Unsupervised Learning: clustering algorithms

Florence d'Alché-Buc

Télécom ParisTech

florence.dalche@telecom-paristech.fr

Adapted and presented by Slim ESSID
slim.essid@telecom-paristech.fr

Outline

Learning from unlabeled data

Unlabeled data

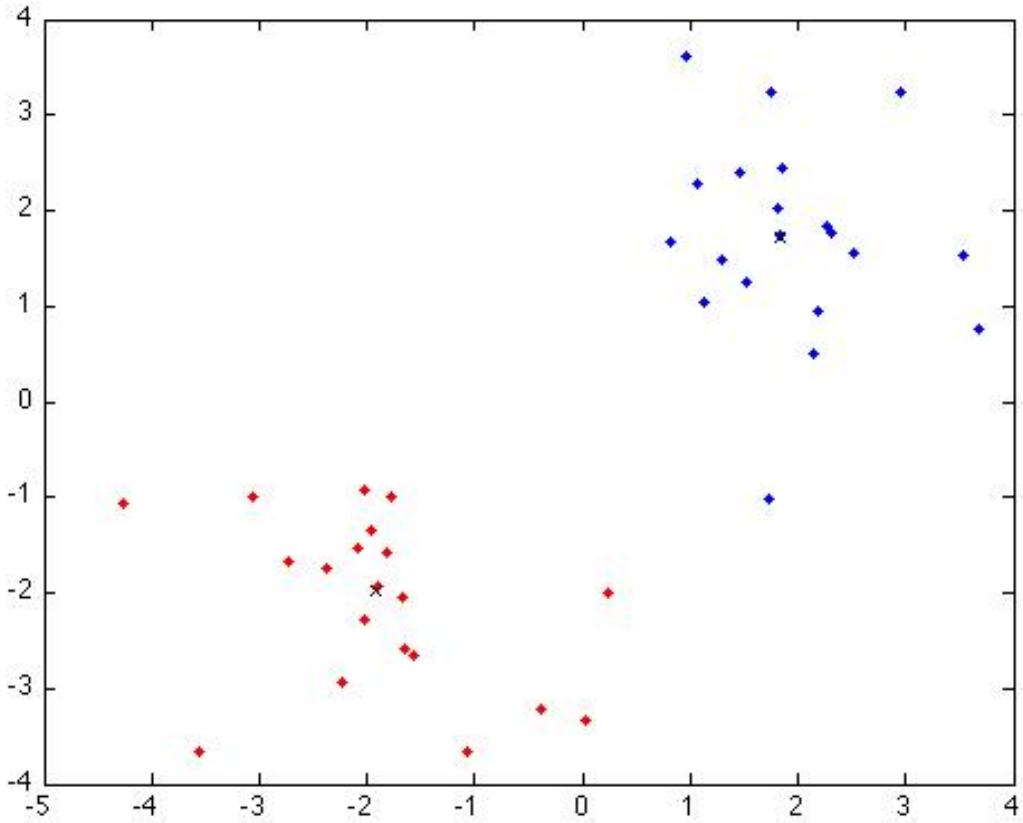
- Available data are unlabeled : documents, webpages, clients database ...
- Labeling data is expensive and requires some expertise

Learning from unlabeled data

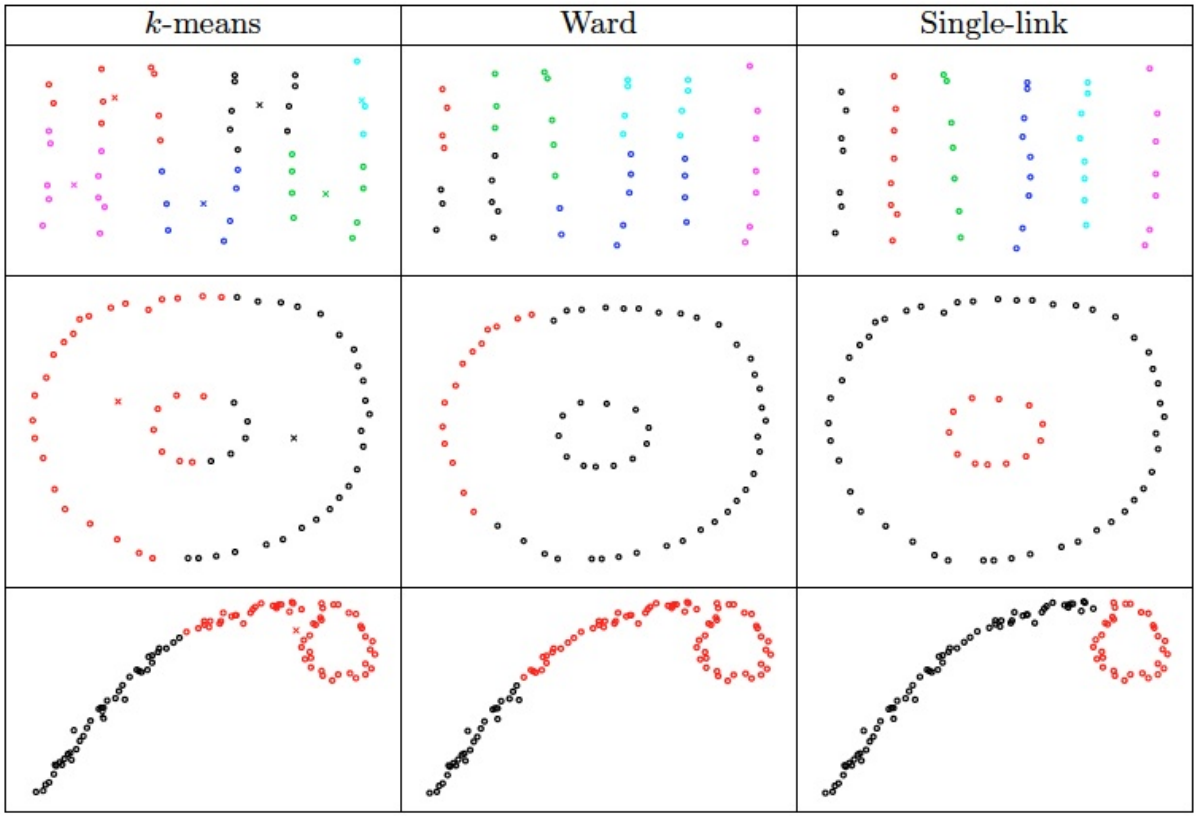
- Modeling probability distribution $\rightarrow$ graphical models
- Dimensionality reduction $\rightarrow$ pre-processing for pattern recognition
- **Clustering** : group data into homogeneous clusters $\rightarrow$ organize your data, make easier access to them, pre and post processing

What is clustering ?

Here is a clustering in 2 clusters



Different clusterings



Clustering for image segmentation

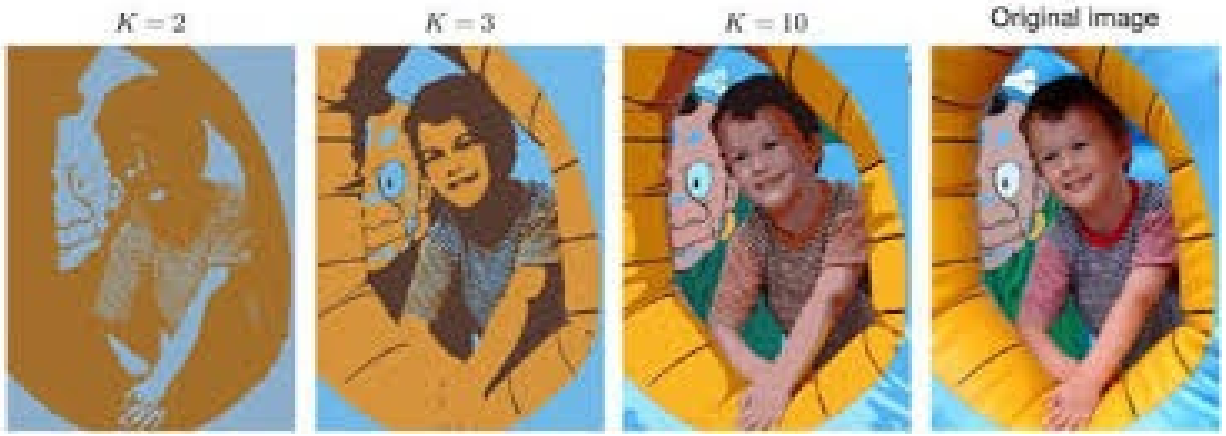


Image from C. Bishop's book, Pattern recognition and Machine Learning, Springer

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 $\mathcal{S} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, a chosen K and a dissimilarity d, you want to find 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 $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$, 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 :

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

where $\mu_k = \frac{\sum_{\mathbf{x}_i \in C_k} \mathbf{x}_i}{|C_k|}$

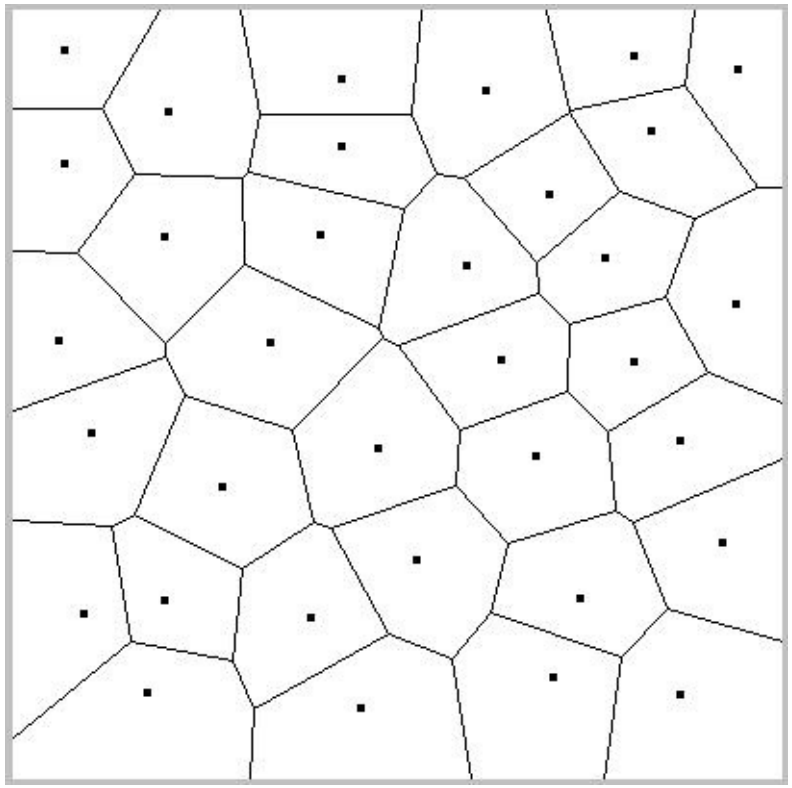
$|C_k|$: cardinal of C_k

The K-means algorithm

1. **Initialization** (t= 0) : initialization of the $\mu'_k s$ 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
   - $t \leftarrow t + 1$
   - $\mu_k^{(t)} = \frac{1}{|C_k^{(t)}|} \sum_{x_j \in C_k^{(t)}} \mathbf{x}_j$
4. **Stopping criterion** : Stop when the assignments no longer change

Convergence properties of the k-means algorithm

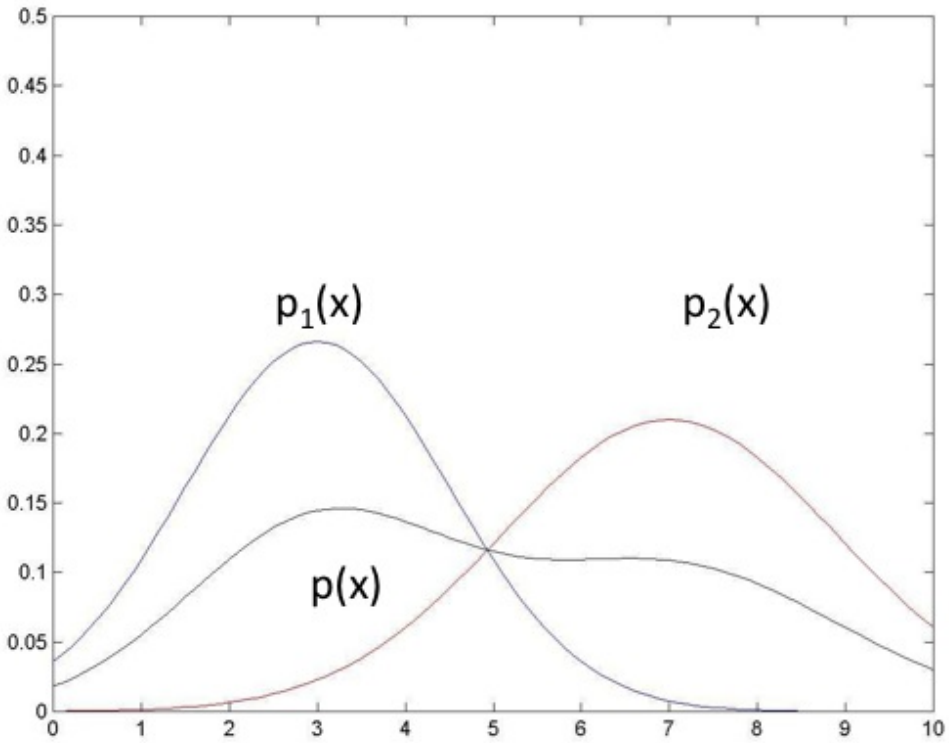
- The algorithm is ensured to converge towards a local minimum
- No guarantee for a global minimum
- The algorithm tends to build Voronoi cells



Outline

Clustering by modeling the data distribution

- Assume $x_1, \ldots, x_n$ is a n-length i.i.d sample
- Model the data distribution by a Gaussian Mixture Model
- For each data, compute the probability that the data comes from a given component

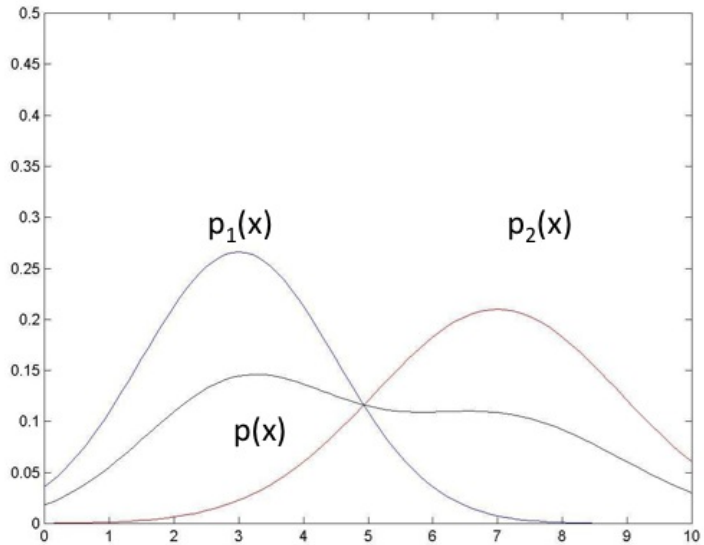


Gaussian mixture model

A parametric model :

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

where $\sum_{k=1}^{K} \pi_k = 1$, $0 \leq \pi_k \leq 1$.



We denote $\theta = \{\pi, \mu, \Sigma\}$.

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 $\mathcal{L}$
- To find μ : $\frac{\partial \mathcal{L}(\mu,\sigma)}{\partial \mu} = 0$
- We get : $\hat{\mu} = \frac{1}{n}\sum_i x_i$ (empirical mean)
- Then, to find σ, we use $\hat{\mu}$: $\frac{\partial \mathcal{L}(\hat{mu},\sigma)}{\partial \sigma} = 0$
- We get : $\hat{\sigma} = \frac{1}{n}\sum_i (x_i - \hat{\mu})^2$ (empirical variance)

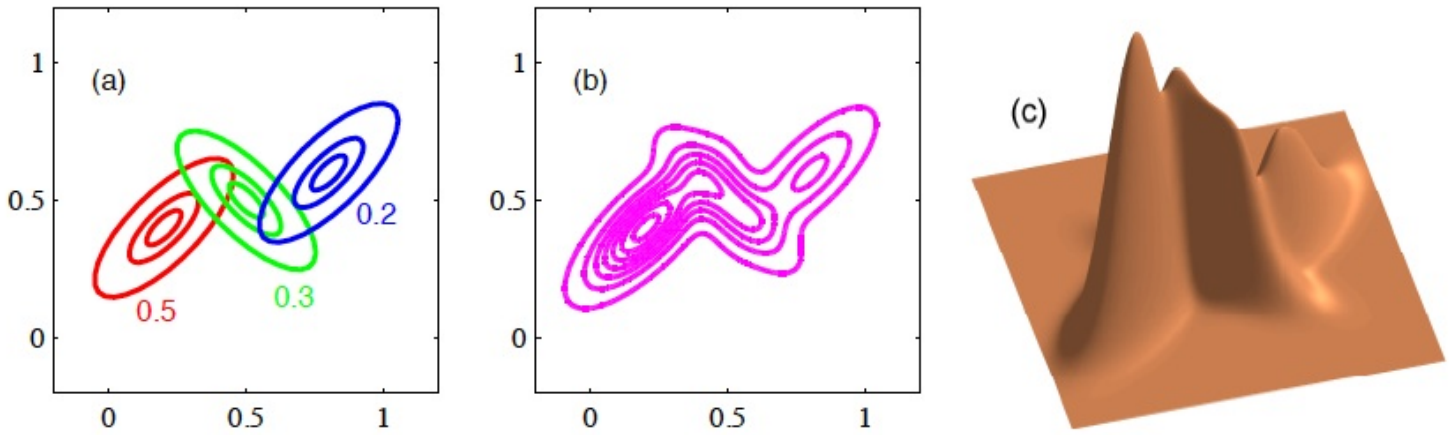
Multivariate Gaussian Distribution

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

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 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 : Kevin Murphy's course notes (2006), Bilmes's tutorial, (1998)

Introduction of latent variables 1/2

Let us introduce the K-dimensional indicator variable $\mathbf{z}$ and its associated observations $\mathbf{z}_i$ jointly observed with x_i.

- $p(z_k = 1) = \pi_k$ and $p(\mathbf{z}) = \prod_{k=1}^{K} \pi_k^{z_k}$
- z_{ik} indicates if data i comes from the k^{th} Gaussian ; $\sum_k z_{ik} = 1$
- $p(x_i|z_{ik} = 1) = p_k(x_i)$

Complete likelihood

$$\prod_{i=1}^{n} p(x_i, \mathbf{z}_i) = \prod_{i=1}^{n} p(\mathbf{z}_i) p(x_i|\mathbf{z}_i) = \prod_{i=1}^{n} \prod_{k=1}^{K} \pi_k^{z_{ik}} p_k(x_i)^{z_{ik}}$$

Introduction of latent variables 2/2

More practical is the *complete* log-likelihood :

$$\begin{aligned}\mathcal{L}((x_1, z_1)\ldots,(x_n, z_n); \theta) &= \ln \prod_{i=1}^{n}\prod_{k=1}^{K} \pi_k^{z_{ik}} p_k(x_i)^{z_{ik}} \\ &= \sum_{i=1}^{n}\sum_{k=1}^{K} z_{ik}(\ln \pi_k + \ln p_k(x_i))\end{aligned}$$

Key idea of Expectation-Maximization algorithm

- Variables $\mathbf{z}_i$ are latent and our target is to estimate the weights, means and covariances
- if we knew the values of the latent variables, then maximizing the complete log-likelihood would be easy.

$\rightarrow$ we would just have to apply the closed form solution to estimate μ_k and Σ_k for data falling into cluster k.

- since we do not know them, let us estimate them
- we will maximize the *expected* complete log-likelihood instead of the complete log-likelihood
- then since the estimate of $\mathbf{z}_i$ depends on the parameters, we'll have to re-estimate them after each update of θ

Expectation maximization algorithm (Demspter et al. 1977) 1/3

E-step :

For given values of the parameters, we can compute the expected values of the latent variables :
r_{ik}= the responsibility of model k for data i.

$$r_{ik} = \mathbb{E}[z_{ik}]$$

Expectation maximization algorithm (Demspter et al. 1977) 2/3

E-step :

$$
\begin{aligned}
r_{ik} &= \mathbb{E}[z_{ik}] \\
&= P(z_{ik} = 1 | x_i, \theta), \text{ now, using the Bayes rule, we get :} \\
&= \frac{P(z_{ik} = 1) p(x_i | z_{ik} = 1, \theta)}{\sum_j P(z_{ij} = 1) p(x_i | z_{ij} = 1, \theta)} \\
&= \frac{\pi_k p(x_i | z_{ik} = 1, \theta)}{\sum_j \pi_j p_j(x_i)}
\end{aligned}
$$

for the sake of simplicity, we denoted by θ, μ and Σ.

Expectation maximization algorithm (Demspter et al. 1977) 3/3

M-step :

Maximize the expected complete log-likelihood :

$$\mathbb{E}[\mathcal{L}((x_1, z_1) \ldots, (x_n, z_n); \theta)] = \sum_{i=1}^{n} \sum_{k=1}^{K} r_{ik} \{\ln \pi_k + \ln p_k(x_i)\}$$

Parameter update :

$$\pi_k = \frac{\sum_i r_{ik}}{n}$$

$$\mu_k = \frac{\sum_i r_{ik} x_i}{\sum_i r_{ik}}$$

$$\Sigma_k = \frac{\sum_i r_{ik} (x_i - \mu_k)(x_i - \mu_k)^T}{\sum_i r_{ik}}$$

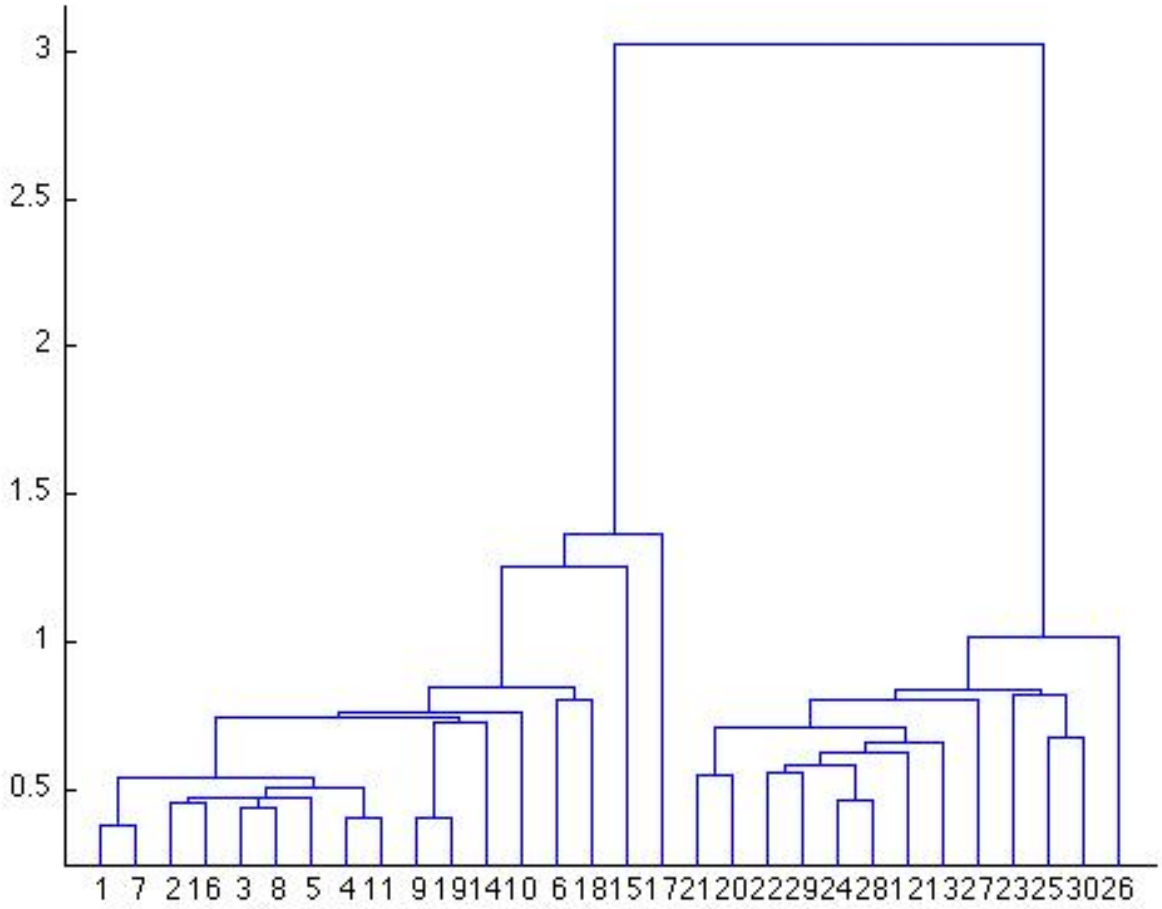
Expectation maximization algorithm

- EM algorithm : iterate E-step and M-step until the log-likelihood does not increase anymore
- Local convergence only
- Need to restart the algorithm with different initial guesses

Outline

Principle of Hierarchical clustering

Goal build a dendrogram

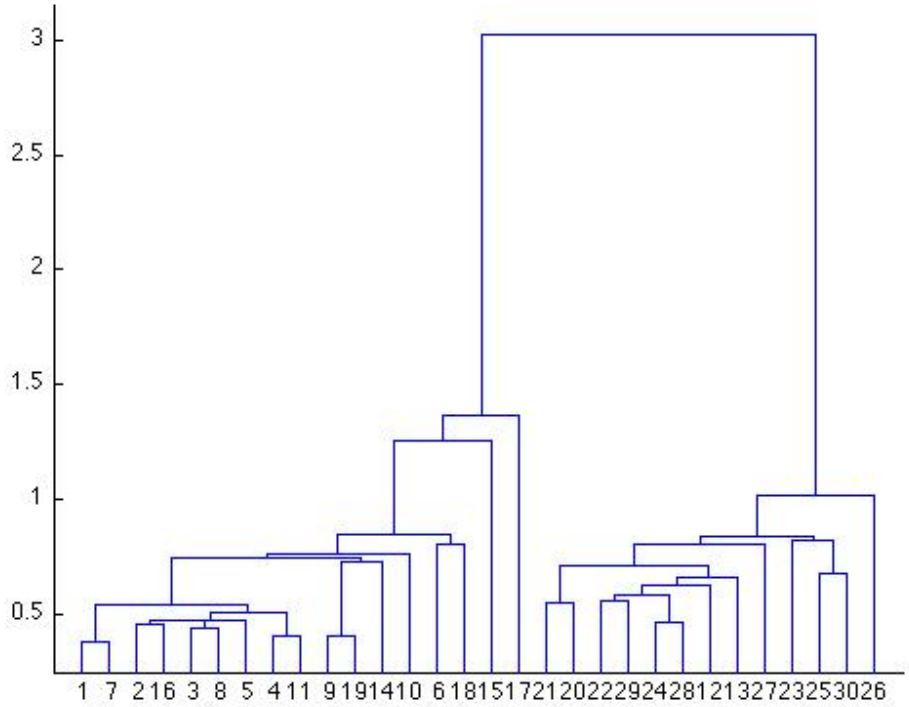


Hierarchical Agglomerative clustering

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

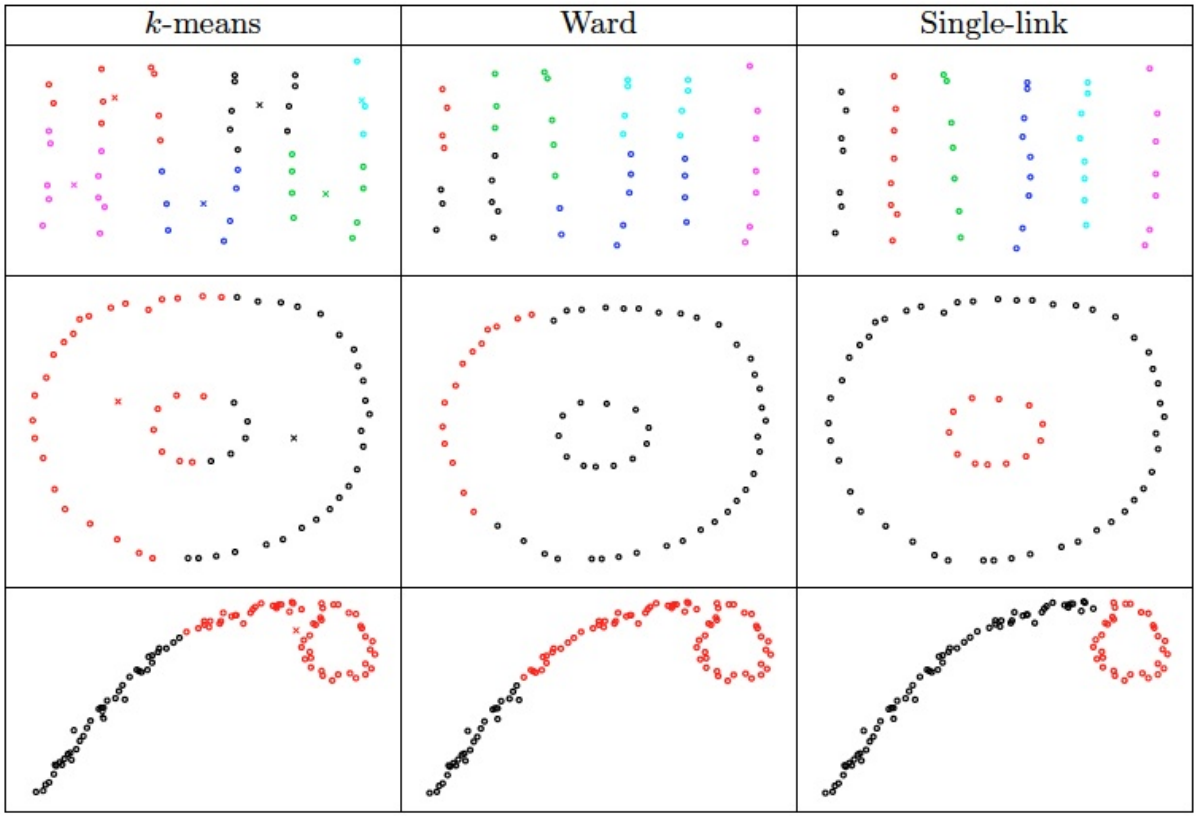
$\rightarrow$ *favours compactness*

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

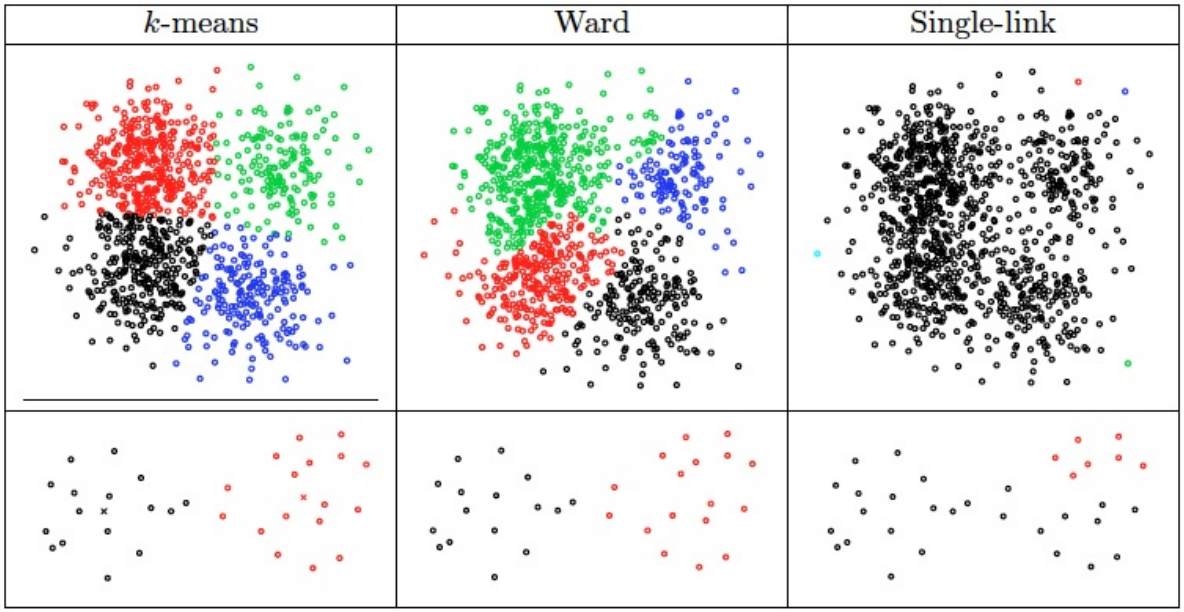
  m_A (resp. m_B) : center of gravity of A (resp. B)

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

Examples 1



Examples 2



Outline

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 S be the dataset.

- $f = 0.8$
- for k=2 to k_{max} do
  - for b=1 to B do
    - S_1 = subsample(S,f) : a subsample with a fraction f of data
    - S_2 = subsample(S,f) : a subsample with a fraction f of data
    - C_1 = cluster(S_1,k)
    - C_2 = cluster(S_2,k)
    - intersect = $S_1 \cap S_2$
    - 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