Unsupervised Learning: clustering algorithms

Slim Essid

Télécom ParisTech

slim.essid@telecom-paristech.fr

Largely based on slides by Florence d'Alché-Buc and Alexandre Gramfort

Learning from unlabeled data

Unlabeled data

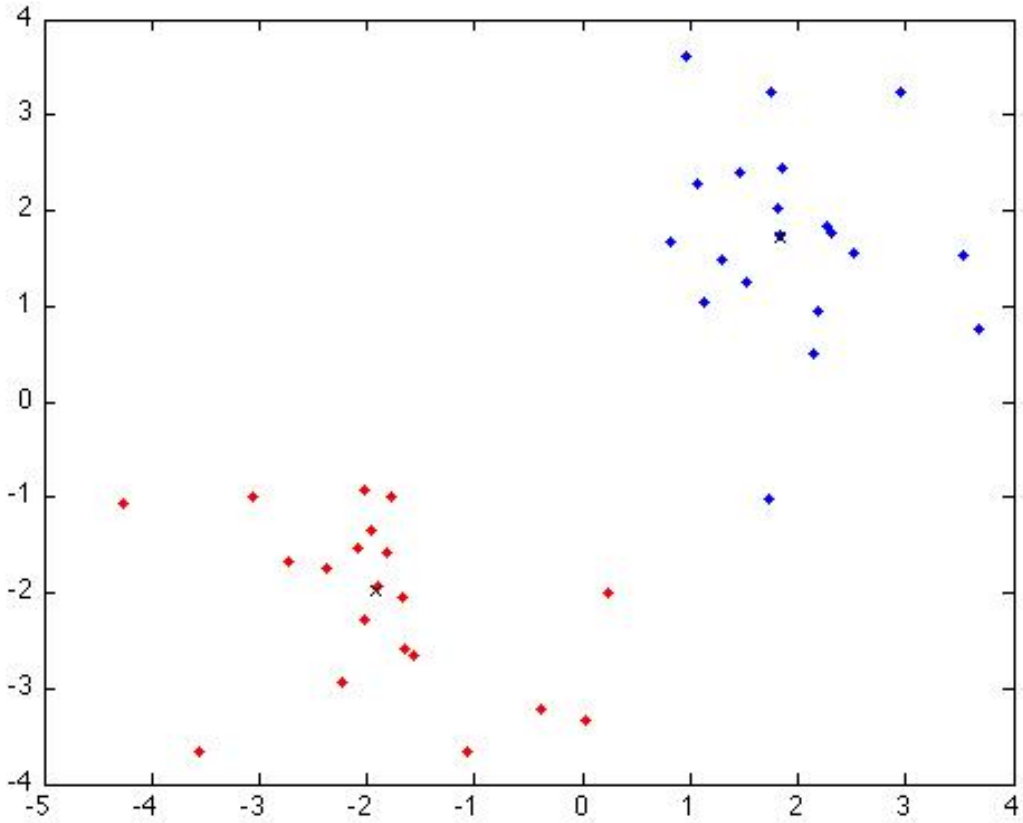
- Available data are unlabeled : documents, webpages, clients database...
- Labeling data is expensive and requires some expertise

Learning from unlabeled data

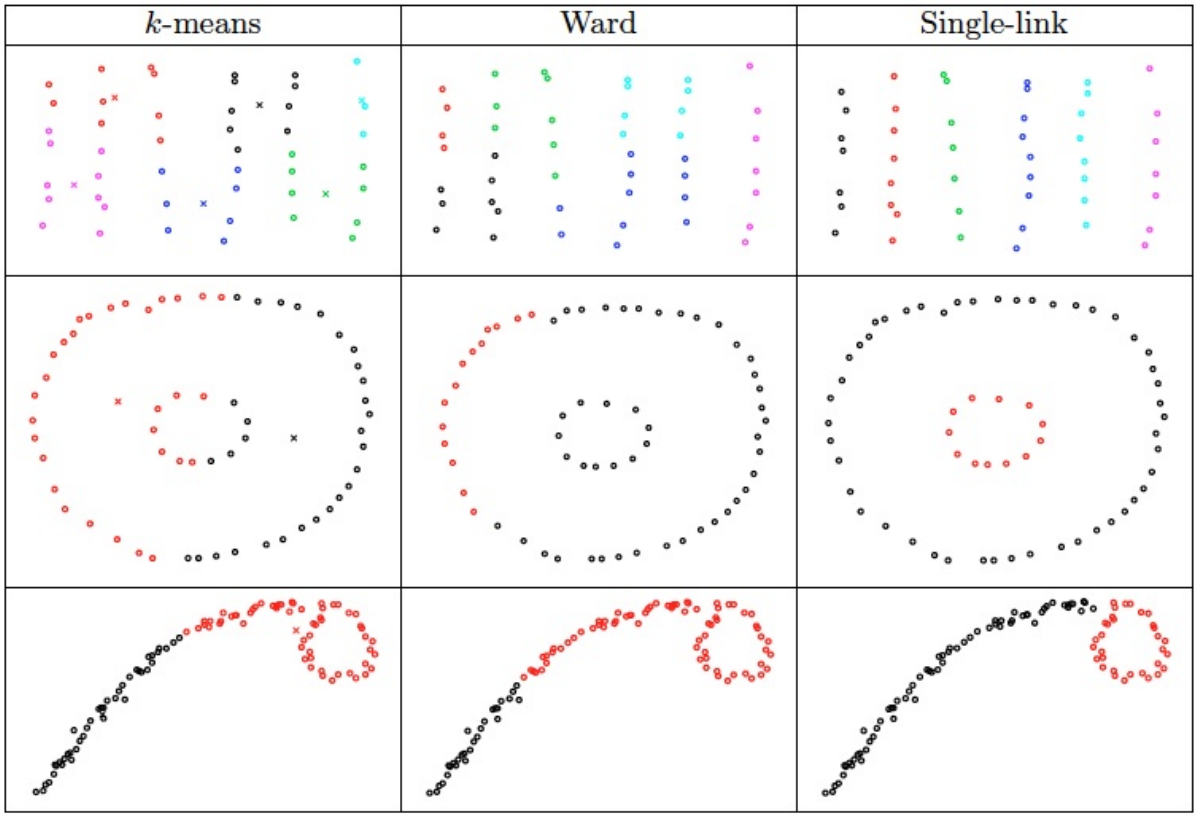
- Modeling probability distribution $\rightarrow$ graphical models
- Dimensionality reduction $\rightarrow$ pre-processing for pattern recognition
- **Clustering** : group data into homogeneous clusters $\rightarrow$ organize your data, make easier access to them, pre and post processing

What is clustering ?

Here is a clustering in 2 clusters



Different clusterings



Clustering for image segmentation

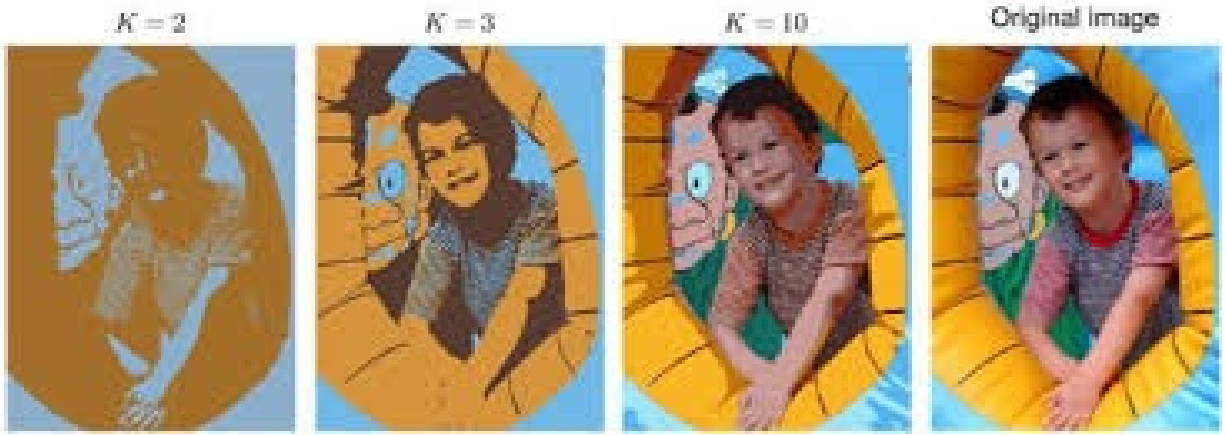


Image from C. Bishop's book, Pattern recognition and Machine Learning, Springer

Clustering algorithms : a data-analysis point of view

Definitions

- **Dissimilarity :** $d(x_i, x_j)$, a distance (without the triangle inequality)
- **Between-class dispersion :** for a given K-clustering $\mathcal{C}$:
  $B(\mathcal{C}) = \frac{1}{2} \sum_k \sum_{i,j,C(i)=k,C(j)\neq k} d(x_i, x_j)$
- **Within-class dispersion :**
  $W(\mathcal{C}) = \frac{1}{2} \sum_k \sum_{i,j,C(i)=k,C(j)=k} d(x_i, x_j)$
- **Total dispersion :**
  $T(x_1, \ldots, x_n) = \frac{1}{2} \sum_{i,j} d(x_i, x_j)$

NB :

$T = B(\mathcal{C}) + W(\mathcal{C})$, for all $\mathcal{C}$

Clustering algorithms

Definition : a data-analysis point of view

Given a set of data $\mathcal{S} = \{x_1, x_2, \ldots, x_n\}$, a chosen K and a dissimilarity d, one seeks a K-partition of $\mathcal{S}$, such that the between-class dispersion (inertia) is the largest and/or the within-class dispersion is the smallest.

Outline

The K-means algorithm : an example of vector quantization model

Given a set of vectors $x_1, x_2, \ldots, x_n$, the K-means algorithm seeks a partition of this set into K clusters $C_1, C_2, \ldots, C_k$ that minimizes the following loss function :

$$R(\{C\}_{k=1}^{K}) = \sum_{k=1}^{K} \sum_{x_i \in C_k} ||x_i - \mu_k||^2, \quad (1)$$

where $\mu_k = \frac{\sum_{x_i \in C_k} x_i}{|C_k|}$

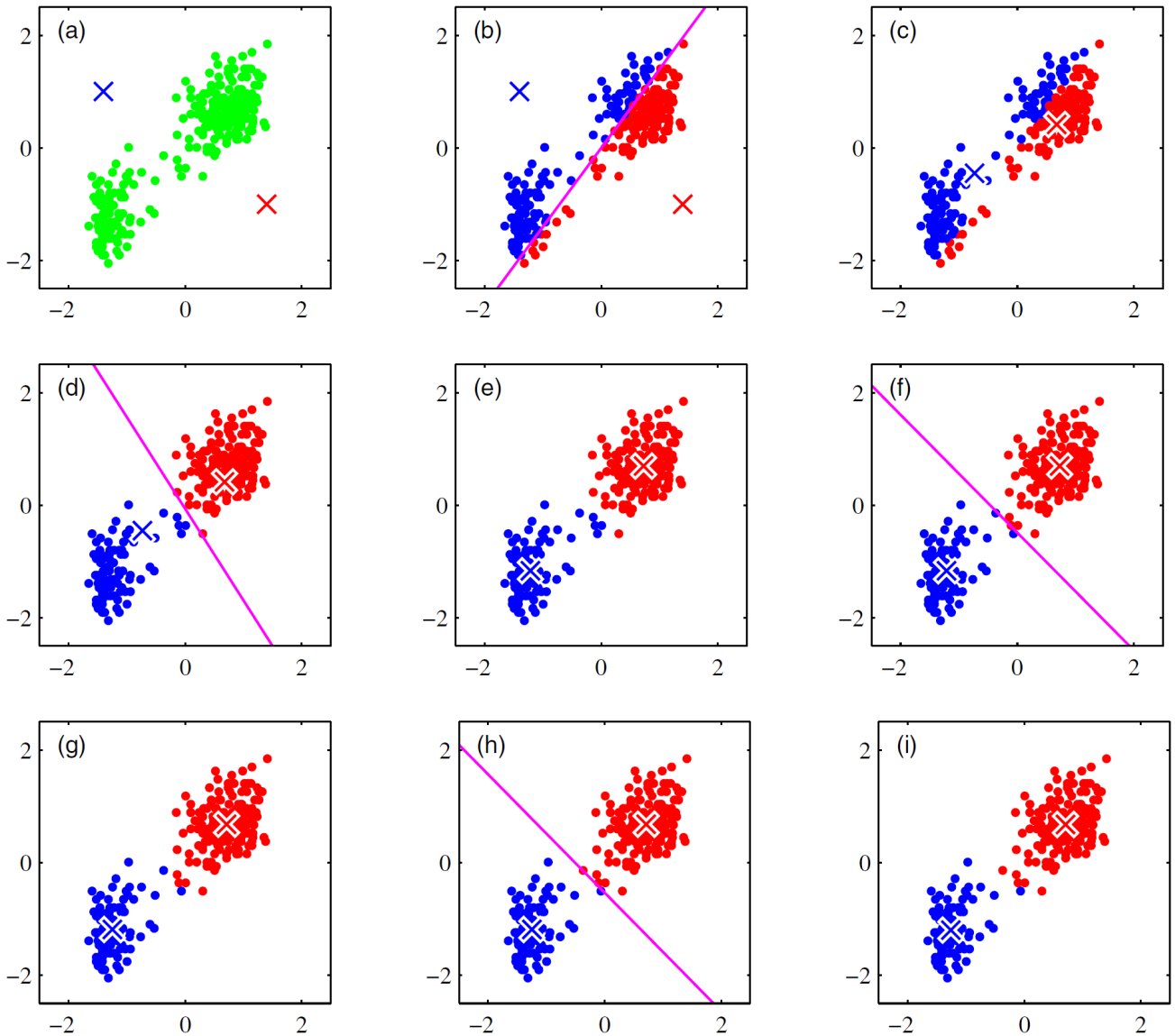
$|C_k|$: cardinal of C_k

The K-means algorithm

1. **Initialization** ($t = 0$) : initialization of the μ_k with K randomly chosen observations
2. **Assignment step** : assign each observation to the cluster whose mean yields the least within-cluster quantization error :
   - $C_k^{(t)} = \{x_m, ||x_m - \mu_k^{(t)}|| \leq ||x_m - \mu_j^{(t)}||, \forall j, 1 \leq j \leq K\}$
3. **Update step** : compute the new means
   - $t \leftarrow t + 1$
   - $\mu_k^{(t)} = \frac{1}{|C_k^{(t)}|} \sum_{x_j \in C_k^{(t)}} x_j$
4. **Stopping criterion** : Stop when the assignments no longer change

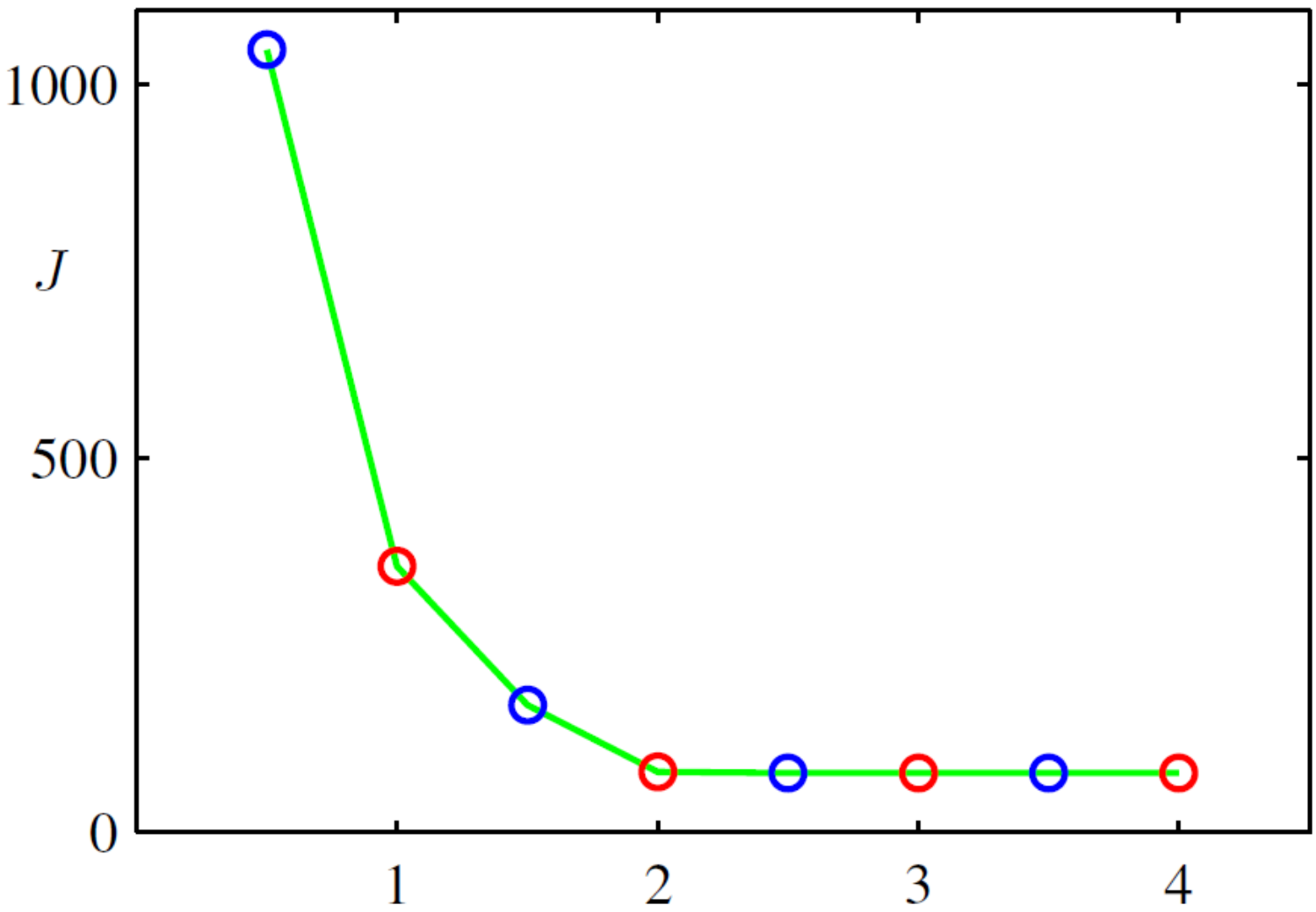
The K-means algorithm

After Bishop, 2006



The K-means algorithm

After Bishop, 2006



Remarks

- The K-means algorithm converges monotonically : each iteration of the algorithm does not increase the K-means objective function.
- There is no guarantee on the number of iterations the k-means algorithm needs in order to reach convergence.
- There is no nontrivial lower bound on the gap between the value of the K-means objective of the algorithm output and the minimum possible value of that objective function.
- K-means might converge to a point which is not even a local minimum !
- To improve the results of K-means it is recommended to **repeat the procedure several times with different randomly chosen initial centroids**.

The K-medoids objective function

Similar to the K-means objective, except that a more general dissimilarity $\mathcal{V}(x, \mu_i)$ is considered and the cluster centroids are required to be members of the input set :

$$G_{\text{K-medoids}}((\mathcal{X}, d), (C_1, \ldots, C_K)) = \min_{\mu_1, \ldots \mu_k \in \mathcal{X}} \sum_{i=1}^{K} \sum_{x \in C_i} \mathcal{V}(x, \mu_i)$$

The K-median objective function

Similar to the K-medoids objective, except that the "distortion" between a data point and the centroid of its cluster is measured by distance, rather than by the square of the distance :

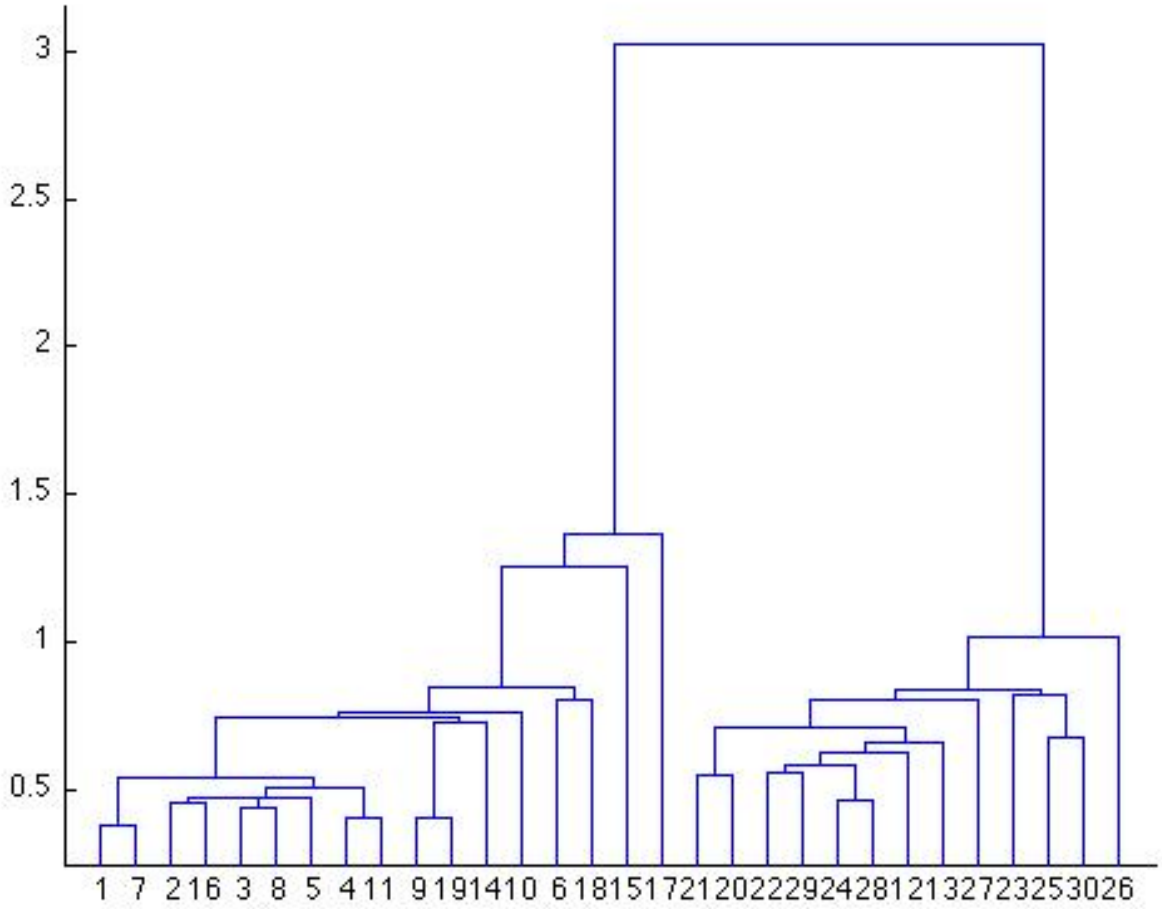
$$G_{\text{K-median}}((\mathcal{X}, d), (C_1, \ldots, C_K)) = \min_{\mu_1, \ldots \mu_k \in \mathcal{X}} \sum_{i=1}^{K} \sum_{x \in C_i} d(x, \mu_i)$$

An example is the facility location problem. Consider the task of locating K fire stations in a city. One can model houses as data points and aim to place the stations so as to minimize the average distance between a house and its closest fire station.

Outline

Principle of Hierarchical clustering

Goal build a dendrogram

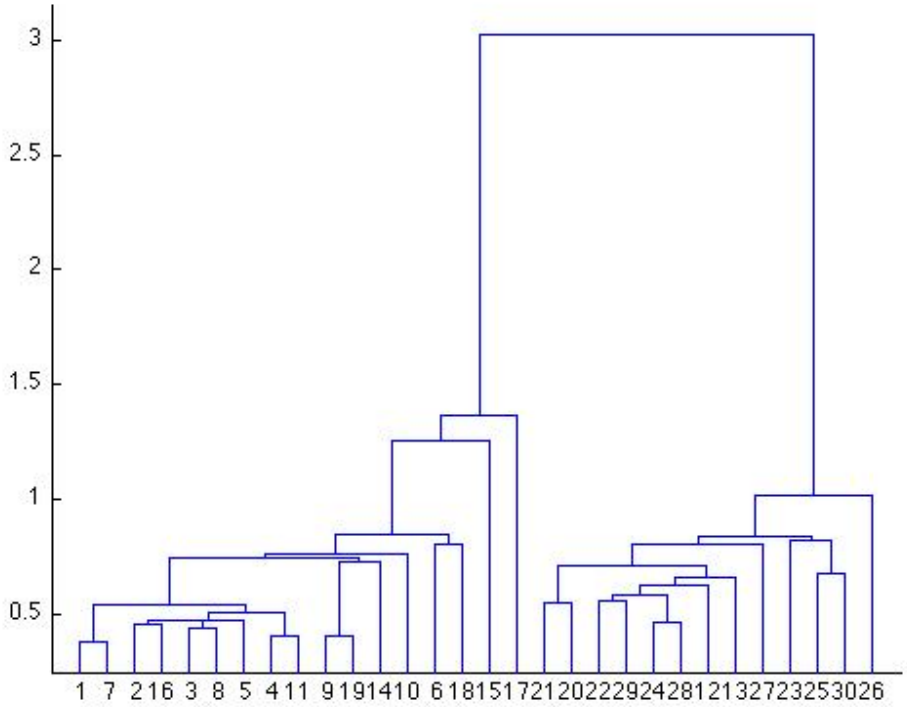


Hierarchical Agglomerative clustering

Building a dendrogram

1. Singletons containing a single data are initial clusters
2. nb = n
3. Build the distance matrix between the clusters
4. While (nb $>$ 1) do
   - The two closest clusters are joined using a node/branch whose length is equal to the distance between the two clusters
   - The two clusters are removed and nb = nb-1 ;
   - The distance between the new cluster and all remaining ones are computed

Clustering from a dendrogram



- In order to obtain a clustering, the dendrogram is cut using some cutoff value
- As for K-means or Gaussian Mixture Models, finding the right cutoff is a difficult issue

Distance D between two clusters A and B

Common choices :

- **Single linkage** : $D(A,B) = min_{x \in A, y \in B} d(x,y)$

$\rightarrow$ *favours connectivity*

- **Complete linkage** : $D(A,B) = max_{x \in A, y \in B} d(x,y)$

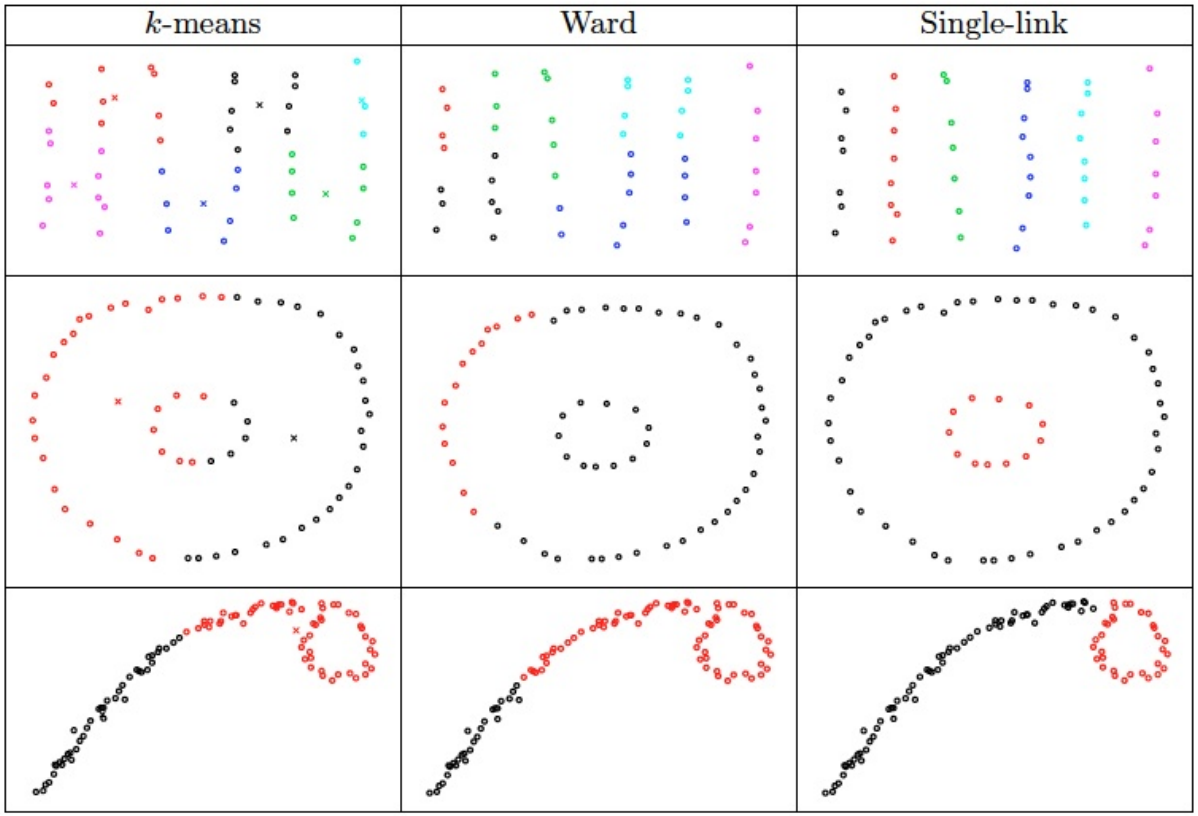
$\rightarrow$ *favours compactness*

- **Ward's method** : $D(A,B) = \frac{n_A n_B}{n_A + n_B} d(m_A, m_B)$

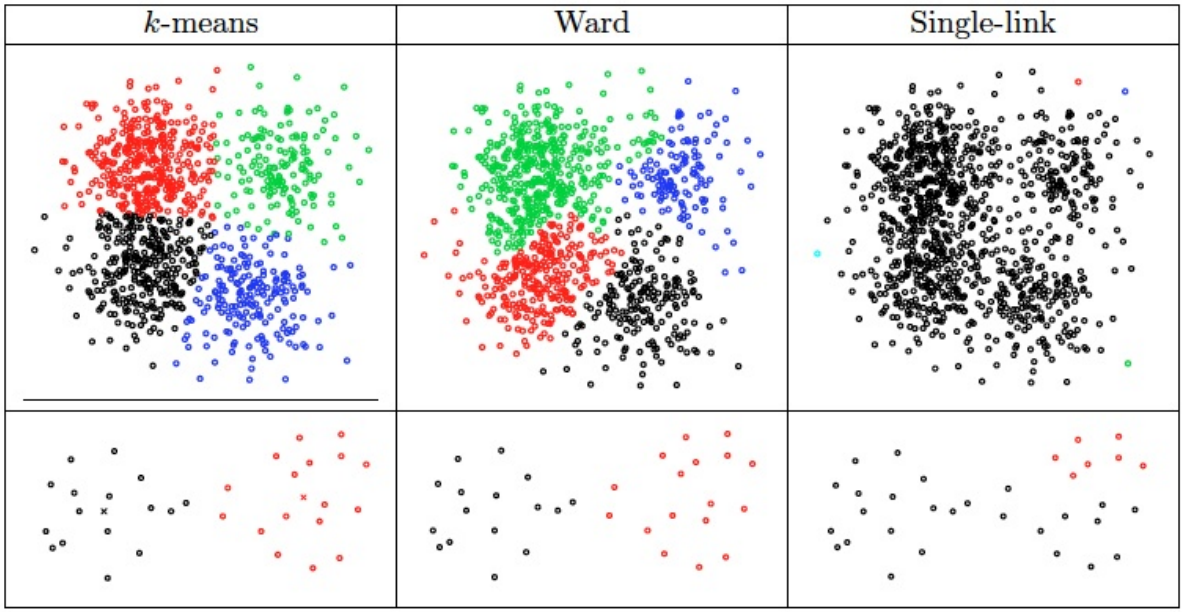
  m_A (resp. m_B) : center of gravity of A (resp. B)

$\rightarrow$ *minimises the total within-cluster dispersion*

Examples 1



Examples 2



Outline

DBSCAN

- "Density-based spatial clustering of applications with noise" (DBSCAN) is a very popular, simple and powerful algorithm first proposed by Ester et al. 1996.
- DBSCAN is one of the most common clustering algorithms and also most cited in scientific literature.
- In 2014, it was awared the test of time award at the leading data mining conference, KDD.

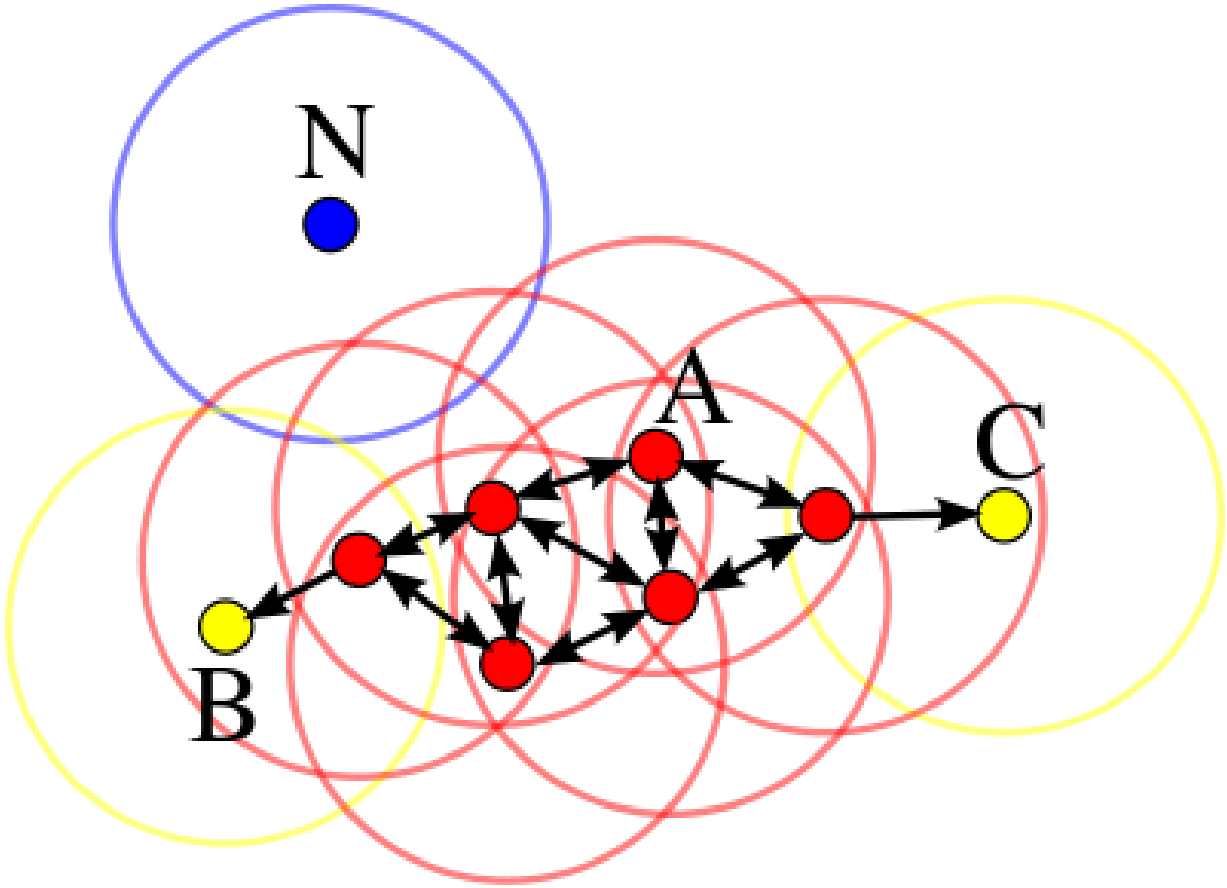
DBSCAN Algorithm

- 2 parameters : ϵ and the minimum number of points required to form a dense region q.
- Start with an arbitrary starting point not yet visited. Retrieve its ϵ-neighborhood. If it contains sufficiently many points, a cluster is started. Otherwise, the point is labeled as noise. [1]
- If a point is found to be a dense part of a cluster, its ϵ-neighborhood is also part of that cluster. All points that are found within the ϵ-neighborhood are added, so is their own ϵ-neighborhood when they are also dense.
- Process continues until the density-connected cluster is completely found.
- Start again with a new point, until all points have been visited.

1. A point marked as noise might later be found in a sufficiently sized ϵ-environment of a different point and hence be made part of a cluster.

DBSCAN Illustration

With q=4 in 2D :



Red : core points, Yellow : non core but in cluster, Blue : noise

Source : https ://en.wikipedia.org/wiki/DBSCAN

Algorithm 1 DBSCAN

procedure $\mathrm{DBSCAN}(\mathcal{X}, \epsilon, q)$
Initialize : $C = 0$.
    for each point x in $\mathcal{X}$ **do**
        if x is visited **then**
            continue to next point.
        end if
        mark x as visited.
        neighbors = getNeighbors(x, ϵ)
        if —neighbors— $< q$ **then**
            mark x as noise.
        else
            C = next cluster
            expandCluster(x, neighbors, C, ϵ, q)
        end if
    end for
Output : All produced clusters.
end procedure

```
1: procedure expandCluster(x, neighbors, C, ϵ, q)
2:     add x to C
3:     for each y in neighbors do
4:         if y is not visited then
5:             mark y as visited
6:             neighbors_y = regionQuery(y, ϵ)
7:             if —neighbors_y— ≥ q then
8:                 neighbors = neighbors joined with neighbors_y
9:             end if
10:        end if
11:        if y is not yet member of any cluster then
12:            add y to cluster C
13:        end if
14:    end for
15: end procedure
16: procedure regionQuery(x, ϵ)
17:    Output : all points within x's ϵ-neighborhood (including x)
18: end procedure
```

DBSCAN Pros

- No need to specify the number of clusters in the data a priori, as opposed to k-means.
- It can find arbitrarily shaped clusters. It can even find a cluster completely surrounded by (but not connected to) a different cluster.
- Due to the q parameter, the so-called single-link effect (different clusters being connected by a thin line of points) is reduced.
- It has a notion of noise, and is robust to outliers.

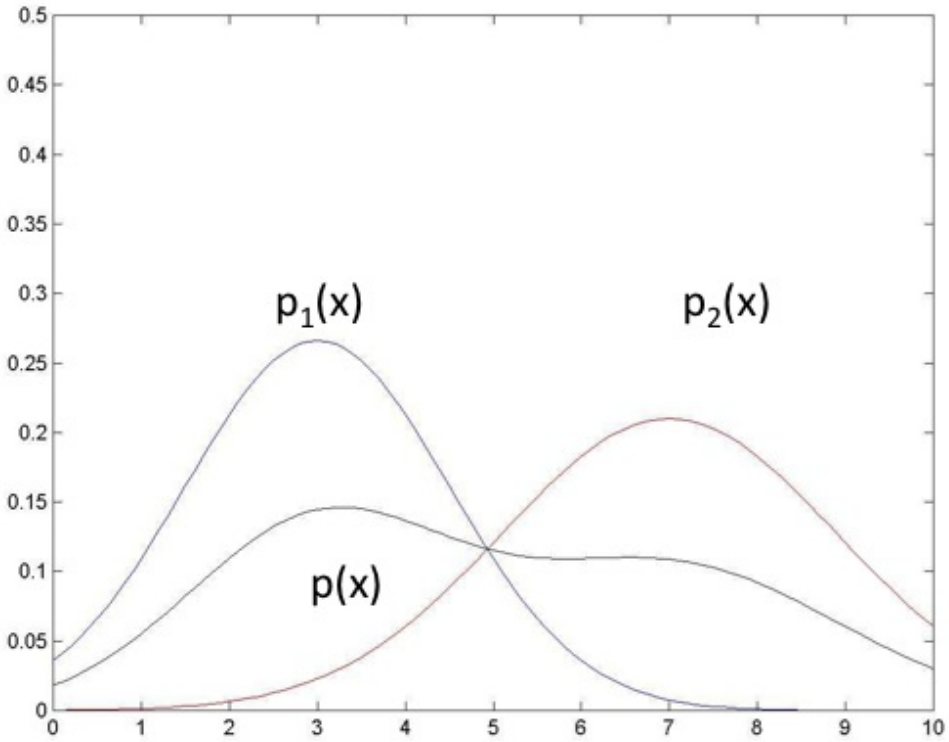
DBSCAN Cons

- It is not entirely deterministic (output depends on the order of the points).
- It still needs to specify a distance measure (like k-means or spectral clustering).
- It can not cluster data sets with a large difference in densities as the $q - \epsilon$ combination cannot then be chosen appropriately for all clusters.

Outline

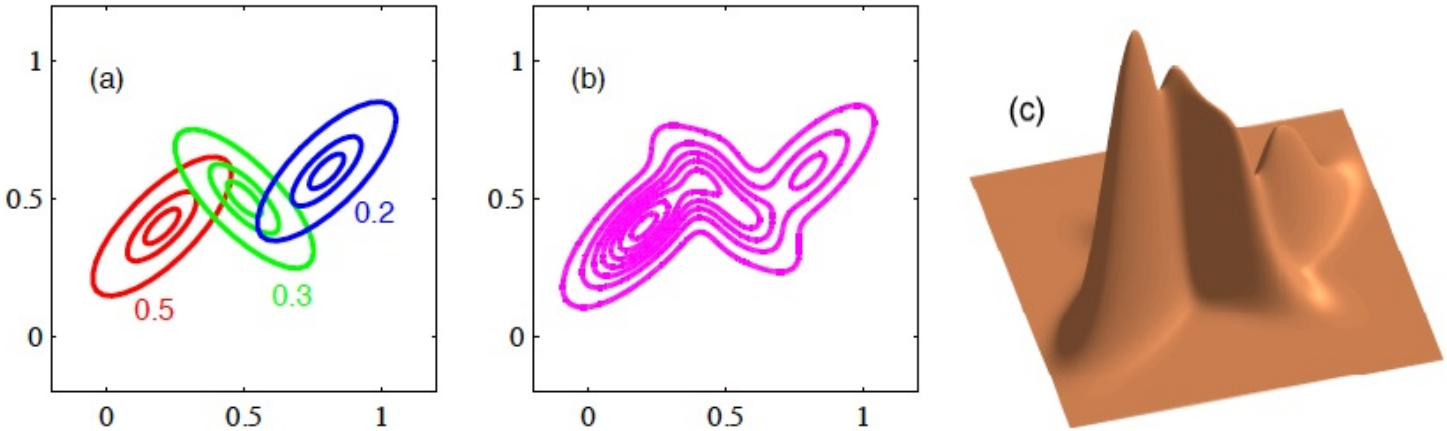
Clustering by modelling the data distribution

- Assume $x_1, \ldots, x_n$ is an i.i.d sample of n data points
- Model the data distribution by a Gaussian Mixture Model
- Each data point is to be associated with the component that best explains it



Clustering by modelling the data distribution

- Assume $x_1, \ldots, x_n$ is an i.i.d sample of n data points
- Model the data distribution by a Gaussian Mixture Model
- Each data point is to be associated with the component that best explains it



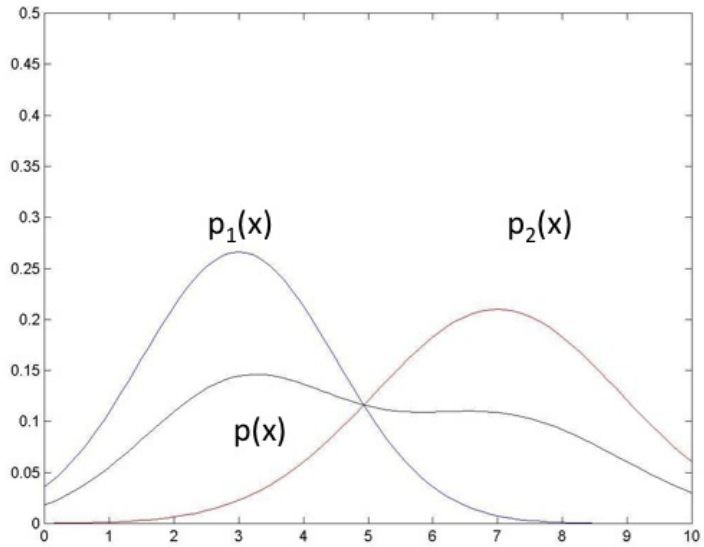
The Gaussian mixture model (GMM)

A parametric model :

$$p(x) = \sum_{k=1}^{K} \pi_k p(x|\mu_k, \Sigma_k)$$

where

- $p(x|\mu_k, \Sigma_k) = \mathcal{N}(x|\mu_k, \Sigma_k)$
- $\sum_{k=1}^{K} \pi_k = 1,\ 0 \leq \pi_k \leq 1.$



GMM formulation using latent variables

Let's introduce the K-dimensional indicator variable $\mathbf{z} = [z_k]_{1 \leq k \leq K}$, such that $z_k \in \{0, 1\}$, and $\sum_k z_k = 1$. Hence

- $p(z_k = 1) = \pi_k$ and $p(\mathbf{z}) = \prod_{k=1}^{K} \pi_k^{z_k}$
- $p(x|z_k = 1) = \mathcal{N}(x|\mu_k, \Sigma_k)$ and $p(x|\mathbf{z}) = \prod_{k=1}^{K} \mathcal{N}(x|\mu_k, \Sigma_k)^{z_k}$

The marginal distribution is obtained by summing over all states of $\mathbf{z}$:

$$p(x) = \sum_{\mathbf{z}} p(x, \mathbf{z}) = \sum_{\mathbf{z}} p(\mathbf{z}) p(x|\mathbf{z})$$

$$= \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k)$$

GMM formulation using latent variables

Let's introduce the K-dimensional indicator variable $\mathbf{z} = [z_k]_{1 \leq k \leq K}$, such that $z_k \in \{0, 1\}$, and $\sum_k z_k = 1$. Hence

- $p(z_k = 1) = \pi_k$ and $p(\mathbf{z}) = \prod_{k=1}^{K} \pi_k^{z_k}$
- $p(x|z_k = 1) = \mathcal{N}(x|\mu_k, \Sigma_k)$ and $p(x|\mathbf{z}) = \prod_{k=1}^{K} \mathcal{N}(x|\mu_k, \Sigma_k)^{z_k}$

The marginal distribution is obtained by summing over all states of $\mathbf{z}$:

$$\begin{aligned} p(x) = \sum_{\mathbf{z}} p(x, \mathbf{z}) &= \sum_{\mathbf{z}} p(\mathbf{z}) p(x|\mathbf{z}) \\ &= \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k) \end{aligned}$$

Cluster assignment and responsibilities

Assignment to a particular cluster C_k can be done based on :

$$\gamma(z_k) = p(z_k = 1|x) = \frac{p(z_k = 1)p(x|z_k = 1)}{\sum_{j=1}^{K} p(z_j = 1)p(x|z_j = 1)}$$
$$= \frac{\pi_k \mathcal{N}(x|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x|\mu_j, \Sigma_j)}$$

$\rightarrow \gamma(z_k)$ is the *responsibility* that component k takes for explaining x

Cluster assignment and responsibilities

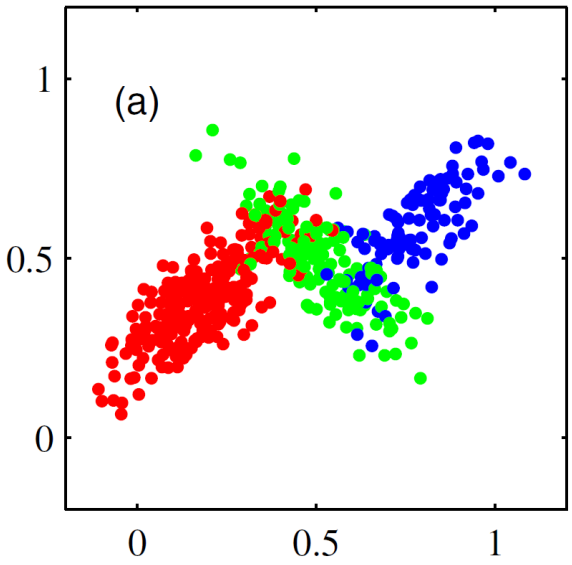
Assignment to a particular cluster C_k can be done based on :

$$\gamma(z_k) = p(z_k = 1|x) = \frac{p(z_k = 1)p(x|z_k = 1)}{\sum_{j=1}^{K} p(z_j = 1)p(x|z_j = 1)}$$
$$= \frac{\pi_k \mathcal{N}(x|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x|\mu_j, \Sigma_j)}$$

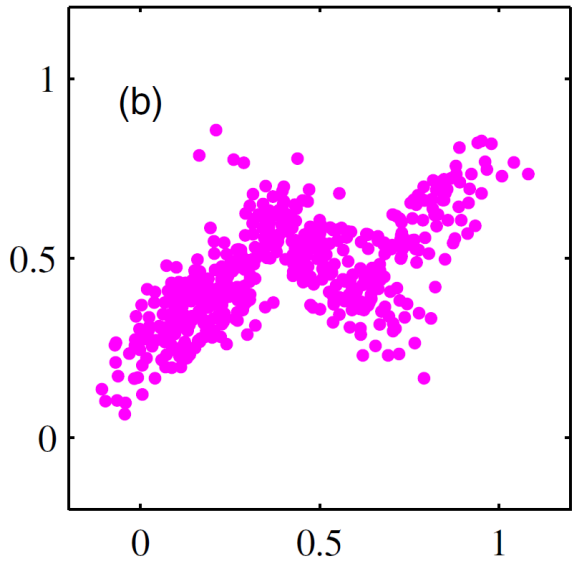
$\rightarrow \gamma(z_k)$ is the *responsibility* that component k takes for explaining x

Cluster assignment and responsibilities

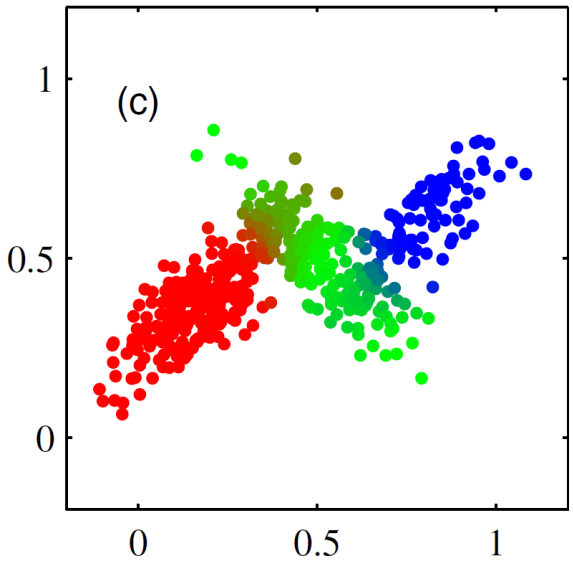
After Bishop, 2006



Original data



Unlabelled



Responsibilities

Mean and variance estimation in a 1D Gaussian distribution

We observe $x_1, \ldots, x_n$, n i.i.d samples from an unknown Gaussian distribution :

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

Maximum Likelihood Principle

- Likelihood : probability that data have been generated by the model
- Find μ and σ such that the likelihood $\ell(x_1, \ldots, x_n; \mu, \sigma) = \prod_{i=1}^{n} p(x_i|\mu,\sigma)$ be maximal

In practice, for exponential distributions, we maximize $\ln \ell(.;.)$.

Likelihood

$$\begin{aligned}
\mathcal{L}(x_1, \ldots, x_n; \mu, \sigma) &= \ln \prod_{i=1}^{n} p(x_i|\mu, \sigma) \\
&= \sum_{i=1}^{n} \ln p(x_i|\mu, \sigma) \\
&= -n \ln(\sqrt{2\pi}\sigma) - \frac{1}{2\sigma^2} \sum_{i=1}^{n} (x_i - \mu)^2
\end{aligned}$$

Maximum Likelihood Principle estimates for μ and σ

(Strict) convexity of $\mathcal{L}$ makes the problem easy to solve :

- To find μ : $\frac{\partial \mathcal{L}(\mu,\sigma)}{\partial \mu} = 0$

$\rightarrow$ We get : $\hat{\mu} = \frac{1}{n}\sum_i x_i$ (empirical mean)

- Then, to find σ, we use $\hat{\mu}$: $\frac{\partial \mathcal{L}(\hat{\mu},\sigma)}{\partial \sigma} = 0$

$\rightarrow$ We get : $\hat{\sigma} = \frac{1}{n}\sum_i (x_i - \hat{\mu})^2$ (empirical variance)

Multivariate Gaussian Distribution

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

Mean and covariance estimation by maximum likelihood estimation :

$$\begin{aligned}\hat{\mu} &= \frac{1}{n}\sum_{i=1}^{n} x_i \\ \hat{\Sigma} &= \frac{1}{n}\sum_{i=1}^{n}(x_i-\mu)(x_i-\mu)^T\end{aligned}$$

Gaussian Mixture Model estimation (general case)

Log likelihood to be maximized

$$\ln \prod_{i=1}^{n} p(x_i|\pi,\mu,\Sigma) = \sum_{i=1}^{n} \ln\{\sum_{k=1}^{K} \pi_k p(x_i|\mu_k,\Sigma_k)\}$$

Gaussian Mixture Model estimation (general case)

Log likelihood to be maximized

$$\ln \prod_{i=1}^{n} p(x_i|\pi,\mu,\Sigma) = \sum_{i=1}^{n} \ln\{\sum_{k=1}^{K} \pi_k p(x_i|\mu_k,\Sigma_k)\}$$

A difficult function to optimize

- the log is outside the sum
- the model is not identifiable : many latent settings have the same likelihood

Expectation-Maximization (EM) algorithm

- A general algorithm to solve estimation problems with incomplete data
- this algorithm is used in many other probabilistic models (not only GMM)

Refs : Demspter, Laird and Rubin1977 : more than 40000 citations
Good introductions : Bishop's book (2006), Kevin Murphy's course notes (2006), Bilmes's tutorial, (1998)

Key idea : exploit the responsibilities $\gamma(z_k)$

EM algorithm for GMM estimation

After Bishop, 2006

1. Initialise μ_k, Σ_k and π_k
2. **E-step :** evaluate the responsibilities using the current parameter values :

$$\gamma(z_{ik}) = \frac{\pi_k \mathcal{N}(x_i|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x_i|\mu_j, \Sigma_j)}$$

   where z_{ik} indicates if x_i comes from the k^{th} Gaussian

3. **M-step :** re-estimate the parameters using the current responsibilities :

$$\mu_k = \frac{1}{n_k} \sum_{i=1}^{n} \gamma(z_{ik}) x_i$$

$$\Sigma_k = \frac{1}{n_k} \sum_{i=1}^{n} \gamma(z_{ik})(x_i - \mu_k)(x_i - \mu_k)^T$$

$$\pi_k = \frac{n_k}{n}; \quad \text{where} \quad n_k = \sum_{i=1}^{n} \gamma(z_{ik})$$

EM algorithm for GMM estimation

After Bishop, 2006

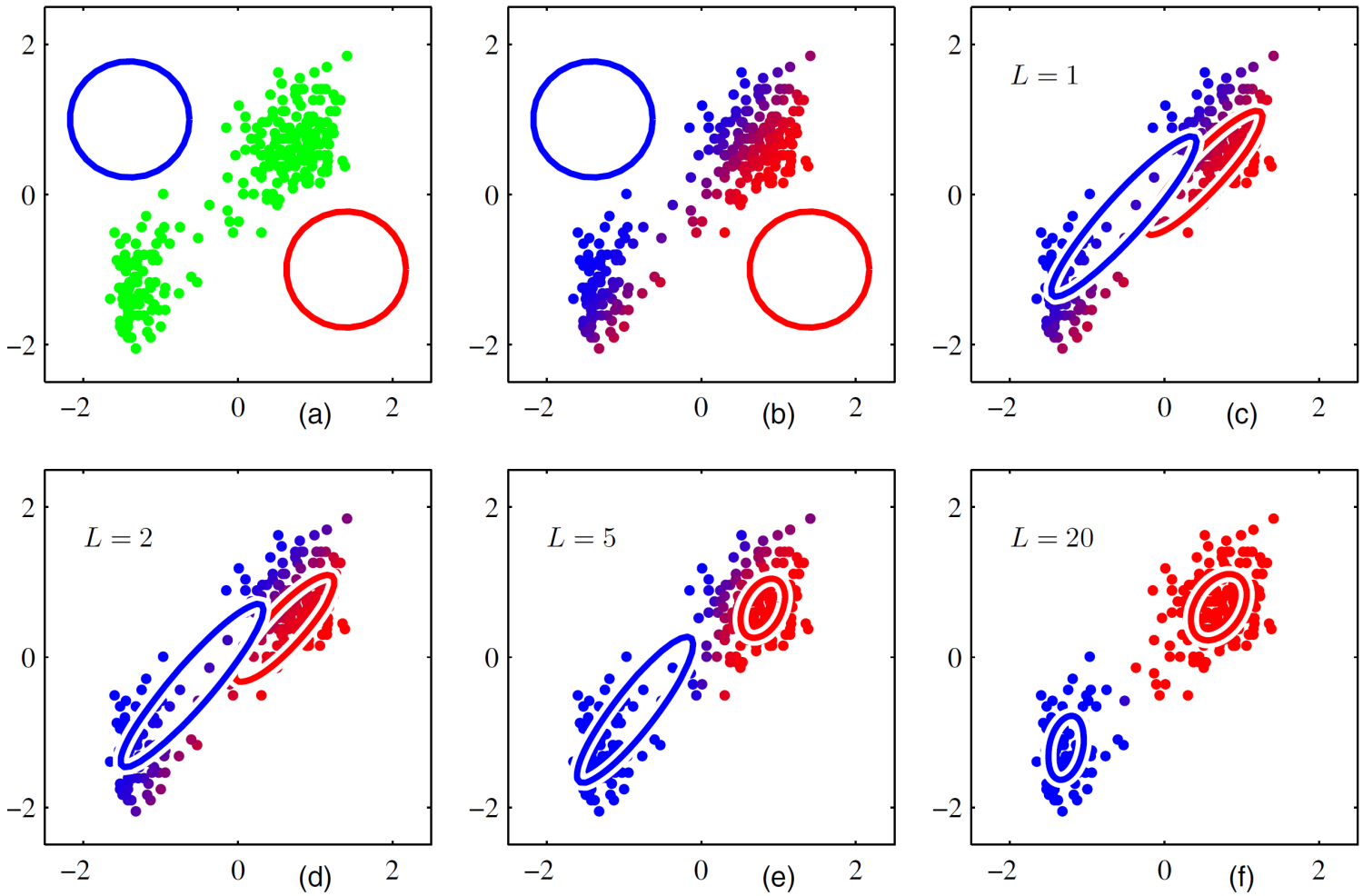
4. Evaluate the log likelihood

$$\sum_{i=1}^{n} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(x_i | \mu_k, \Sigma_k) \right\}$$

and check for convergence of either the parameters or the log likelihood. If no convergence, return to step 2.

EM algorithm for GMM estimation

After Bishop, 2006



Expectation maximization algorithm

- Local convergence only
- Need to restart the algorithm with different initial guesses
- K-means are a good way of initialising the algorithm

Outline

How to select K the number of clusters ?

Numerous criteria have been proposed with varying success in practise.

- Stability criterion (Ben-Hur and Elisseef, 2002)
- BIC criterion for GMM

Stability

A clustering algorithm is *stable* if when run twice on two close datasets it provides almost similar clusterings.

In practice, use bootstrap samples without replacement to measure stability.

Stability Algorithm

Let $\mathcal{S}$ be the dataset.

- $f = 0.8$
- for k=2 to k_{max} do
  - for b=1 to B do
    - $\mathcal{S}_1$ = subsample($\mathcal{S}$,f) : a subsample with a fraction f of data
    - $\mathcal{S}_2$ = subsample($\mathcal{S}$,f) : a subsample with a fraction f of data
    - C_1 = cluster($\mathcal{S}_1$,k)
    - C_2 = cluster($\mathcal{S}_2$,k)
    - intersect = $\mathcal{S}_1 \cap S_2$
    - S(b,k) = sim(C_1(intersect), C_2(intersect))
  - endfor
  - S(k) = mean(S(b,k))
- endfor

Model selection for GMM

How do we select the number of components ?

- A simple way is to use cross-validation to find the K valued that maximize the log likelihood.
- Alternatively, we can use the BIC (Bayesian information criterion) score

Model selection for GMM

BIC score :

$$BIC(\theta) = \log p(S|\hat{\theta}^{ML}) - \frac{d}{2} \log n,$$

where d is the dimensionality of the model and n the number of data points.

d, the dimensionality of the model, is here the number of estimated parameters : $(K-1)$ mixing probabilities, KP mean coefficients and $K\frac{P(P+1)}{2}$ covariance parameters.

References

- Video-lectures :
  - `http://videolectures.net/ecmlpkdd08_jain_dcyb/`
- Books
  - The Elements of Statistical Learning, Hastie, Tibshirani and Friedman, Springer. [chapitre 14]
  - Pattern Recognition and Machine Learning, C. Bishop, 2006, Springer