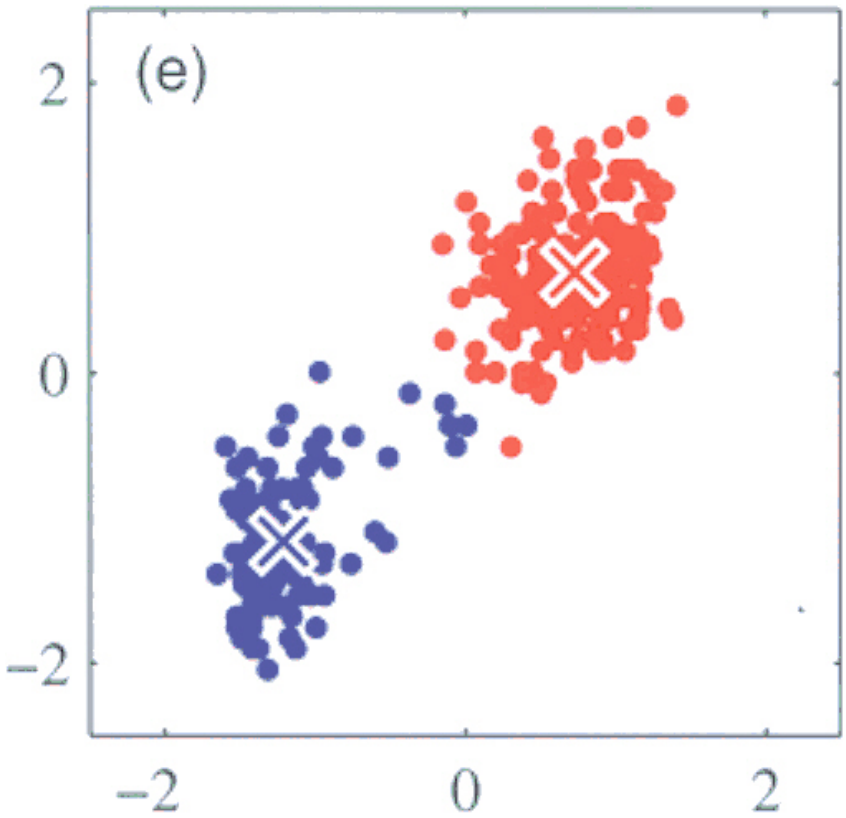


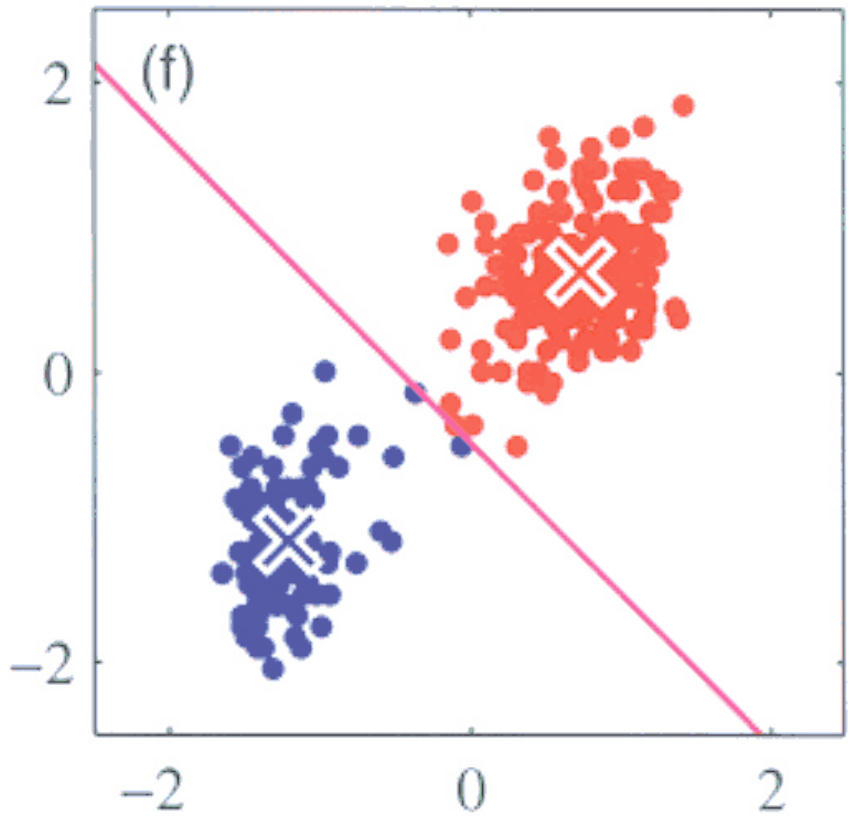
K-Means: Example



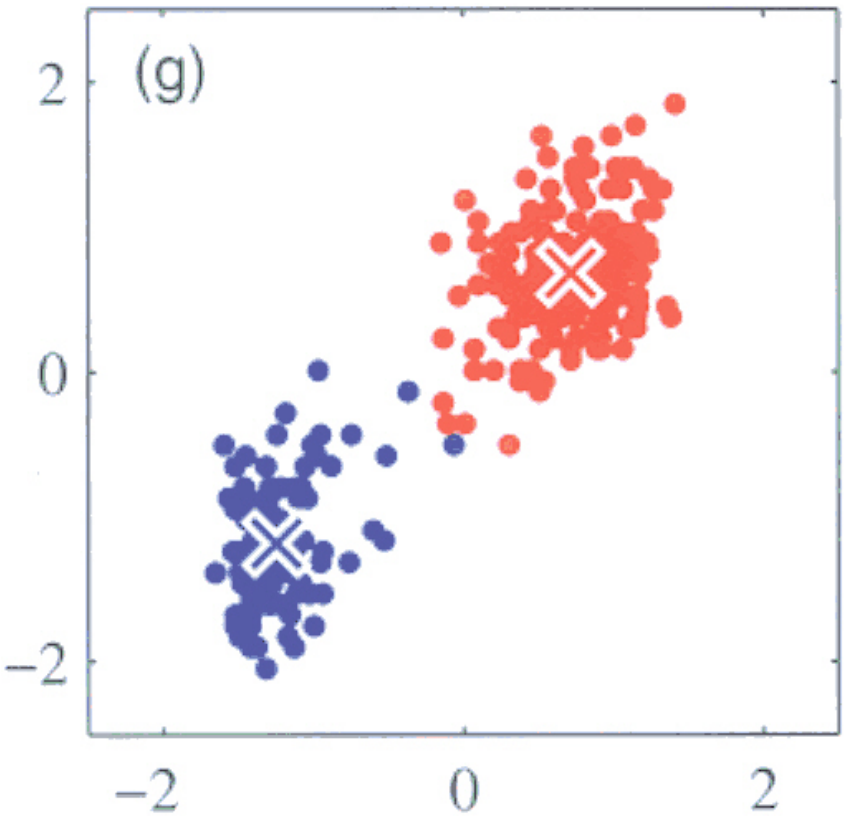


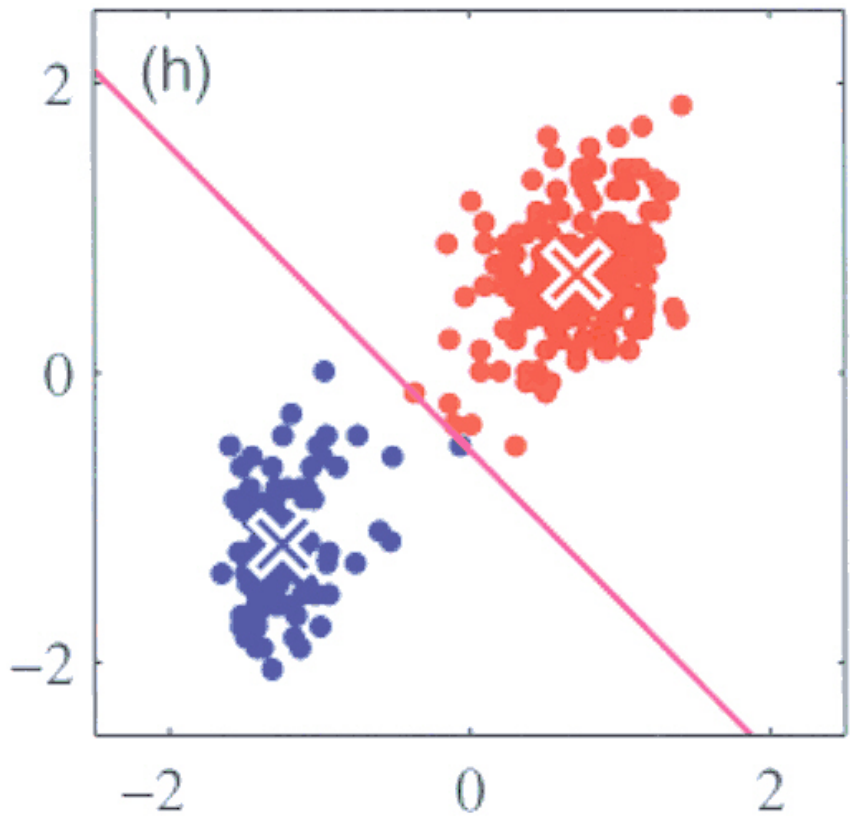
K-Means: Example





K-Means: Example





K-Means: Properties

- Guaranteed to monotonically decrease average squared distance in each iteration

$$L(\mu)^{(t)} = \sum_{i=1}^{N} \min_j ||\mu_j - x_i||_2^2$$

$$L(\mu)^{(t+1)} \leq L(\mu)^{(t)}$$

K-Means: Properties

- Guaranteed to monotonically decrease average squared distance in each iteration

$$L(\mu)^{(t)} = \sum_{i=1}^{N} \min_j ||\mu_j - x_i||_2^2$$

$$L(\mu)^{(t+1)} \leq L(\mu)^{(t)}$$

- Convergence to a local minimum

K-Means: Properties

- Guaranteed to monotonically decrease average squared distance in each iteration

$$L(\mu)^{(t)} = \sum_{i=1}^{N} \min_j ||\mu_j - x_i||_2^2$$

$$L(\mu)^{(t+1)} \leq L(\mu)^{(t)}$$

- Convergence to a local minimum
- Algorithmic complexity at each iteration : $O(n \cdot d \cdot K)$

K-Means: Properties

- Guaranteed to monotonically decrease average squared distance in each iteration

$$L(\mu)^{(t)} = \sum_{i=1}^{N} \min_j ||\mu_j - x_i||_2^2$$

$$L(\mu)^{(t+1)} \leq L(\mu)^{(t)}$$

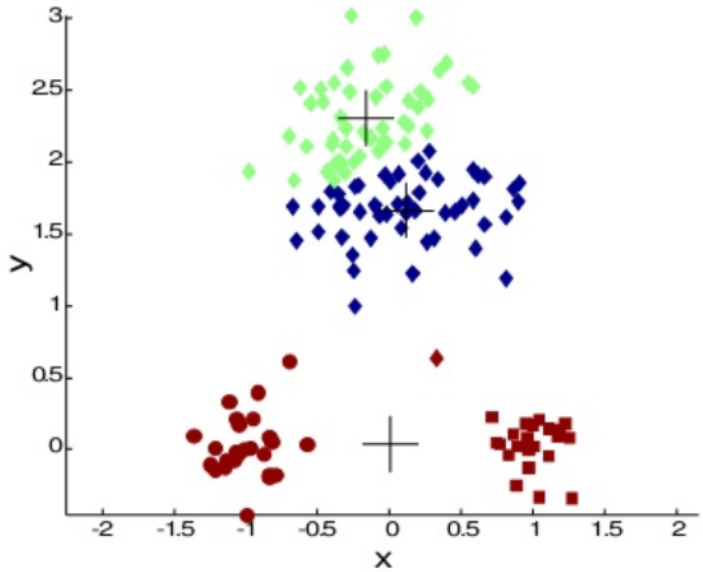
- Convergence to a local minimum
- Algorithmic complexity at each iteration : $O(n \cdot d \cdot K)$
- Non-convex optimization : **NP-hard problem**

K-Means: Limits

Instability

The K-Means algorithm is highly dependent on the initialization of the centers.

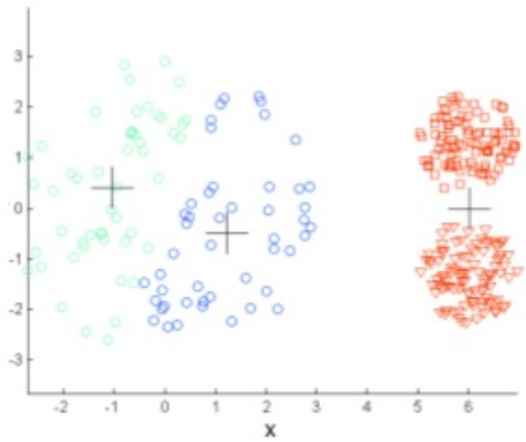
- This problem can be reduced by running the algorithm several time.
- It is also possible to initialize the centers using another clustering algorithm.



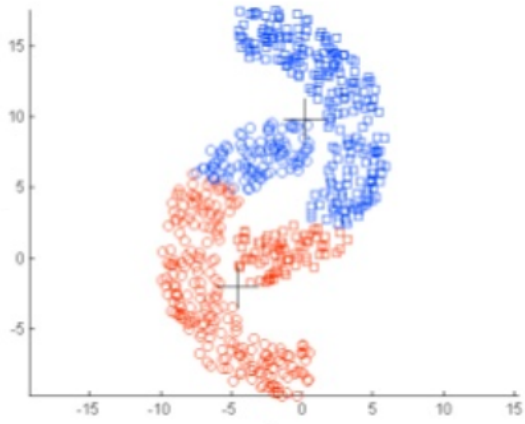
K-Means: Limits

Clusters shapes, size and density

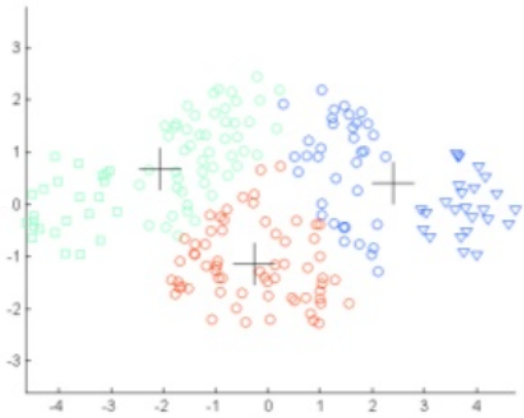
- The K-Means algorithm can only capture spherical or almost spherical clusters. Any non-convex shape will be missed.
- Clusters with different density or sizes can also be difficult to find using the K-Means algorithm.



K-means (3 Clusters)



K-means (2 Clusters)



K-means (3 Clusters)

K-Means: Limits

Choosing the number of clusters

The K-means algorithms requires the number of clusters "K" to be provided as a parameter. In purely exploratory data mining tasks, this number is not always known and must be guessed.

- Sometimes the only solution is to try several values for K and to see which gives the best results.

EM algorithm for the GMM: introduction

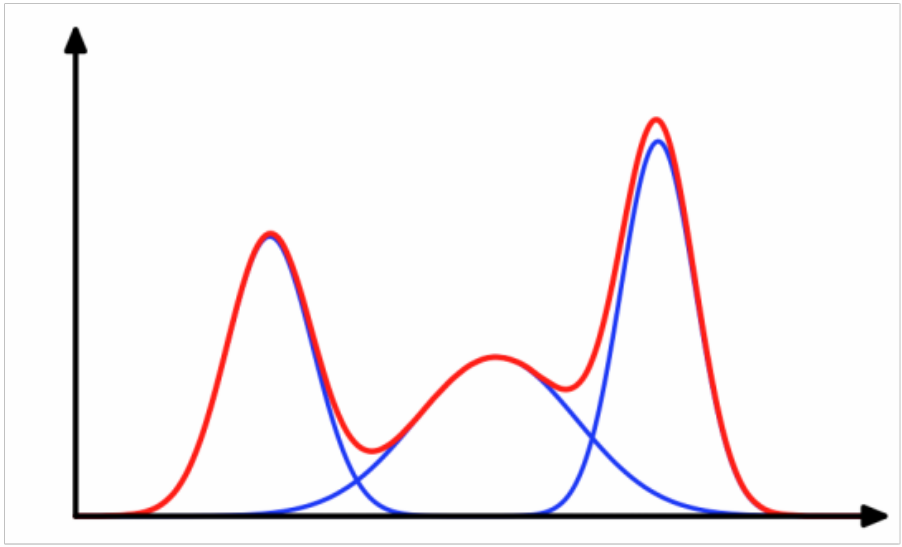
The Expectation-Maximization *algorithm (EM) is an iterative method used to find the maximum likelihood or the maximum a posteriori estimate of parameters in probabilistic and statistical models.*

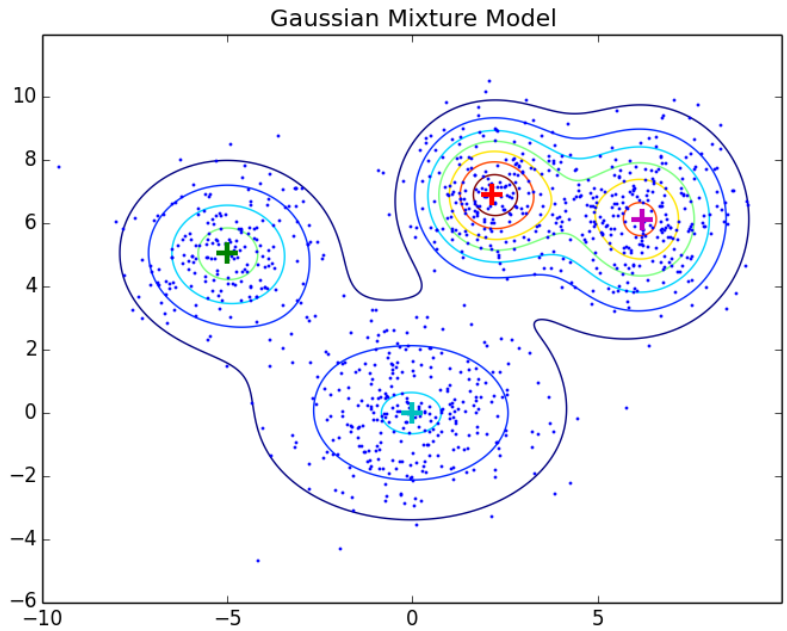
Adapatation to clustering

- In clustering, it is possible to represent clusters using parametric distribution: Each cluster has its own distribution with its own parameters.
- The EM algorithm can then be used to figure out these parameters and link each data to a cluster.

EM algorithm for the GMM: Gaussian mixture model

The Gaussian mixture model *consists in modeling clusters as Gaussians described by their mean value, their covariance matrix and a mixing probability*.





EM algorithm for the GMM: Gaussian mixture model

Gaussian model in dimension 1

- $\mu \in \mathbb{R}$ the mean value of the gaussian distribution
- $\sigma \in \mathbb{R}^+$ the standard deviation of the distribution

$$\mathcal{N}(\mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp^{-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}}$$

Gaussian model in dimension d

- $\mu \in \mathbb{R}^d$ the mean value of the gaussian distribution
- $\Sigma = (\sigma_{i,j})_{d\times d}$ the variance-covariance matrix of the distribution

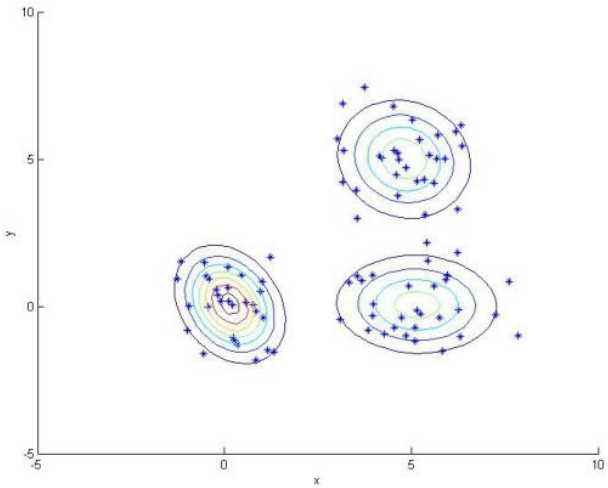
$$\mathcal{N}(\mu, \Sigma) = \frac{\exp^{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)}}{\sqrt{|\Sigma|(2\pi)^d}}$$

EM algorithm for the GMM: Gaussian mixture model

The Gaussian mixture model *consists in modeling clusters as Gaussians described by their mean value, their covariance matrix and a mixing probability*.

$$P(x_i \in c) = \frac{\pi_c \cdot \mathcal{N}(\mu_c, \Sigma_c, x_i)}{\sum_{k=1}^{K} \pi_k \cdot \mathcal{N}(\mu_k, \Sigma_k, x_i)}$$

On note $P(x_i|\Theta)$ la probabilité d'observer x_i sachant les paramètres de tous les clusters, avec $\sum_{c=1}^{K} \pi_c = 1$ et $\Theta = \{(\mu_1, \Sigma_1, \pi_1), \cdots, (\mu_K, \Sigma_K, \pi_K)\}$.



$$P(x_i|\Theta) = \prod_{i=1}^{K} \pi_k \cdot \mathcal{N}(\mu_k, \Sigma_k, x_i)$$

EM algorithm for the GMM: Optimization process

The goal of the EM algorithm is to find the parameters Θ that maximize the likelihood of generating the observed data:

$$\Theta_{ML} = \underset{\Theta}{\text{Argmax}}\{\log P(X|\Theta)\} = \underset{\Theta}{\text{Argmax}} \sum_{i=1}^{N} P(x_i|\Theta)$$

$$\Theta_{ML} = \underset{\Theta}{\text{Argmax}} \sum_{i=1}^{N} \log \left(\sum_{k=1}^{K} \pi_K \cdot \mathcal{N}(\mu_k, \Sigma_k, x_i) \right)$$

This is done via an iterative two-step process:

- **Expectation step**: Update the latent variables (cluster labels) using the current parameters Θ: assign each data to the cluster c that maximizes $p(x \in c)$.
- **Maximisation step**: Use the new latent variables to update the parameters: Update the mean value, covariance matrix and mixing probabilities of all clusters.

EM algorithm for the GMM: Update rules

E-Step

$$P(x_n \in c) = s_{n,c} = \frac{\pi_c \mathcal{N}(\mu_c, \Sigma_c, x)}{\sum_{k=1}^{K} \pi_k \mathcal{N}(\mu_k, \Sigma_k, x)} = \frac{1}{Z} \pi_c \frac{\exp^{-\frac{1}{2}(x-\mu_c)^T \Sigma_c^{-1}(x-\mu_c)}}{\sqrt{|\Sigma_c|(2\pi)^d}}$$

M-Step

$$N_c = \sum_{n=1}^{N} s_{n,c} \implies \pi_c = \frac{N_c}{N}$$

$$\mu_c = \frac{1}{N_c} \sum_{n=1}^{N} s_{n,c} \cdot x_n$$

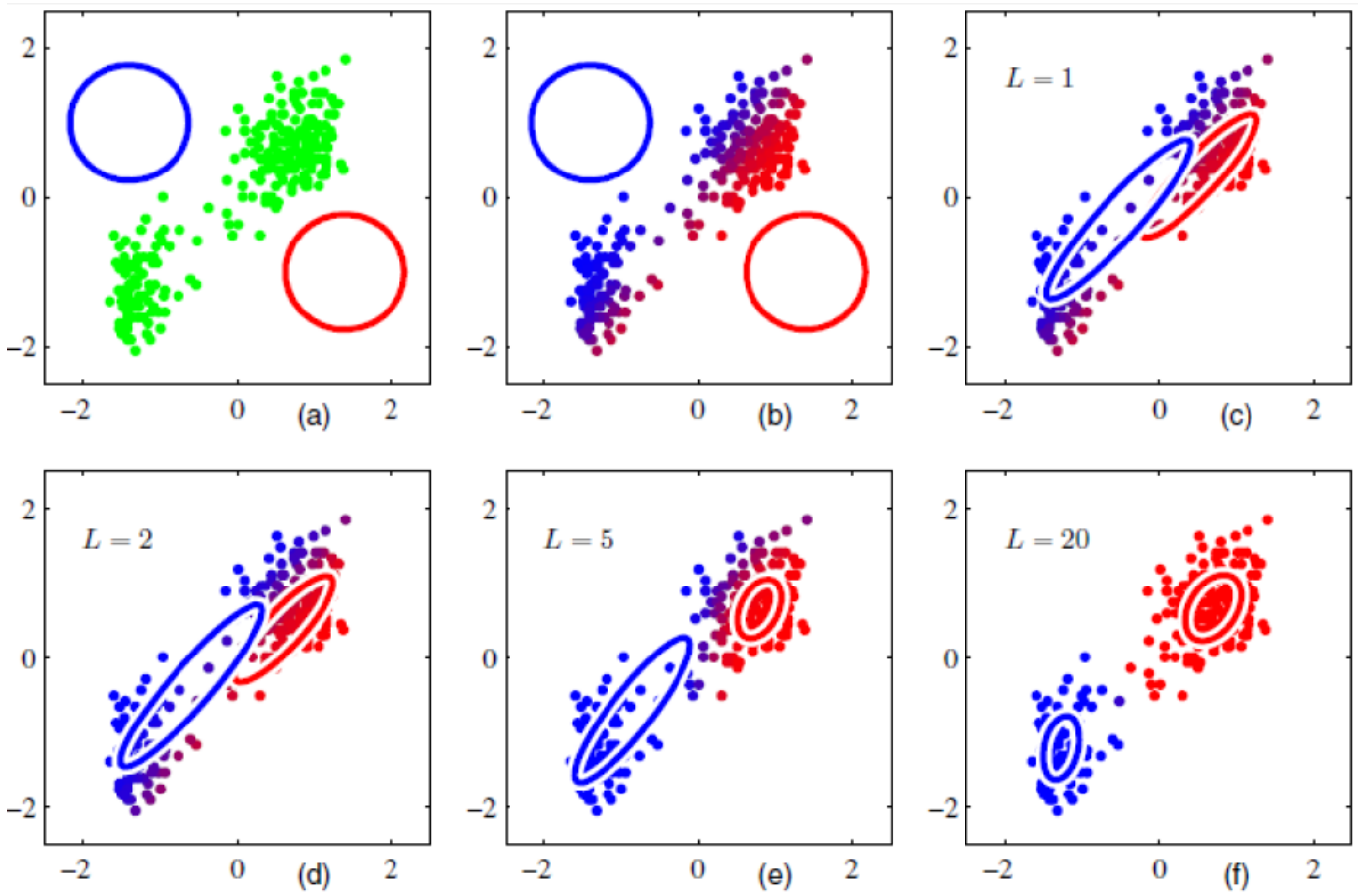
$$\Sigma_c = \frac{1}{N_c} \sum_{n=1}^{N} s_{n,c} \cdot (x_n - \mu_c)(x_n - \mu_c)^T$$

EM algorithm for the GMM: Algorithm

Algorithm 1: EM Algorithm for the GMM

Initialize Θ randomly
while *the partition S is not stable* **do**
    E-Step: evaluate $S = argmax_S p(S|X, \Theta)$
    forall $x_n \in X$ **do**
        $s_n(k) = \frac{1}{Z}\pi_c \mathcal{N}(x_n, \mu_c, \Sigma_c)$
    end
    M-Step: Re-evaluate Θ
    forall $c \in [1..K]$ **do**
        Update μ_c, Σ_c and π_c
    end
end

EM algorithm for the GMM: Example



EM algorithm for the GMM: Strengths

- Is tolerant to clusters with different densities
- Can detect clusters that are in contact
- Strong statistical model:
  - Can be adapted to other types of mixtures (Bernouilli, Multinomials, etc.)
  - Relies on probabilistic models that are very common for modelisation
  - Existence of a convergence proof

EM algorithm for the GMM: Limits

- The EM algorithm for the GMM model detects only ellipsoid clusters.
- Computing Σ^{-1} the inverse of the variance-covariance matrix can be both difficult and time consuming in high dimension.
  - Can be solved by using diagonal matrices, but it is less accurate.
- Very much like the K-Means algorithm, the K needs to be provided and the EM algorithm is dependent on its initialization.
  - Can be improved by initializing the EM algorithm using K-Means or HCA ...

EM algorithm for the GMM: Limits

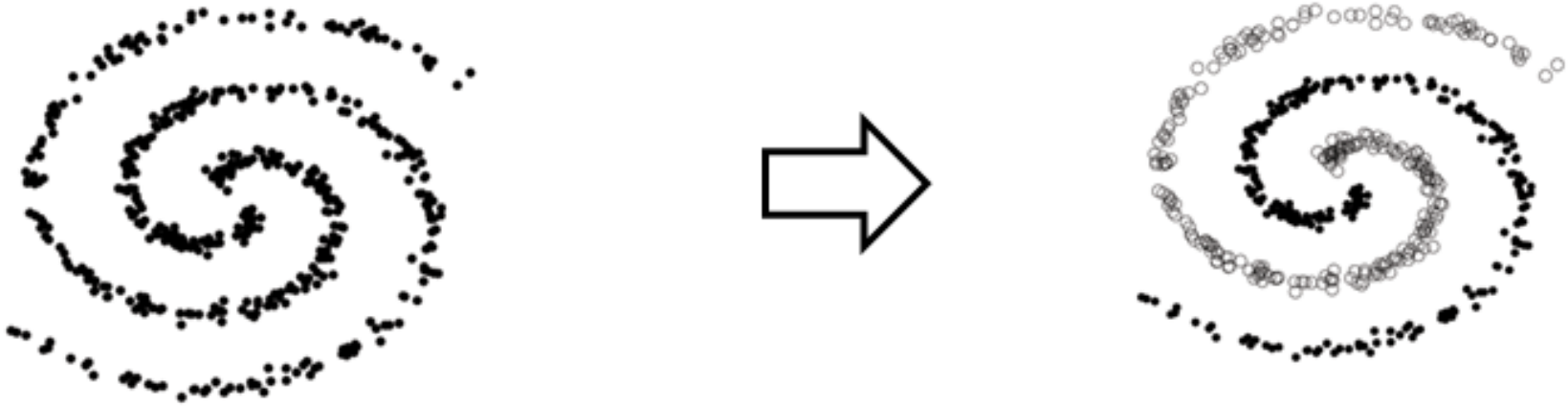
- The EM algorithm for the GMM model detects only ellipsoid clusters.
- Computing Σ^{-1} the inverse of the variance-covariance matrix can be both difficult and time consuming in high dimension.
  - Can be solved by using diagonal matrices, but it is less accurate.
- Very much like the K-Means algorithm, the K needs to be provided and the EM algorithm is dependent on its initialization.
  - Can be improved by initializing the EM algorithm using K-Means or HCA ...

Remark

The K-Means algorithm is a degenerate case of EM algorithm for the GMM where $\Sigma = I_d$ and $\forall c, \pi_c = \frac{1}{K}$.

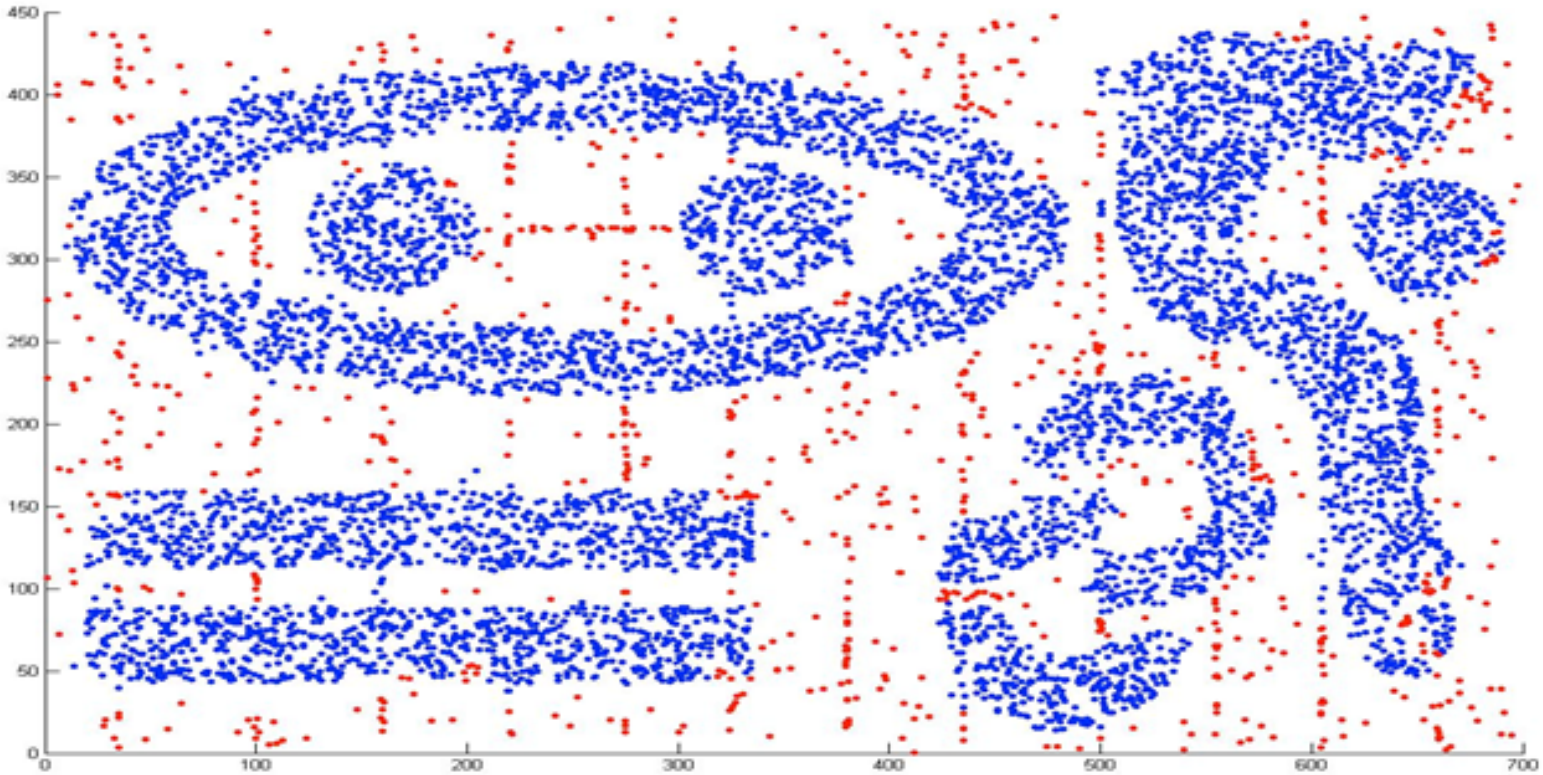
Spectral clustering

- Simple, yet powerful clustering method
- Requires less assumptions on the form of the clusters.
- Outperforms the traditional clustering approaches.
- Reasonably fast for sparse data, very slow otherwise.



Difficult clusterings

- Some data don't lend themselves for a centroid, a prototype-based, or a density-based clustering.
- Spectral clustering proposes a different approach.



Difficult clusterings

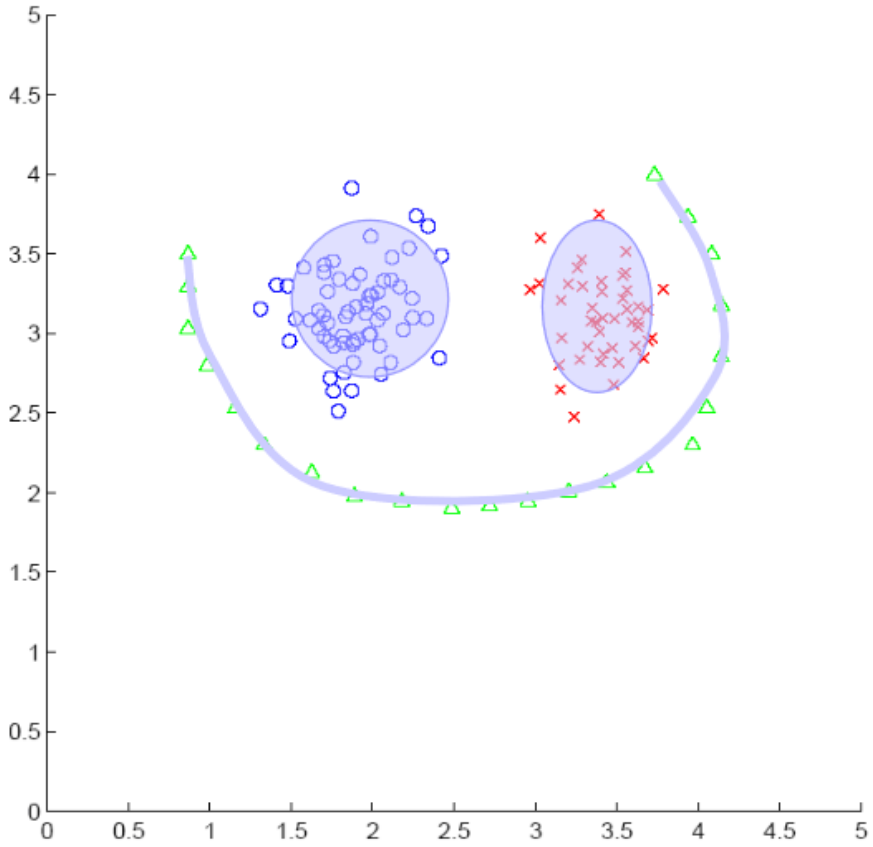


Figure: Example of a partitioning that most clustering methods cannot find

Spectral clustering and graph theory

Spectral clustering treats the data clustering as a graph partitioning problem:

- Let us note $X = \{x_1, \cdots, x_n\}, x_i \in \mathbb{R}^d$ the data.
- The points are characterized by pairwise similarities: To each pair (x_i, x_j) we can associate a similarity value $w_{ij} = f(d(x_i, x_j), \theta)$.
  - The most common model is:

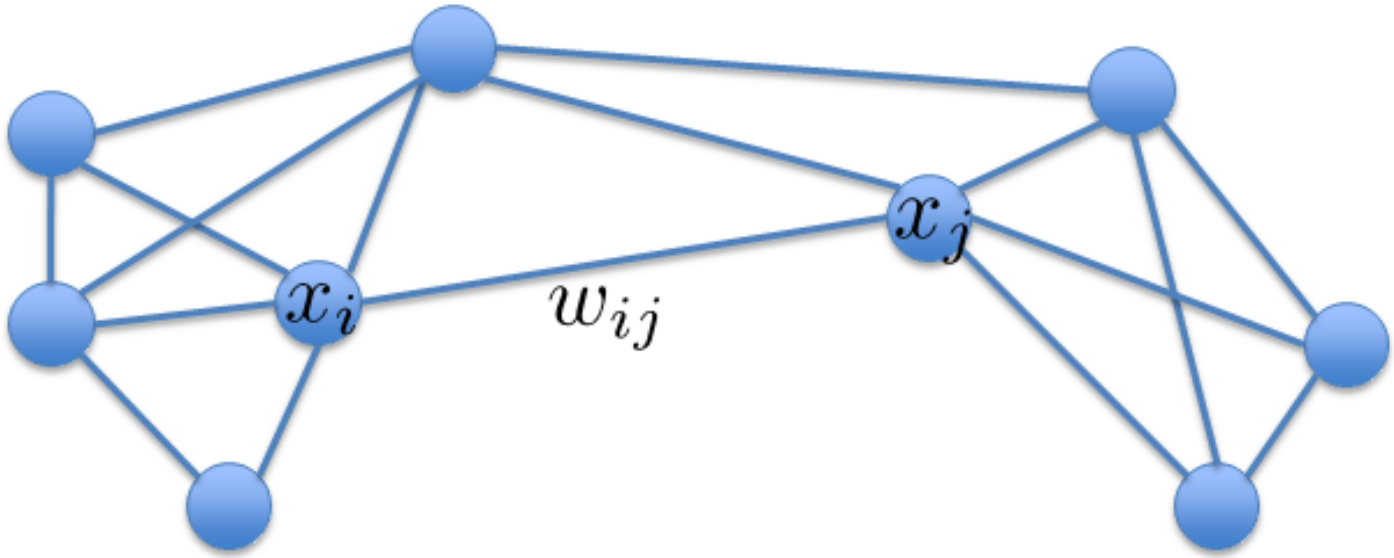
$$w_{i,j} = \exp\left(\frac{-1}{2\sigma^2}||x_i - x_j||_2^2\right)$$

- Let us note $W = (w_{ij})_{n \times n}$ this similarity matrix.

Spectral clustering: Graph construction

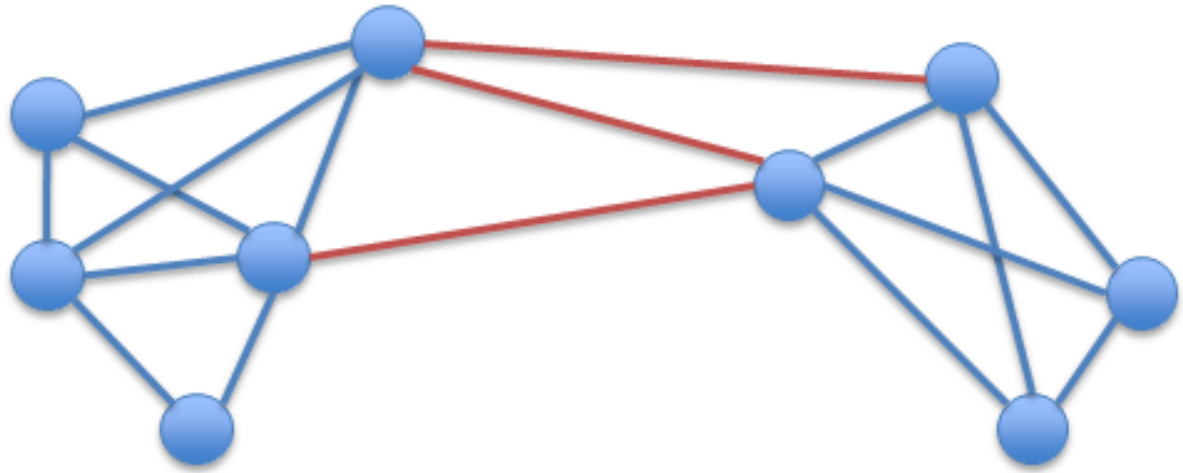
There are different ways to build a graph based on the similarity matrix:

- Fully connected graphs: All vertices having non-null similarities are connected to each other.
- Radius-based graphs: Each vertex is connected to all the vertices falling inside a ball of similarity of radius r.
- K-nearest neighbor graphs: Each vertex is connected to its K most similar neighbors.
- Hybrid KNN and radius-based graphs



Spectral clustering and graph partitioning

- In spectral clustering, the goal is to find two clusters that have the minimum weight sum connections.



This can be done by minimizing:

$$Cut(c_1, c_2) = \sum_{i \in c_1} \sum_{j \in c_2} w_{i,j}$$

Spectral clustering and graph partitioning

It is easy to prove that the previous MinCut problem can be written as:

$$J_{MinCut} = \frac{1}{4} q^T (D - W) q$$

where D is a diagonal matrix so that $d_{ii} = \sum_{j=1}^{n} w_{ij}$, and q is a vector of size n so that:

$$q_i = \begin{cases} 1 & \text{if } x_i \in c_1 \\ -1 & \text{if } x_i \in c_2 \end{cases}$$

- We are therefore looking for the vector/partition q that minimizes this equation.
- This can be done by finding the eigenvalues and eigenvectors of the Laplacian matrix $L = D - W$.

Spectral clustering and graph partitioning

Ideally, we also would like to maximize the intra-cluster similarity:

- First constraint: Minimize $Cut(c_1, c_2)$
- Second constraint: Maximize $Cut(c_1, c_1)$ and $Cut(c_2, c_2)$

This can be done by minimizing the following objective function:

$$J_{Ncut}(c_1, c_2) = Cut(c_1, c_2)\left(\frac{1}{\sum_{i \in c_1}\sum_{j=1}^{n} w_{ij}} + \frac{1}{\sum_{i \in c_2}\sum_{j=1}^{n} w_{ij}}\right)$$

- It is proved that this minimization is obtained through the second smallest eigenvector of the matrix $L_{Ncut} = D^{-1/2}(D - W)D^{-1/2}$

Spectral clustering: steps

Pre-processing

- Compute the similarity matrix and build the graph

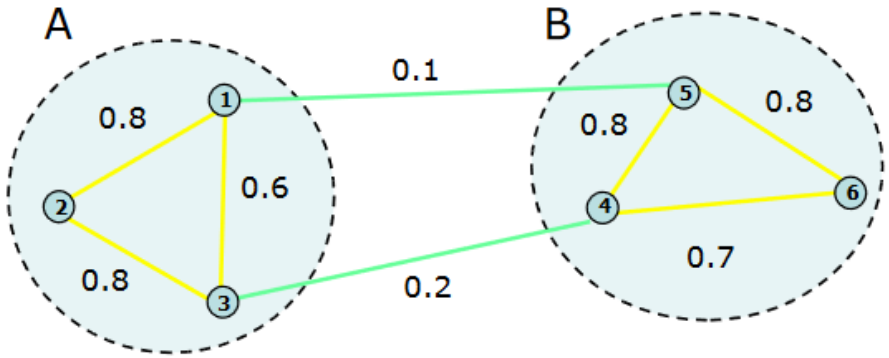
Spectral representation

- Compute the associated Laplacian Matrix.
- Compute the eigenvalues and eigenvectors.

Clustering

- Assign each point to a cluster depending on the values in the second eigenvector.

Spectral clustering: Example



	x_1	x_2	x_3	x_4	x_5	x_6
x_1		0.8	0.6	0	0.1	0
x_2	0.8		0.8	0	0	0
x_3	0.6	0.8		0.2	0	0
x_4	0.8	0	0.2		0.8	0.7
x_5	0.1	0	0	0.8		0.8
x_6	0	0	0	0.7	0.8	

Spectral clustering: Example

Pre-processing:

- Build the Laplacian Matrix $L = D - W$.

Decomposition:

- Find the eigenvalues Λ.
- Find the eigenvectors

Clustering:

- Assign each data to a cluster depending on the sign of the 2nd eigenvector components.

X_1	1.5	-0.8	-0.6	0	-0.1	0
X_2	-0.8	1.6	-0.8	0	0	0
X_3	-0.6	-0.8	1.6	-0.2	0	0
X_4	-0.8	0	-0.2	2.5	-0.8	-0.7
X_5	-0.1	0	0	0.8	1.7	-0.8
X_6	0	0	0	-0.7	-0.8	1.5

$\Lambda =$

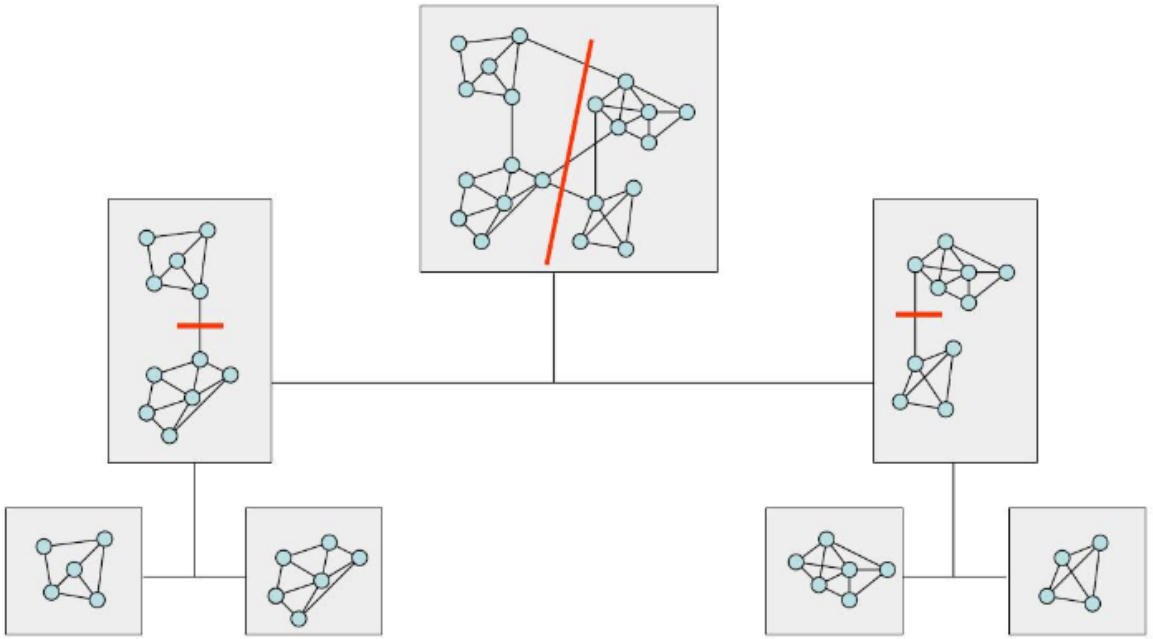
0.0
0.3
2.2
2.3
2.5
3.0

0.4	0.2	0.1	0.4	-0.2	-0.9
0.4	0.2	0.1	-0.	0.4	0.3
0.4	0.2	-0.2	0.0	-0.2	0.6
0.4	-0.4	0.9	0.2	-0.4	-0.6
0.4	-0.7	-0.4	-0.8	-0.6	-0.2
0.4	-0.7	-0.2	0.5	0.8	0.9

Extension to several clusters

This method so far only allows to have 2 clusters. We would like to have more !

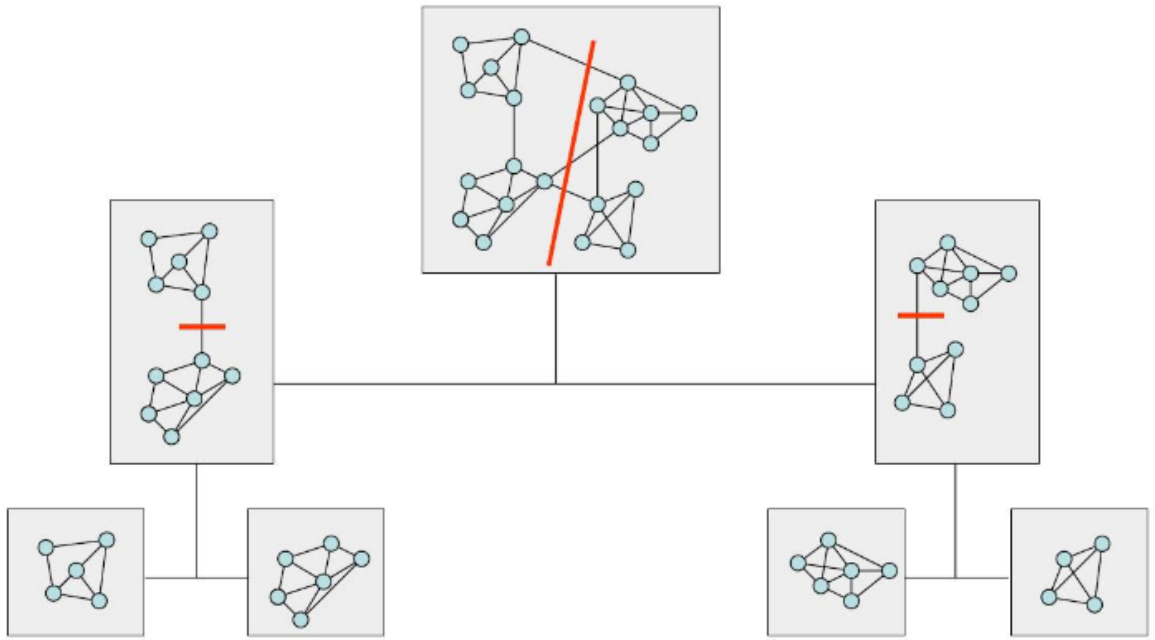
- Method 1: Recursively applying the algorithm in a hierarchical divisive manner.



Extension to several clusters

This method so far only allows to have 2 clusters. We would like to have more !

- Method 1: Recursively applying the algorithm in a hierarchical divisive manner.



Weaknesses: Time consuming, inefficient, unstable.

Extension to several clusters

- Method 2: Change the objective function to handle K clusters.

Example:

$$J_{Ncut}(c_1, \cdots, c_K) = \sum_{i=1}^{K} \frac{Cut(c_i, \bar{c}_i)}{\sum_{j \in c_i} \sum_{k=1}^{n} w_{jk}}$$

Extension to several clusters

- Method 2: Change the objective function to handle K clusters.

Example:

$$J_{Ncut}(c_1, \cdots, c_K) = \sum_{i=1}^{K} \frac{Cut(c_i, \bar{c}_i)}{\sum_{j \in c_i} \sum_{k=1}^{n} w_{jk}}$$

Weaknesses: The optimization process is tedious and often mathematically unpractical.

Extension to several clusters

- Method 3: Use the other eigenvectors

Combining the K-Means algorithm and spectral clustering

- The matrix U containing eigenvectors 2 to K can be used as a low dimension representation of the data.
- The matrix must first go through a unit normalization process:

$$Y_{ij} = \frac{U_{ij}}{\sqrt{\sum_{j=1}^{n} U_{ij}^2}}$$

- After a unit normalization of each row, the K-Means algorithm (or any other clustering algorithm) can be applied to Y to obtain a partition with K clusters.

Spectral clustering for dimensionality reduction

To use "Spectral clustering" for dimensional reduction purposes, the eigenvectors need to be transformed in order to keep good topological properties. The clustering step is then removed as it is not needed here.

Transforming the eigenvectors into good low dimensional representations

- Let us define U the matrix containing eigenvectors 2 to d can be used to represent the data in a d-dimensional space.
- The matrix must first go through a unit normalization process:

$$Y_{ij} = \frac{U_{ij}}{\sqrt{\sum_{j=1}^{n} U_{ij}^2}}$$

Outline

Picking a similarity measure

Whatever the type of clustering algorithm, they all rely on a similarity measure. Picking a good similarity measure is therefore of paramount importance.

- It is the core of any model.
- It determines what type of structures are considered interesting.
- It will ultimately change the clustering result.

Remarks

- Choosing a similarity measure, or building one often require a good knowledge of the data to analyze, or at least of the field they come from.
- Picking a similarity measure already introduces a bias in a task that is supposed to be exploratory.

Picking a similarity measure

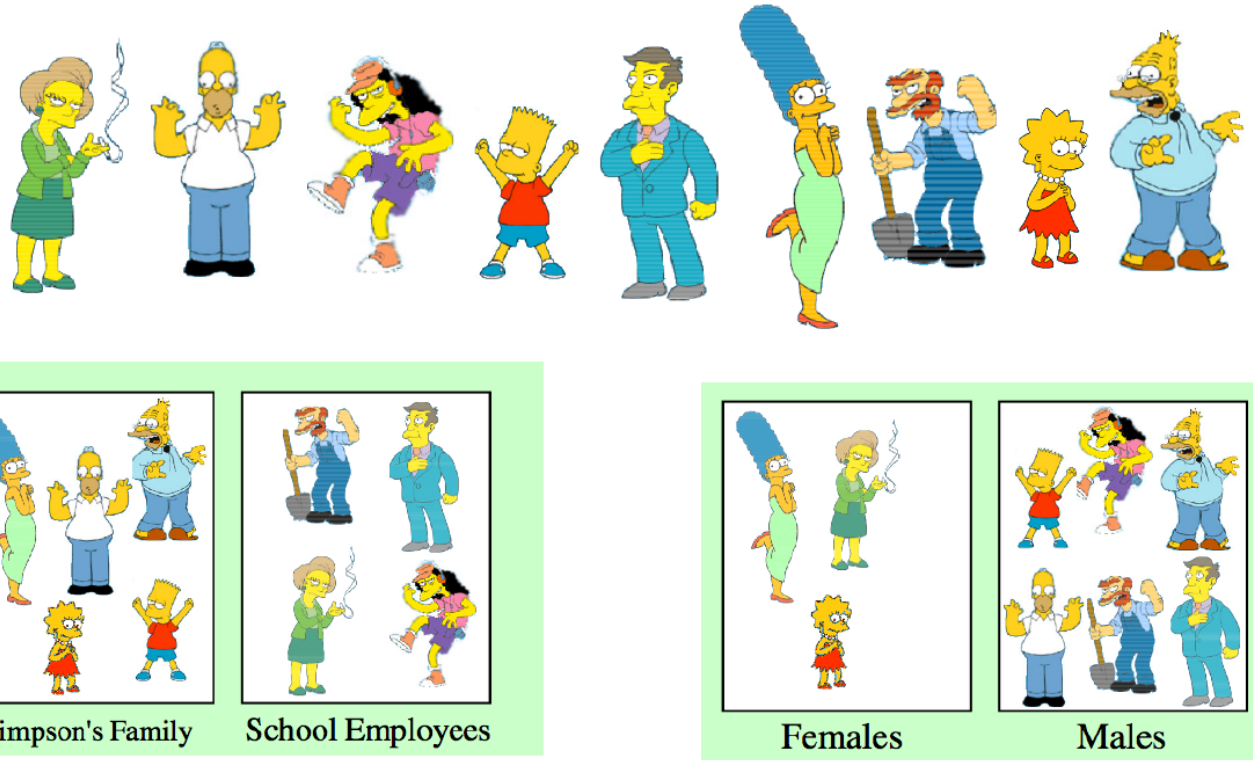


Figure: Depending on the similarity measure, both partitions can make sense.

Validating clustering results

Validating clustering results is a difficult process:

- There is no proper definition of what a "good cluster" is, or looks like.
- There is no "ground truth" in unsupervised learning to check the quality of a partition.
- Clusters come in all shapes and forms.
- The number of clusters to be found is usually unknown, and sometimes there are none.
- Not all data sets have well-defined clusters.

Identifying clusters

While it is usually obvious to visually spot the clusters in 2D or 3D data sets, it is almost impossible to do so when there are more features !

Internal indexes

Internal indexes *are criteria that can be used to evaluate the quality of a clustering partition.*

- Internal indexes can be subjective as they favor some cluster shapes, and can be biased toward a lower number of clusters.
- They usually assess the compactness of the clusters and whether they are well separated.

Remark

Internal indexes are said to be "internal", by opposition to "external indexes" that compare a result with an external ground truth.

Internal indexes: Davies-Bouldin Index

Let S_i be the average scatter of a cluster c_i around its mean value μ_i:

$$S_i = \frac{1}{|c_i|} \sum_{x \in c_i} ||x - \mu_i||_2$$

Let $M_{i,j} = ||\mu_i - \mu_j||_2$ be the average distance between two clusters c_i and c_j.

Davies-Bouldin Index for K clusters

$$DB = \frac{1}{K} \sum_{i=1}^{K} \max_{j \neq i} \frac{S_i + S_j}{M_{i,j}}$$

Internal indexes: Davies-Bouldin Index

The Davies-Bouldin index has the following properties:

- A lower DB value means a better clustering.
- This index is not normalized.
- It favors spherical clusters.
- it is biased so that it gives lower values with less clusters.

Remark

The Davies-Bouldin index is usually the favored internal index used to evaluate partitions created using the K-Means algorithm.

Internal indexes: Silhouette Index

The Silhouette index is another internal index with interesting properties:

- It is normalized between -1 and 1 (a value bellow 0 meaning a bad partition).
- It can evaluate if a a data belongs to a cluster, if a cluster is well formed, or the whole partition.

Let $a_x \simeq ||x - \mu_i||_2$ be the mean distance between $x \in c_i$ and the data that belong to the same cluster. And b_x the mean distance to the data that belong to other clusters.

$$b_x = \frac{1}{N - |c_i|} \sum_{y \notin c_i} ||x - y||_2 = \sum_{k \neq i} \frac{|c_k|}{N - |c_i|} ||x - \mu_k||_2$$

Silhouette index of an element x

$$SC(x \in c_i) = \frac{b_x - a_x}{max(a_x, b_x)}$$

Internal indexes: Silhouette Index

Silhouette index of a cluster c_i

$$SC(c_i) = \frac{1}{|c_i|} \sum_{x \in c_i} SC(x)$$

Silhouette index of a partition

$$SC = \frac{1}{K} \sum_{i=1}^{K} SC(c_i)$$

- The silhouette index best value is 1.
- It favors spherical clusters.

Range	Interpretation
$S > 0.70$	Strong structures have been found
$0.5 < S < 0.70$	reasonable structures have been found
$0.25 < S < 0.5$	Weak structures have been found
$0 < S < 0.25$	No substantial structure found
$S < 0$	This clustering is most likely garbage

Table: Interpreting Silhouette values

Internal indexes: Calinski & Harabasz

This index is defined as follows:

$$CH(k) = \frac{B/(k-1)}{W/(N-k)}$$

- B is the sum of square distances between clusters
- W is the sum of square distances within clusters
- k is the number of clusters

Properties of the CH-Index

- Not normalized
- Better when higher
- With balanced clusters, the CH index is generally a good criterion to indicate the correct number of clusters.

The notion of stability

Stability is a recent concept used to assess the quality of a clustering partition.

Stability: Definition

A cluster, or a partition are said to be stable if they are insensitive to small changes in the underlying data.

- Stability is usually tested by bootstrapping the same clustering algorithm many times over several sub-samples of the data.

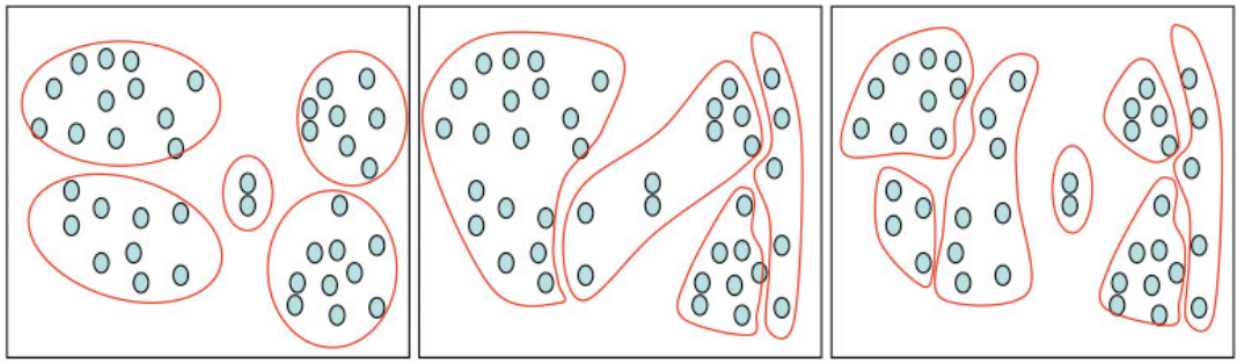


Figure: Example of an unstable partition

The notion of stability

Given the results of R simulations $C_1, \cdots, C_R$, we have:

$$S = \frac{1}{(R-1)^2} \sum_{i=1}^{R} \sum_{j \neq i}^{R} (1 - \Delta(C_i, C_j))$$

Pros

- Very intuitive criterion
- Assesses the quality of a partition without any bias

Cons

- Time consuming to compute: Requires bootstrapping
- Requires the use of an operator to assess the difference between 2 partitions (Δ), or of an external criterion.

Using External indexes to asses cluster quality

Quite often clustering algorithms will be tested in a non-exploratory setting:

- A data set for which the real classes are known will be used for test purposes (with the added advantage that the number of clusters will be known)
- In this case, the cluster found will be compared to the real classes using external indexes or purity measures.

Cluster purity

Given K real classes $Y^1, \cdots, Y^K$:

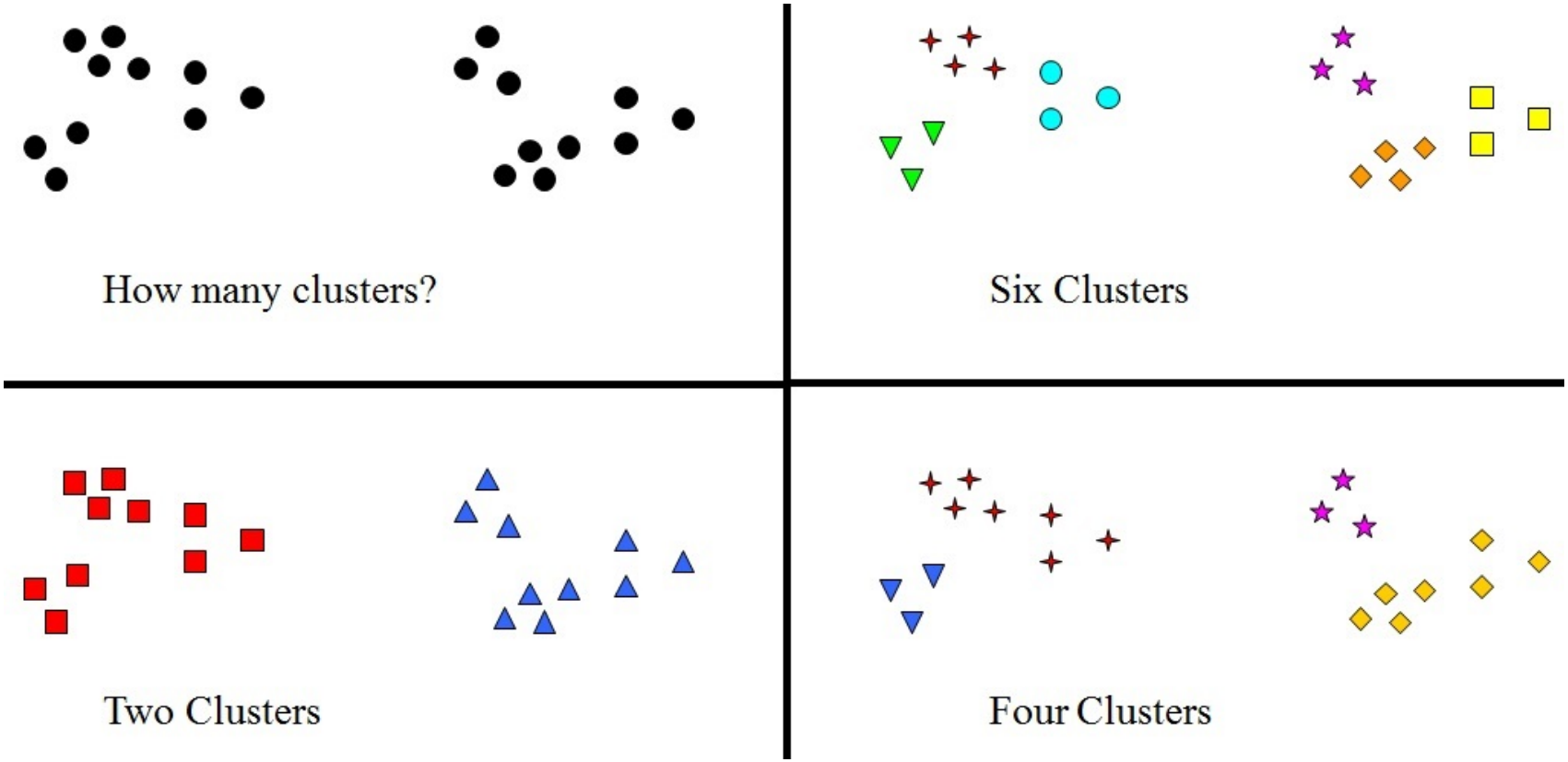
- Purity of a cluster C^i: $CP(C^i) = \frac{1}{|C^i|} argmax_j |C^i \cap Y^j|$
- Purity of a partition: $CP = \frac{1}{N} \sum_{i=1}^{k} argmax_j |C^i \cap Y^j|$

Examples of external indexes

- Rand Index, Adjusted Rand Index, Accuracy, etc.

How many clusters ?

Guessing the right number of clusters is one of the oldest problem in clustering, and several methods require it as a parameter.



Criteria to pick the right number of clusters

Some criteria exist to pick the optimal number of clusters, but they require to run the algorithm several times with different number of clusters in order to evaluate the solutions:

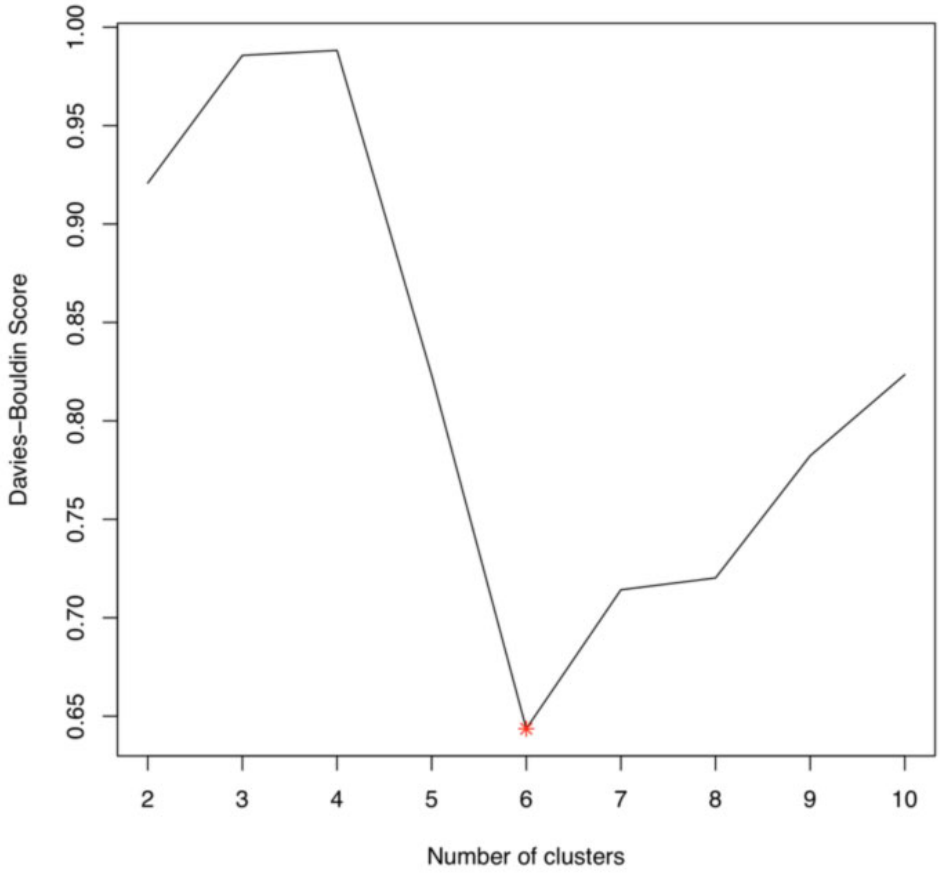
- The **Bayes Information Criterion** (BIC) and **Akaike Information Criterion** (AIC) work well with probabilistic algorithms.
- The **Minimum Description Length** criterion (MDL) and the **Minimum Message Length** criterion (MML) are recommended for hierarchical clustering.
- Stability can also be used to guess the right number of clusters.

Remark

Most of the time, the best solution is to ask the advice of an expert in the field of the studied data.

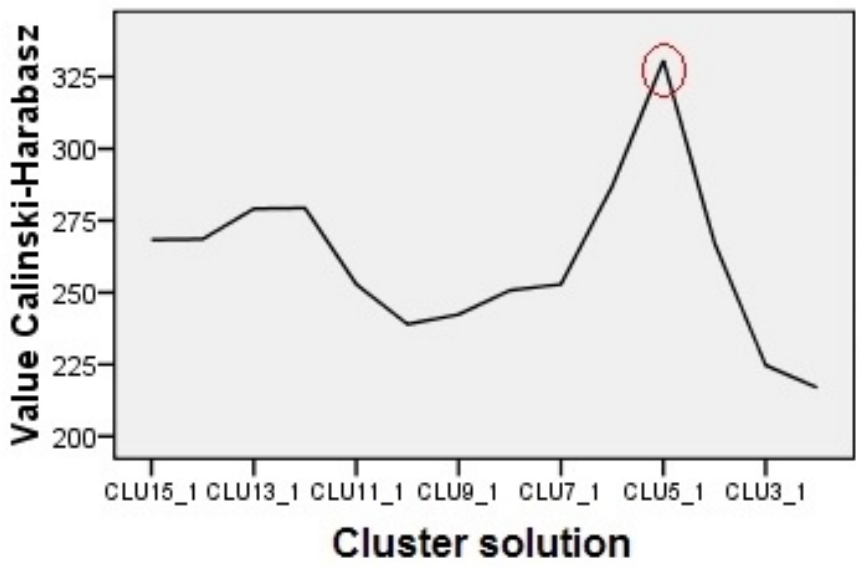
Criteria to pick the right number of clusters

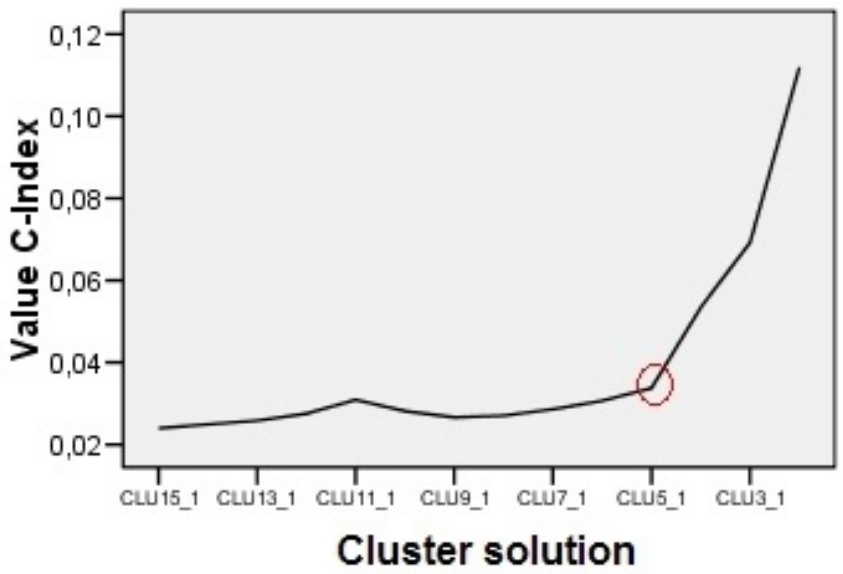
Sometimes clustering indexes can be used to help picking the right number of clusters:



Criteria to pick the right number of clusters

Another example with the C-Index and the Calinski-Harabasz indexes:





Picking a clustering algorithm

Picking the right clustering algorithm depends on several factors and requires prior knowledge on the data. With high dimensional data sets, you usually don't have this knowledge:

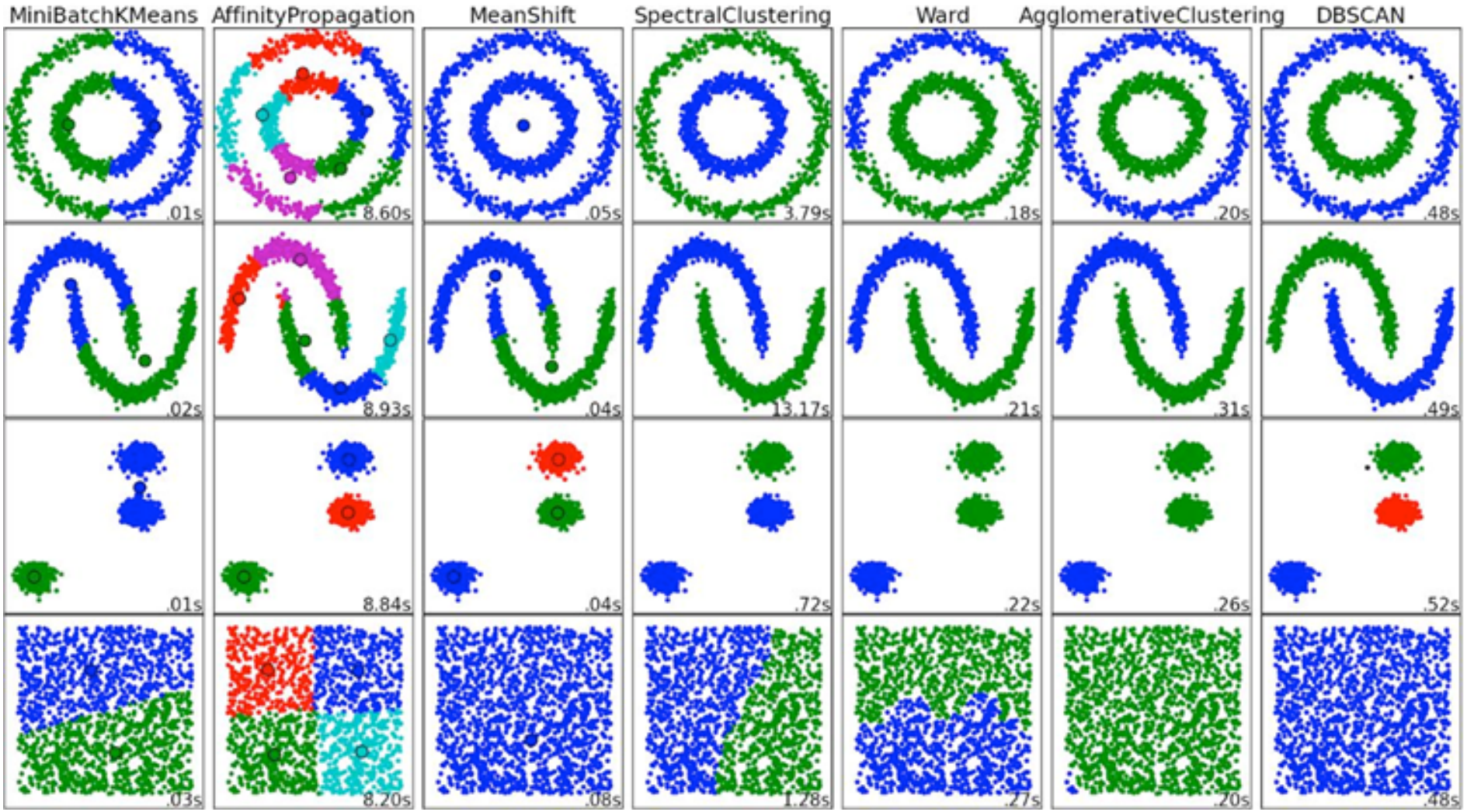
- Are there any clusters ?
- What is the shape of the clusters and how well separated are they ?

Other considerations may include the complexity of your algorithm:

- Complexity depending on the size of the data set.
- Complexity depending on the number of attributes of the data set.

Picking an algorithm often ends up being a trade-off between the expected quality of the result and the computational complexity of your algorithm.

Picking a clustering algorithm



Summary of open problems in clustering

- Defining the notion of similarity, and picking or defining a distance function accordingly.
- Picking the right algorithm and the right parameters for this algorithm.
- Figuring out how many clusters to search for.
- Evaluating clustering results.
- Comparing clustering results.
- All known data analysis problems: normalizing the data or not, picking relevant attributes, removing the outliers or not, etc.

Outline

Bibliography

- Christopher M. Bishop, Pattern Recognition and Machine Learning (2006)
- Brian S. Everitt et al., Cluster Analysis 5th edition (2011)