

Machine Learning – Course n°2

M1 CHEL[s] – Jan.-Apr. 2020

Yohann De Castro & Aurélien Garivier



Unsupervised Learning: Clustering

Unsupervised learning

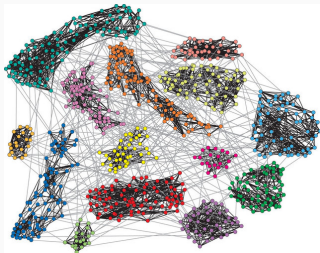
- **Marketing:** finding groups of customers with similar behavior given a large database of customer data containing their properties and past buying records
- **Biology:** classification of plants and animals given their features
- **Insurance:** identifying groups of motor insurance policy holders with a high average claim cost, identifying frauds
- **City-planning:** identifying groups of houses according to their house type, value and geographical location
- **Internet:** document classification, clustering weblog data to discover groups of similar access patterns

Two main directions

- **Data:** base of customer data containing their properties and past buying records
- **Goal:** Use the customers *similarities* to find groups

Two directions:

- **Clustering:** propose an explicit *grouping* of the customers
- **Visualization:** propose a representation of the customers so that the groups are *visibles*



Supervised learning reminder

- Training data $D_n = [(x_1, y_1), \dots, (x_n, y_n)]$
- (x_i, y_i) i.i.d \mathbb{P} on $\mathcal{X} \times \mathcal{Y}$
- Construct a predictor $\hat{f} : \mathcal{X} \rightarrow \mathcal{Y}$ using D_n
- Loss $\ell(y, f(x))$ measures how well $f(x)$ predicts y well
- Aim is to minimize the **generalization error**

$$\mathcal{E}_{\mathcal{X}, \mathcal{Y}}(\ell(Y, \hat{f}(X)) | D_n) = \int \ell(y, \hat{f}(x)) d\mathbb{P}(x, y).$$

The goal is clear

- Predict y based on feature x

Heard on the street

- Supervised learning is **solved**. Unsupervised learning **isn't**

Unsupervised learning

- Training data $D_n = [x_1, \dots, x_n]$
- **Loss:** Not Clear
- **Aim:** Not Clear

The goal is **unclear**.

Classical tasks

- **Clustering:** construct groups of data in *homogeneous* classes
- **Dimension reduction:** construct a *map* of the data in a *low-dimension* space without *distorting* it too much

Motivations

- Interpretation of the groups
- Use of the groups in further processing

Clustering

Clustering

- Training data $D_n = \{x_1, \dots, x_n\}$ with $x_i \in \mathbb{R}^d$
- Recover **Latent** groups
- Construct $f : \mathbb{R}^d \rightarrow \{1, \dots, K\}$ which affects cluster number to x_i

$$f : x_i \mapsto k_i$$

- No ground truth for k_i

Warning

- Choice of K is hard

Roughly two approaches

- Partition-based
- Model-based

K-means

K-means

- Fix $K \geq 2$, n data points $x_i \in \mathbb{R}^d$
- Find centroids c_1, \dots, c_K that minimizes the **quantification** error

$$\sum_{i=1}^n \min_{k=1, \dots, K} \|x_i - c_k\|_2^2$$

- **Impossible** to find the exact solution (NP Complete)

K-means algorithm

Lloyd (1981) proposes a way of finding local solutions

K-means algorithm

- Choose at random K centroids $\{c_1, \dots, c_K\}$
- For each $k \in \{1, \dots, K\}$, find the set C_k of points that are closer to c_k than any $c_{k'}$ for $k' \neq k$
- Update the centroids:

$$c_k = \frac{1}{|C_k|} \sum_{i \in C_k} x_i$$

- Repeat the two previous steps until the sets C_k don't change

K-means algorithm

Lloyd (1981) proposes a way of finding local solutions

K-means algorithm

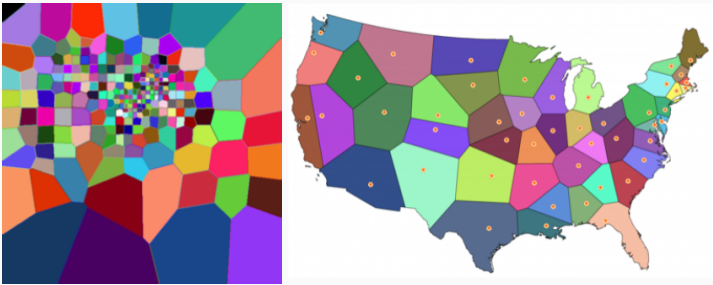
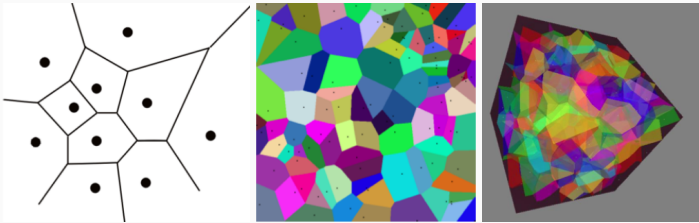
- Choose at random K centroids $\{c_1, \dots, c_K\}$
- For each $k \in \{1, \dots, K\}$, find the set C_k of points that are closer to c_k than any $c_{k'}$ for $k' \neq k$
- Update the centroids:

$$c_k = \frac{1}{|C_k|} \sum_{i \in C_k} x_i$$

- Repeat the two previous steps until the sets C_k don't change

Remark: K-means computes a **Voronoi partitioning**, it implicitly assumes **convex clusters**, that are **uniquely** defined by their **centroids**.

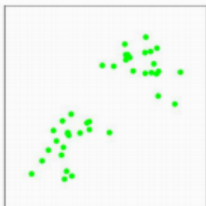
Voronoi partitioning



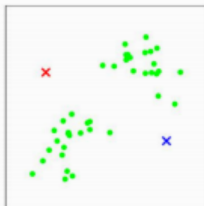
Bryant park



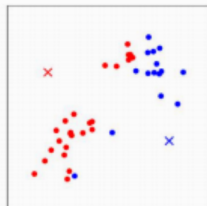
K-means



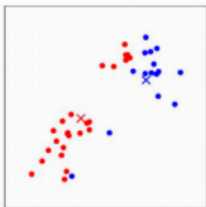
(a)



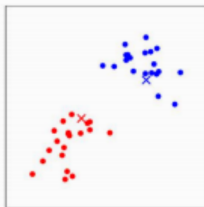
(b)



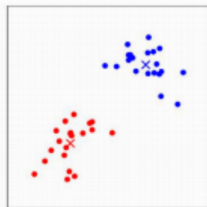
(c)



(d)

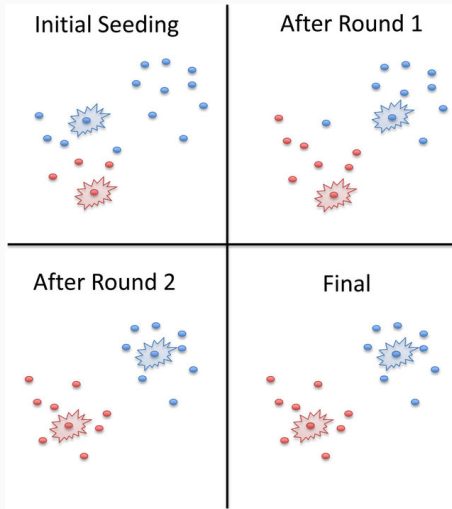


(e)



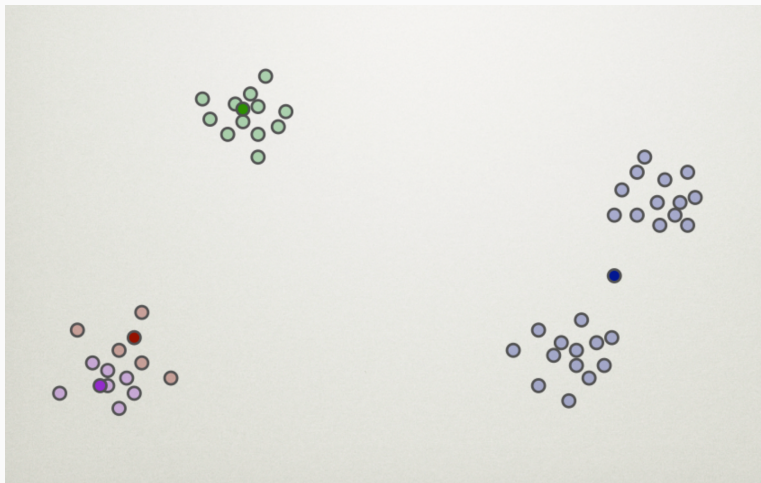
(f)

K-means



Initialization problem

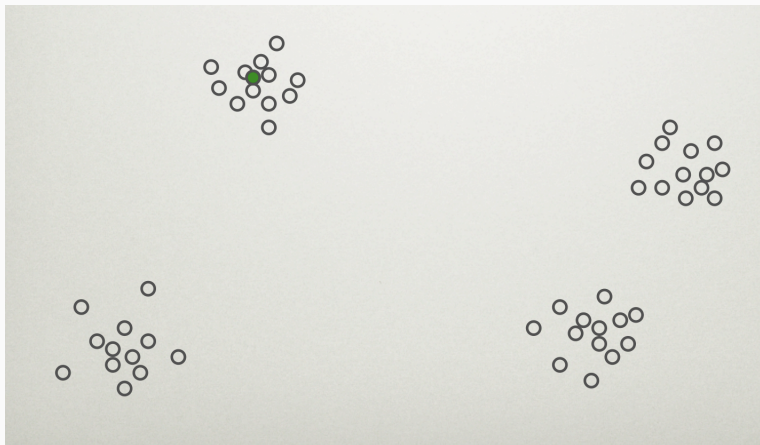
K-means very sensitive to the choice of initial points



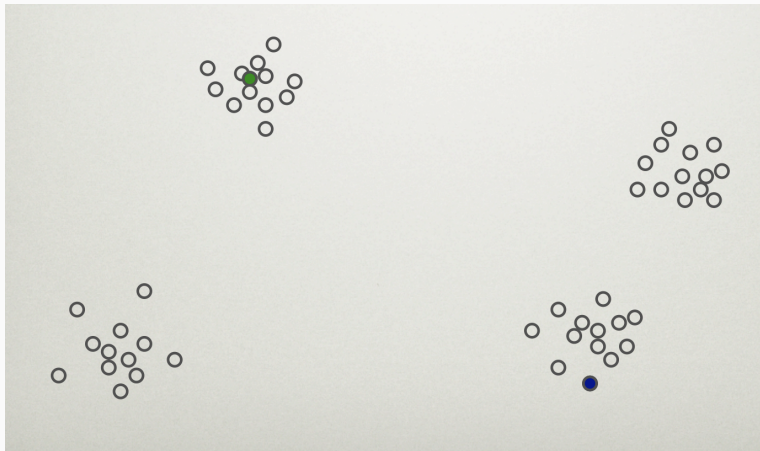
An easy solution:

- Pick a first point at random
- Choose the next initial point the **farthest** from the previous ones

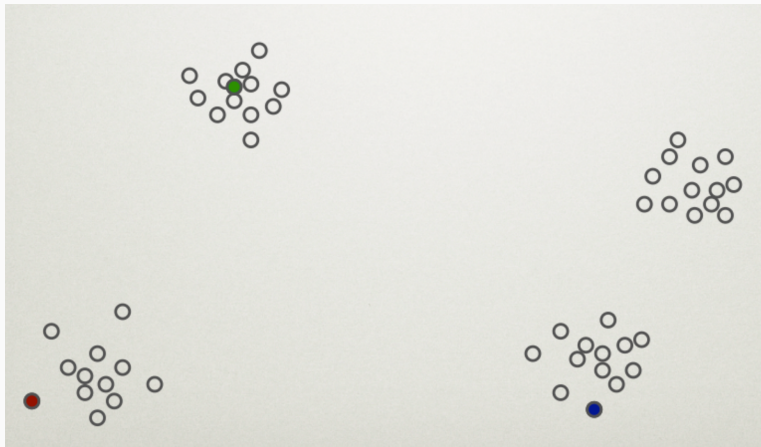
Farthest point initialization



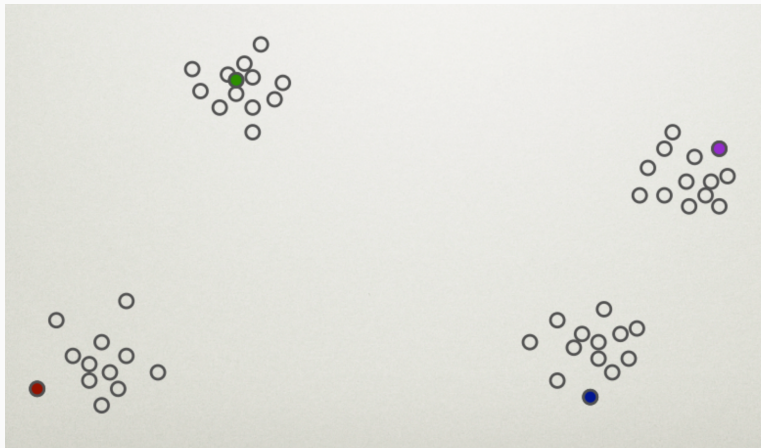
Farthest point initialization



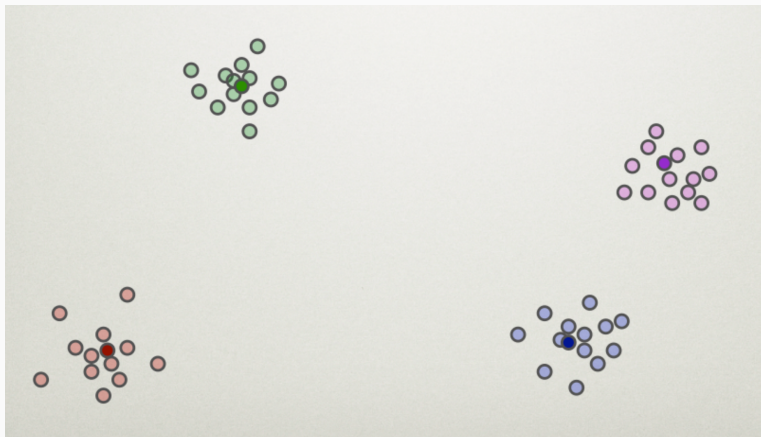
Farthest point initialization



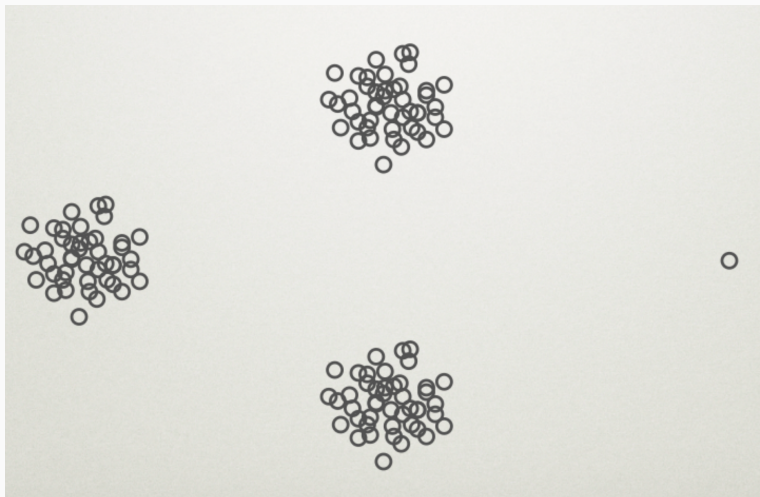
Farthest point initialization



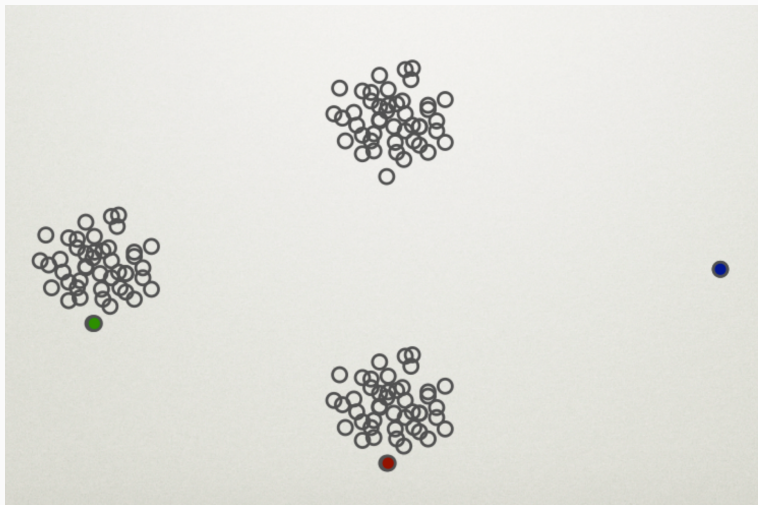
Farthest point initialization



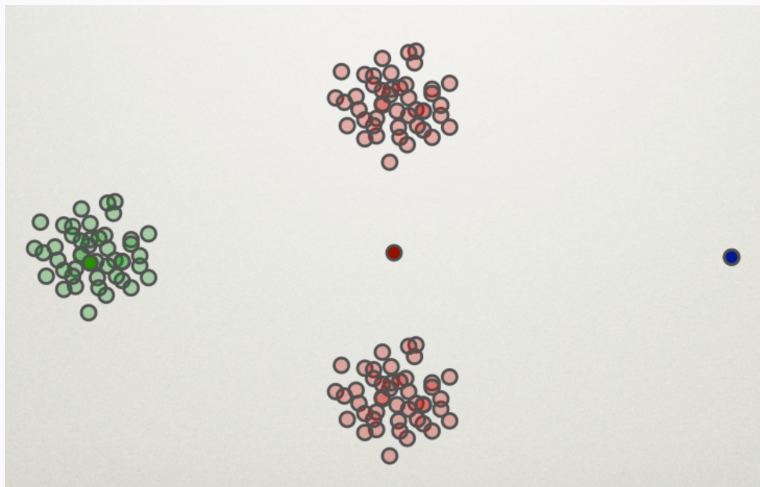
But, **very sensitive to outliers**



But, very sensitive to outliers



But, **very sensitive to outliers**



Robustness to outliers: K -means++

Principle

Pick the initial centroids as follows

- 1 Pick uniformly at random $i \in \{x_1, \dots, x_n\}$, put $c_1 \leftarrow x_i$
- 2 $k \leftarrow k + 1$
- 3 Sample $i \in \{X_1, \dots, X_n\}$ with probability

$$\frac{\min_{k'=1, \dots, k-1} \|x_i - c_{k'}\|_2^2}{\sum_{i'=1}^n \min_{k'=1, \dots, k-1} \|X_{i'} - c_{k'}\|_2^2}$$

- 4 Put $c_k \leftarrow x_i$
- 5 If $k < K$ go back to step 2.

Then use K -means based on these initial clusters

This is between random initialization and furthest point initialization

In expectation, leads to a solution **close to the optimum**. Define the quantification error

$$Q_n(c_1, \dots, c_K) = \sum_{i=1}^n \min_{k=1, \dots, K} \|x_i - c_k\|_2^2.$$

[Arthur and Vassilvitskii, 2006]

If c_1, \dots, c_K are centroids obtained with K-means++, then

$$\mathcal{E}[Q_n(c_1, \dots, c_K)] \leq 8(\log K + 2) \min_{c'_1, \dots, c'_K} Q_n(c'_1, \dots, c'_K)$$

where \mathcal{E} is with respect to random choice of initial centroids

Complexity

$$O(n \times K \times n_{it})$$

K-means: Pros and Cons

Pros

- **Simple:** easy to implement
- **Efficient:** guaranteed to converge in finite number of iterations
 $O(n \times K \times n_{it})$
- **Popular**

Cons

- **Notion of mean:** means need to be defined
- **Number of clusters:** K needs to be specified
- **Sensitive** to outliers
↳ can be fixed by subsampling and/or outlier detection
- **Roundish clusters:** not suited for spherical data, fails if clusters are not convex/round

Mixture models

Model-based clustering

- use a **model** on data with clusters
- using a **mixture** of distributions with different location/mean

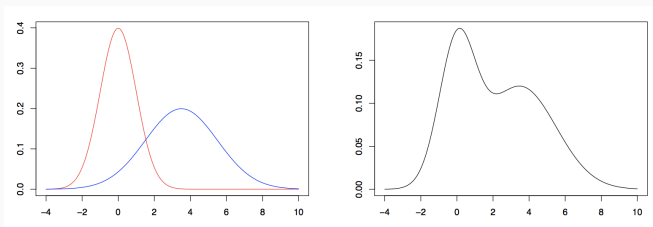
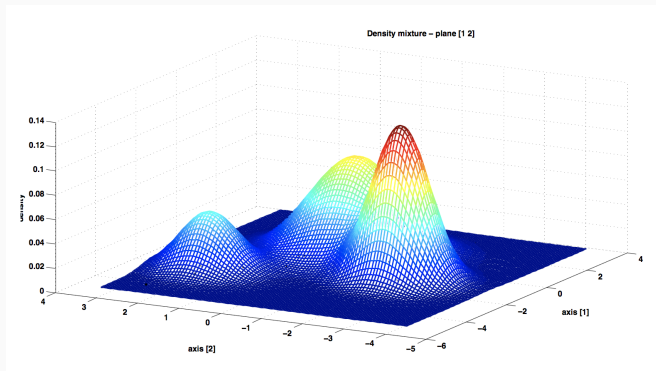


Figure 1: Gaussian Mixture in dimension 1

Model-based clustering

- use a **model** on data with clusters
- using a **mixture** of distributions with different location/mean



Model-based clustering

- use a **model** on data with clusters
- using a **mixture** of distributions with different location/mean

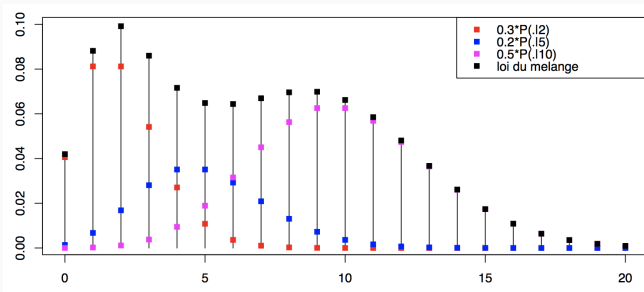
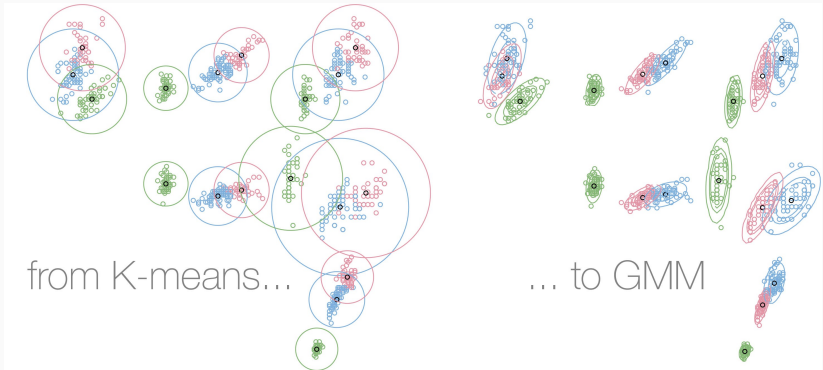
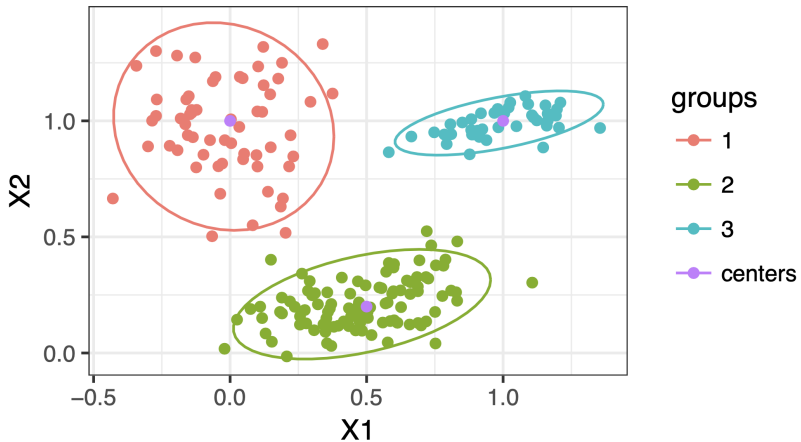


Figure 1: Poisson Mixture

Gaussian Mixture Models (GMM)

from K-means to GMM





Mixture Models

Mixture of densities f_1, \dots, f_K

- $K \in \mathbb{N}^*$ (number of clusters)
- $(p_1, \dots, p_K) \in (\mathbb{R}^+)^K$ s.t. $\sum_{k=1}^K p_k = 1$

Mixture density

$$f = \sum_{k=1}^K p_k f_k$$

Gaussian Mixtures Model (GMM)

- Put $f_k = \varphi_{\mu_k, \Sigma_k}$ = density of $N(\mu_k, \Sigma_k)$, where we recall

$$\varphi_{\mu_k, \Sigma_k}(x) = \frac{1}{(2\pi)^{d/2} \sqrt{\det(\Sigma_k)}} \exp\left(-\frac{1}{2}(x - \mu_k)^\top \Sigma_k^{-1}(x - \mu_k)\right)$$

with $\Sigma_k \succ 0$

Proposition (Latent variable)

- Let $\{P_\theta = f_{\theta\mu} : \theta \in \Theta\}$ be a statistical model (for r.v. in \mathbb{R}^d) dominated by μ , K be a positive integer, $(\theta_1, \dots, \theta_K) \in \Theta^K$
- Let (p_1, \dots, p_K) be a probability vector.
- Let now $Z \sim \mathcal{M}(\mathbf{1}, p_1, \dots, p_K)$ be a multinomial variable
- $Y := \sum_{k=1}^K k \mathbb{1}_{Z_k=1}$.

Then

$$\forall k \in [K]: \mathbb{P}(Y = k) = p_k.$$

In addition, let X be a random variable such that $X|Y \sim P_{\theta_Y}$. Then

$$X \sim \sum_{k=1}^K p_k P_{\theta_k}.$$

Proposition bridges the gap between mixture models and clustering:

- when X is distributed w.r.t to a mixture model with k components, we describe it with k clusters defined by a latent variable $Y \in [k]$.
- Conversely, clustering is naturally modeled by a mixture model: clusters are distributed w.r.t conditional variables $X|Y$?

Thus, we focus on the **marginal distribution** of X , which is, by Bayes' theorem:

$$\forall x \in \mathbb{R}^d: f(x) = \sum_{k=1}^K p_k f_{\theta_k}(x),$$

where $p_k = \mathbb{P}(Y = k)$ is the prior probability of a cluster.

Then, the Bayes rule for clustering is given by

$$g^* : x \mapsto \arg \max_{1 \leq k \leq K} \mathbb{P}_{(p_1, \dots, p_K, \theta)} (Y = k | X = x) = \arg \max_{1 \leq k \leq K} p_k f_{\theta_k}(x).$$

The final partitioning $\{C_1, \dots, C_K\}$ is $C_k = \{x \in \mathbb{R}^d : g^*(x) = k\}$.

\rightsquigarrow explains how to sample X according to a mixture model.

Gaussian Mixtures Model

- Statistical model with density

$$f_{\theta} = \sum_{k=1}^K p_k \varphi_{\mu_k, \Sigma_k},$$

- Parameter $\theta = (p_1, \dots, p_K, \mu_1, \dots, \mu_K, \Sigma_1, \dots, \Sigma_K)$
- Goodness-of-fit is:

$$R_n(\theta) = -\log\text{-likelihood} = -\sum_{i=1}^n \log \left(\sum_{k=1}^K p_k \varphi_{\mu_k, \Sigma_k}(x_i) \right)$$

- A local minimizer $\hat{\theta}$ is typically obtained using an algorithm called **Expectation-Minimization (EM) algorithm**

Gaussian Mixture Models (GMM)

EM algorithm

Idea:

- there is a **hidden structure** in the model
- knowing this structure, the optimization problem is easier

Idea:

- there is a **hidden structure** in the model
- knowing this structure, the optimization problem is easier

Indeed, each point X_i belongs to an unknown class $k \in \{1, \dots, K\}$

- Put $C_{i,k} = 1$ when i belongs to class k , $C_{i,k} = 0$ otherwise.
- We don't observe $\{C_{i,k}\}_{1 \leq i \leq n, 1 \leq k \leq K}$
- We say that these are **latent variables**
- Put $\mathcal{C}_k = \{i : C_{i,k} = 1\}$, then $\mathcal{C}_1, \dots, \mathcal{C}_K$ is a partition of $\{1, \dots, n\}$.

Generative model:

- i belongs to class \mathcal{C}_k with probability p_k , namely

$$C_i = (C_{i,1}, \dots, C_{i,K}) \sim \mathcal{M}(\mathbf{1}, p_1, \dots, p_K)$$

[multinomial distribution with parameter $(\mathbf{1}, p_1, \dots, p_K)$].

- $X_i \sim \varphi_{\mu_k, \Sigma_k}$ if $C_{i,k} = 1$

- The joint distribution of (X, C) is, according to this model

$$f_{\theta}(x, c) = \prod_{k=1}^K (p_k \varphi_{\mu_k, \Sigma_k}(x))^{c_k}$$

$[c_k = 1$ for only one k and 0 elsewhere] and the **marginal density in X is the one of the mixture:**

$$f_{\theta}(x) = \sum_{k=1}^K p_k \varphi_{\mu_k, \Sigma_k}(x)$$

- This generative model adds a latent variable C , in such a way that the marginal distribution of (X, C) in X is indeed the one of the mixture f_{θ}

Complete Likelihood

- Put $\mathbf{X} = (X_1, \dots, X_n)$ and $\mathbf{C} = (C_1, \dots, C_n)$
- Do as if **we observed** the latent variables \mathbf{C}
- Write a **completed** likelihood for these “virtual” observations (joint distribution of (\mathbf{X}, \mathbf{C})):

$$L_c(\theta; \mathbf{X}, \mathbf{C}) = \prod_{i=1}^n \prod_{k=1}^K (p_k \varphi_{\mu_k, \Sigma_k}(X_i))^{C_{i,k}}$$

and the completed log-likelihood:

$$l_c(\theta; \mathbf{X}, \mathbf{C}) = \sum_{i=1}^n \sum_{k=1}^K C_{i,k} (\log p_k + \log \varphi_{\mu_k, \Sigma_k}(X_i)).$$

[Dempster et al. (77)]

(E=Expectation, M=Maximization)

Initialize $\theta^{(0)}$

for $t = 0, \dots$, until *convergence*, repeat:

- 1 (E)-step: [Expectation with respect to the latent variables, for the previous value of θ]

Compute

$$\theta \mapsto Q(\theta, \theta^{(t)}) = \mathcal{E}_{\theta^{(t)}} \left[\ell_c(\theta; \mathbf{X}, \mathbf{C}) \mid \mathbf{X} \right]$$

- 2 (M)-step: [Maximize this expectation]

Compute

$$\theta^{(t+1)} \in \arg \max_{\theta \in \Theta} Q(\theta, \theta^{(t)})$$

- (E) and (M) steps often have explicit solutions

Theorem

The sequence $\theta^{(t)}$ obtained using EM Algorithm satisfies

$$\ell(\theta^{(t+1)}; \mathbf{X}) \geq \ell(\theta^{(t)}; \mathbf{X})$$

for any t .

- A each step, EM increases the likelihood
- Initialization will be very important (usually done using K-Means or K-Means++)

Remains to check that $Q_1(\theta^{(t+1)}, \theta^{(t)}) - Q_1(\theta^{(t)}, \theta^{(t)}) \leq 0$:

$$\begin{aligned} Q_1(\theta^{(t+1)}, \theta^{(t)}) - Q_1(\theta^{(t)}, \theta^{(t)}) &= \mathcal{E}_{\theta^{(t)}} [\ell(\theta^{(t+1)}; \mathbf{C}|\mathbf{X}) - \ell(\theta^{(t)}; \mathbf{C}|\mathbf{X})|\mathbf{X}] \\ &= \int \log \left(\frac{f_{\theta^{(t+1)}}(c|\mathbf{X})}{f_{\theta^{(t)}}(c|\mathbf{X})} \right) f_{\theta^{(t)}}(c|\mathbf{X}) \mu(dc) \\ &\stackrel{(\text{Jensen})}{\leq} \log \int f_{\theta^{(t+1)}}(c|\mathbf{X}) \mu(dc) = 0, \end{aligned}$$

This proves $\ell(\theta^{(t+1)}; \mathbf{X}) \geq \ell(\theta^{(t)}; \mathbf{X})$ for any t

So what is the EM Algorithm, and where do we use this?

- It is an algorithm that allows to optimize a likelihood with **missing or latent data**
- For a mixture distribution, we come up with natural latent variables, that simplify the original optimization problem

Gaussian Mixture Models (GMM)

EM and GMM

Consider the completed likelihood

$$\ell_c(\theta; \mathbf{X}, \mathbf{C}) = \sum_{i=1}^n \sum_{k=1}^K C_{i,k} (\log p_k + \log \varphi_{\mu_k, \Sigma_k}(X_i)),$$

where

$$\theta = (p_1, \dots, p_K, \mu_1, \dots, \mu_K, \Sigma_1, \dots, \Sigma_K).$$

What are the (E) and (M) steps in this case?

(E)-Step

Compute

$$\mathcal{E}_{\theta^{(t)}} \left[\ell_c(\theta; \mathbf{X}, \mathbf{C}) \mid \mathbf{X} \right] = \sum_{i=1}^n \sum_{k=1}^K \mathcal{E}_{\theta^{(t)}} [C_{i,k} \mid \mathbf{X}] (\log p_k + \log \varphi_{\mu_k, \Sigma_k}(X_i)),$$

which is simply:

$$\mathcal{E}_{\theta} [C_{i,k} \mid \mathbf{X}] = \mathbb{P}_{\theta}(C_{i,k} = 1 \mid X_i) =: \pi_{i,k}(\theta),$$

where

$$\pi_{i,k}(\theta) = \frac{p_k \varphi_{\mu_k, \Sigma_k}(X_i)}{\sum_{k'=1}^K p_{k'} \varphi_{\mu_{k'}, \Sigma_{k'}}(X_i)}.$$

We call $\pi_{i,k}(\theta)$ the “soft-assignment” of i in class k .

(E)-Step

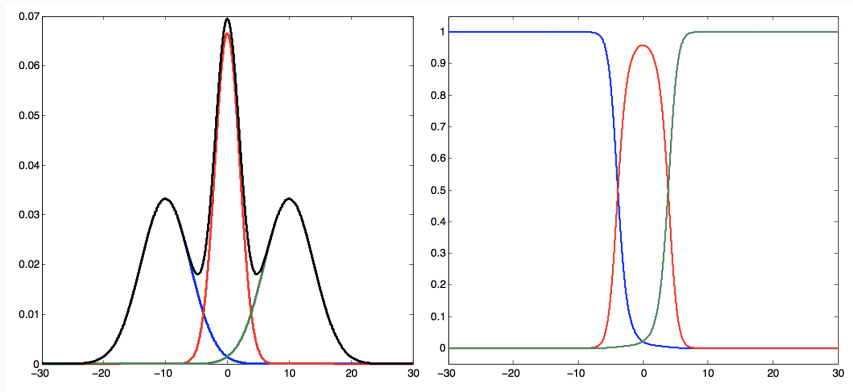


Figure 2: Soft-assignments : $x \mapsto \frac{p_k \varphi_{\mu_k, \Sigma_k}(x)}{\sum_{k'=1}^K p_{k'} \varphi_{\mu_{k'}, \Sigma_{k'}}(x)}$.

Compute

$$\theta^{(t+1)} = (p_1^{(t+1)}, \dots, p_K^{(t+1)}, \mu_1^{(t+1)}, \dots, \mu_K^{(t+1)}, \Sigma_1^{(t+1)}, \dots, \Sigma_K^{(t+1)})$$

using:

$$p_k^{(t+1)} = \frac{1}{n} \sum_{i=1}^n \pi_{i,k}(\theta^{(t)}),$$
$$\mu_k^{(t+1)} = \frac{\sum_{i=1}^n \pi_{i,k}(\theta^{(t)}) X_i}{\sum_{i=1}^n \pi_{i,k}(\theta^{(t)})}$$
$$\Sigma_k^{(t+1)} = \frac{\sum_{i=1}^n \pi_{i,k}(\theta^{(t)}) (X_i - \mu_k^{(t+1)})(X_i - \mu_k^{(t+1)})^\top}{\sum_{i=1}^n \pi_{i,k}(\theta^{(t)})}.$$

This is natural: estimation for the means and covariances, weighted by the soft-assignments


So, for $t = 1, \dots$, iterate (E) and (M) until convergence:

$$\pi_{i,k}(\theta^{(t)}) = \frac{p_k^{(t)} \varphi_{\mu_k^{(t)}, \Sigma_k^{(t)}}(X_i)}{\sum_{k'=1}^K p_{k'}^{(t)} \varphi_{\mu_{k'}^{(t)}, \Sigma_{k'}^{(t)}}(X_i)}$$
$$p_k^{(t+1)} = \frac{1}{n} \sum_{i=1}^n \pi_{i,k}(\theta^{(t)})$$
$$\mu_k^{(t+1)} = \frac{\sum_{i=1}^n \pi_{i,k}(\theta^{(t)}) X_i}{\sum_{i=1}^n \pi_{i,k}(\theta^{(t)})}$$
$$\Sigma_k^{(t+1)} = \frac{\sum_{i=1}^n \pi_{i,k}(\theta^{(t)}) (X_i - \mu_k^{(t+1)})(X_i - \mu_k^{(t+1)})^\top}{\sum_{i=1}^n \pi_{i,k}(\theta^{(t)})}.$$

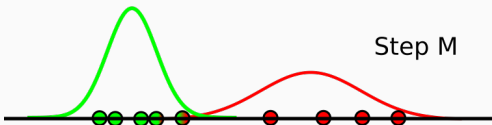
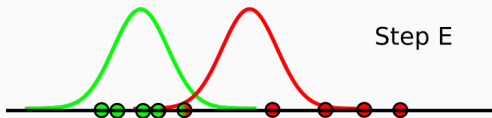
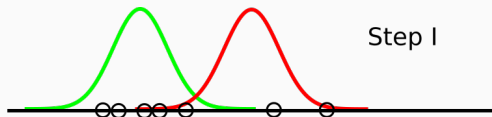
We obtain an estimator $\hat{\theta} = (\hat{p}_1, \dots, \hat{p}_K, \hat{\mu}_1, \dots, \hat{\mu}_K, \hat{\Sigma}_1, \dots, \hat{\Sigma}_K)$.

Complexity

$$O(n \times K \times n_{it})$$

 with n_{it} number of iterations

EM-Algorithm on one picture



Clustering: the maximum a posteriori (MAP) rule

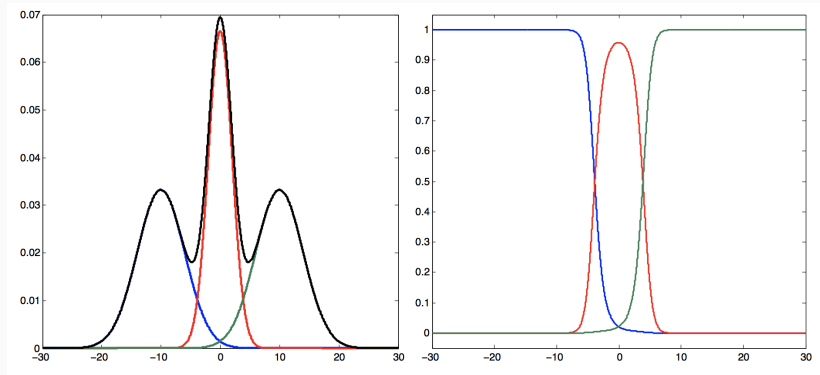
- Given $\hat{\theta}$, how to affect a cluster number k to a given $x \in \mathbb{R}^d$?
- Compute the the **soft-assignments**

$$\pi_k(x) = \frac{\hat{p}_k \varphi_{\hat{\mu}_k, \hat{\Sigma}_k}(x)}{\sum_{k'=1}^K \hat{p}_{k'} \varphi_{\hat{\mu}_{k'}, \hat{\Sigma}_{k'}}(x)}$$

- Consider

$$i \in \mathcal{C}_k \text{ if } \pi_k(x) > \pi_{k'}(x) \text{ for any } k' \neq k$$

The MAP rule



Mixture $f_{\theta}(x)$

Soft-assignment $\pi_k(x)$

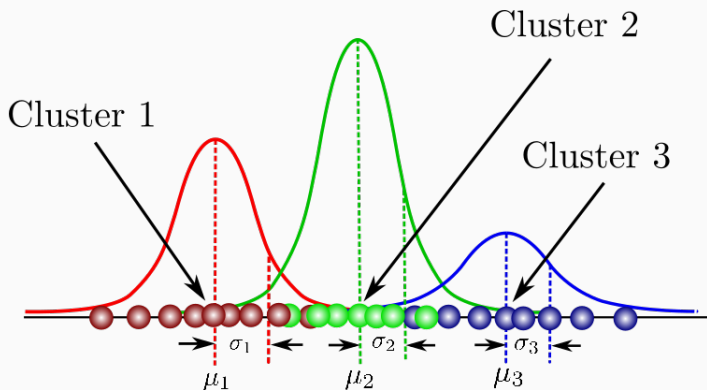
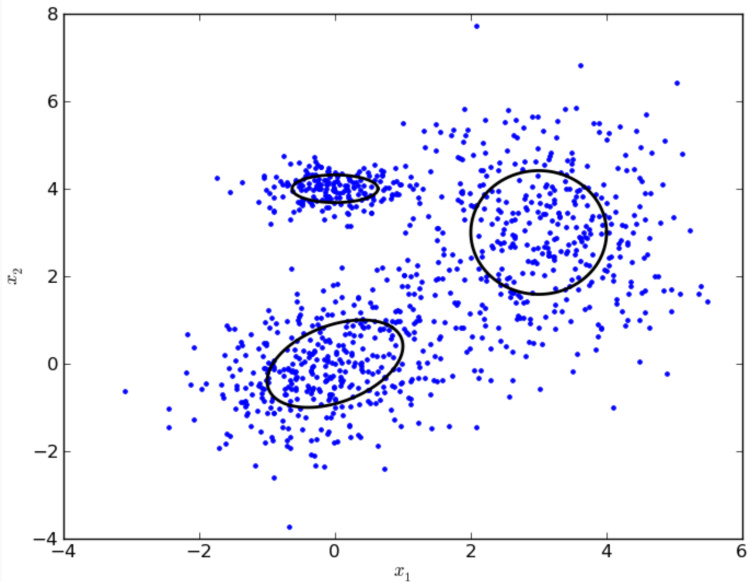
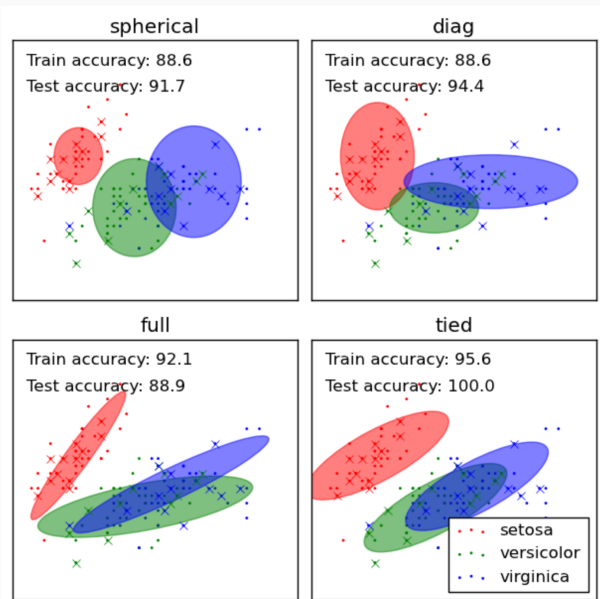


Figure 3: Soft-assignments : $x \mapsto \frac{p_k \varphi_{\mu_k, \Sigma_k}(x)}{\sum_{k'=1}^K p_{k'} \varphi_{\mu_{k'}, \Sigma_{k'}}(x)}$.

Gaussian Mixtures Model



Gaussian Mixtures Model



Point-based objectives

Point-based objectives

Contrarily to center-based objectives, point-based objectives **do not** require to compute a cluster center.

The distortion $D(C_1, \dots, C_k)$ to minimize is computed based on pair of points belonging to clusters. For example, the **sum of in-cluster distances** is

$$\hat{D}(C_1, \dots, C_k) = \sum_{k=1}^K \sum_{X, Y \in \hat{C}_k} d(X, Y).$$

with $\hat{C}_k = C_k \cap \{X_i : i = 1; \dots, n\}$

Point-based objectives

Let $s : \mathcal{X} \times \mathcal{X} \rightarrow [0, 1]$ be a similarity measure. Another example lies in the distortion defined by the **sum of interclass similarities**:

$$D(C_1, \dots, C_K) = \mathbb{E} \left(\sum_{k=1}^K s(X, Y) \mathbb{1}_{X \in C_k \cap Y \notin C_k} \right).$$

- **center-based approach**: making sure that points in the same cluster are similar
- **point-based objectives approach**: points separated into different clusters should be dissimilar

Graph-cut problem

Representation by a similarity graph

- each vertex represents a data point X_i
- vertices are connected by an edge whose weight is their similarity

Such a graph can be defined by the **similarity (or adjacency) matrix**

$$W = (s(X_i, X_j))_{1 \leq i, j \leq n}$$

Given the index sets I_k of each empirical cluster \hat{C}_k , the previous point-based objective function has an empirical twin given by:

$$\hat{D}(C_1, \dots, C_k) = \sum_{j=1}^k \sum_{\substack{i \in I_j \\ \ell \notin I_j}} W_{i,\ell}.$$

Minimizing $\hat{D}(C_1, \dots, C_k)$ is often referred as the **graph cut problem**.

Similarity graph

Consider a **similarity graph** $G = (V, E)$, for which the vertices $V = (v_1, \dots, v_n)$ represent the points (X_1, \dots, X_n)

- Two vertices v_i and v_j are connected if the similarity $s(X_i, X_j) > 0$ (or $> \tau$ with τ a threshold)
- The edge between these two vertices is weighted by their similarity $s(X_i, X_j)$.
- The **weighted adjacency matrix** is $W = (s(X_i, X_j))_{1 \leq i, j \leq n}$.
- The graph G is assumed **undirected**, which is equivalent to W being **symmetric** (via a symmetric similarity measure s).

Definition

- The **degree of a vertex** $v_i \in V$ is $d_i = \sum_{l=1}^n W_{i,l}$.

Given $A \subset V$,

- the **size** of $A = |A|$ = the number of its vertices
- the **volume** of A , $\text{vol}(A) = \sum_{i \in [n]: v_i \in A} d_i$
- A is said **connected** if any two vertices of A can be joined by a path such that all intermediate points also lie in A .
- A is called a **connected component** if it is connected and if there are no connections between vertices in A and $V \setminus A$.

Similarity graph

When constructing a similarity graph, the goal is to model the local neighborhood relationships between data points.

↪ **3 popular similarity graphs** based on a given distance

$$d : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}_+.$$

The ϵ -neighborhood graph

- X_i and X_j are connected iff $d(X_i, X_j) \leq \epsilon$.
- If ϵ is small enough, all connected points are roughly at the same distance.
- The weights assigned are $W_{i,j} = 1$ (if $d(X_i, X_j) \leq \epsilon$ and 0 otherwise).
- Usually considered as an unweighted graph

Similarity graphs

k-nearest neighbor graph

- X_i and X_j are connected iff $X_i \in kNN(X_j)$ or $X_j \in kNN(X_i)$.
- $W_{i,j} = 1$ if X_i and X_j are connected and 0 otherwise.

Similarity graphs

k-nearest neighbor graph

- X_i and X_j are connected iff $X_i \in kNN(X_j)$ or $X_j \in kNN(X_i)$.
- $W_{i,j} = 1$ if X_i and X_j are connected and 0 otherwise.

Mutual k-nearest neighbor graph

- X_i and X_j are connected iff $X_i \in kNN(X_j)$ and $X_j \in kNN(X_i)$.
- $W_{i,j} = 1$ if X_i and X_j are connected and 0 otherwise.

Similarity graphs

k-nearest neighbor graph

- X_i and X_j are connected iff $X_i \in kNN(X_j)$ **or** $X_j \in kNN(X_i)$.
- $W_{i,j} = 1$ if X_i and X_j are connected and 0 otherwise.

Mutual k-nearest neighbor graph

- X_i and X_j are connected iff $X_i \in kNN(X_j)$ **and** $X_j \in kNN(X_i)$.
- $W_{i,j} = 1$ if X_i and X_j are connected and 0 otherwise.

The fully connected graph

- Points are connected if they have a similarity $s(X_i, X_j) > 0$
- The edges are weighted by $s(X_i, X_j)$
- A popular choice is the Gaussian similarity: $s(x, x') = e^{-\frac{d(x, x')^2}{2\sigma^2}}$, in which σ^2 plays a role similar to ϵ and k

Spectral clustering

Normalized graph-cut

- For $k = 2$, finding a minimal cut of a graph can be done efficiently: Stoer-Wagner algorithm
- Problem: often results in separating a vertex from the rest

One solution

Normalizing the empirical distortion either by the size of the clusters:

$$\hat{D}_r(C_1, \dots, C_K) = \sum_{k=1}^K \frac{1}{|\hat{C}_k|} \sum_{\substack{i \in I_k \\ \ell \notin I_k}} W_{i,\ell},$$

(let us remind that $|\hat{C}_k| = |I_k|$) or by their volume:

$$\hat{D}_n(C_1, \dots, C_K) = \sum_{k=1}^K \frac{1}{\text{vol}(\hat{C}_k)} \sum_{\substack{i \in I_k \\ \ell \notin I_k}} W_{i,\ell}.$$



These objectives are respectively called **ratio cut** and **normalized cut**.

- Problem: the balancing introduced by the cluster importance makes the minimization problem computationally hard to solve
- A relaxation procedure: **spectral clustering** algorithm.

Definition (Unnormalized graph Laplacian)

Let $W \in \mathbb{R}^{n \times n}$ be a symmetric matrix.

- The diagonal matrix $D \in \mathbb{R}^{n \times n}$ such that $D_{i,i} = \sum_{j=1}^n W_{i,j}, \forall i \in [n]$ is called the *degree matrix* of the graph defined by W
- $L = D - W$ is called the *Laplacian* of the graph defined by W

Proposition

Let W and L be respectively the adjacency matrix and the Laplacian of the similarity graph of (X_1, \dots, X_n) . For any positive integer K and for all partitioning (C_1, \dots, C_K) of (X_1, \dots, X_n) , we have

$$\hat{D}_r(C_1, \dots, C_K) = \text{tr}(H^\top L H),$$

where $H = \left(\frac{1}{\sqrt{|I_k|}} \mathbb{1}_{i \in I_k} \right)_{\substack{1 \leq i \leq n \\ 1 \leq k \leq K}}$.

In addition, the columns of H are orthonormal to each other ($H^\top H = I$).

First, denoting $h_j \in \mathbb{R}^n$ the columns of H (for $j \in [K]$), we have

$$\operatorname{tr}(H^\top LH) = \operatorname{tr}((L^{1/2}H)^\top (L^{1/2}H)) = \sum_{j=1}^K (L^{1/2}h_j)^\top (L^{1/2}h_j) = \sum_{j=1}^K h_j^\top Lh_j.$$

Proof ii

In addition, for all $u \in \mathbb{R}^n$,

$$\begin{aligned}u^{\top}Lu &= u^{\top}Du - u^{\top}Wu \\&= \sum_{1 \leq i \leq n} D_{i,i}u_i^2 - \sum_{1 \leq i, \ell \leq n} W_{i,\ell}u_iu_{\ell} \\&= \frac{1}{2} \left(\sum_{1 \leq i \leq n} D_{i,i}u_i^2 + \sum_{1 \leq \ell \leq n} D_{\ell,\ell}u_{\ell}^2 - 2 \sum_{1 \leq i, \ell \leq n} W_{i,\ell}u_iu_{\ell} \right) \\&= \frac{1}{2} \left(\sum_{1 \leq i, \ell \leq n} W_{i,\ell}u_i^2 + \sum_{1 \leq i, \ell \leq n} W_{i,\ell}u_{\ell}^2 - 2 \sum_{1 \leq i, \ell \leq n} W_{i,\ell}u_iu_{\ell} \right) \quad (W_{i,\ell} \text{ symm.}) \\&= \frac{1}{2} \sum_{1 \leq i, \ell \leq n} W_{i,\ell}(u_i - u_{\ell})^2.\end{aligned}$$

Therefore, for all $j \in [K]$,

$$\begin{aligned}h_j^\top L h_j &= \frac{1}{2} \sum_{1 \leq i, \ell \leq n} W_{i, \ell} (H_{i, j} - H_{\ell, j})^2 \\&= \frac{1}{2} \sum_{\substack{i \in I_j \\ \ell \notin I_j}} \frac{W_{i, \ell}}{|I_j|} + \frac{1}{2} \sum_{\substack{i \notin I_j \\ \ell \in I_j}} \frac{W_{i, \ell}}{|I_j|} \\&= \frac{1}{|I_j|} \sum_{\substack{i \in I_j \\ \ell \notin I_j}} W_{i, \ell},\end{aligned}$$

since $H_{i, j} - H_{\ell, j}$ is nonzero only if $i \in I_j$ and $\ell \notin I_j$ or the other way around.

Gathering everything, we have:

$$\text{tr}(H^T L H) = \sum_{j=1}^K h_j^T L h_j = \sum_{j=1}^K \frac{1}{|I_j|} \sum_{\substack{i \in I_j \\ \ell \notin I_j}} W_{i,\ell} = \hat{D}_r(C_1, \dots, C_K).$$

one-hot-encoding

- Up to normalization, H represents the **one-hot-encoding**.
- For example, for $K = 3$, if we reorganize the sample (X_1, \dots, X_n) such that \hat{C}_1 appears first, then \hat{C}_2 and so on, we get

$$H = \begin{pmatrix} \frac{1}{|\hat{C}_1|} & 0 & 0 \\ \vdots & \vdots & \vdots \\ \frac{1}{|\hat{C}_1|} & 0 & 0 \\ 0 & \frac{1}{|\hat{C}_2|} & 0 \\ \vdots & \vdots & \vdots \\ 0 & \frac{1}{|\hat{C}_2|} & 0 \\ 0 & 0 & \frac{1}{|\hat{C}_3|} \\ \vdots & \vdots & \vdots \\ 0 & 0 & \frac{1}{|\hat{C}_3|} \end{pmatrix}.$$

Ratio-cut problem

The ratio cut problem

$$\min_{(\hat{C}_1, \dots, \hat{C}_K) \in \mathcal{P}(\{X_1, \dots, X_n\})} \sum_{k=1}^K \frac{1}{|I_k|} \sum_{\substack{i \in I_k \\ \ell \notin I_k}} W_{i,\ell}$$

is equivalent to

$$\begin{aligned} & \min_{H \in \mathbb{R}^{n \times K}} \operatorname{tr}(H^T L H) \\ & \text{s.t.} \quad \begin{cases} H^T H = I \\ \forall j \in [K], \forall i \in [n] : H_{ij} \in \left\{ 0, \frac{1}{\sqrt{|I_j|}} \right\}. \end{cases} \end{aligned}$$

Relaxation for ratio-cut

Problems:

- this is an **integer programming** problem which we may not be able to solve efficiently.
- the values $(|I_1|, \dots, |I_K|)$ are not known in advance.

Idea: discard the last constraint

$$\begin{aligned} \min_{H \in \mathbb{R}^{n \times K}} \quad & \text{tr}(H^\top L H) \\ \text{s.t.} \quad & H^\top H = I. \end{aligned}$$

Relaxation for ratio-cut

Problems:

- this is an **integer programming** problem which we may not be able to solve efficiently.
- the values $(|I_1|, \dots, |I_K|)$ are not known in advance.

Idea: discard the last constraint

$$\begin{aligned} \min_{H \in \mathbb{R}^{n \times K}} \quad & \text{tr}(H^\top L H) \\ \text{s.t.} \quad & H^\top H = I. \end{aligned}$$

- Solved by the matrix H for which the columns are the minor eigenvectors of L .
- Resulting algorithm: **unnormalized spectral clustering**.
 - maps data (X_1, \dots, X_n) to rows of the K minor eigenvectors of L
 - then performs a vanilla K -means

Unnormalized spectral clustering

Unnormalized spectral clustering

Require: $W \in \mathbb{R}^{n \times n}$ (adjacency matrix).

$L \leftarrow$ Laplacian of W

$H \leftarrow K$ minor eigenvectors of L as columns

$Y_i \leftarrow i^{\text{th}}$ row of H (for all $i \in [n]$) $\{Y_i \in \mathbb{R}^K\}$

$(\hat{C}_1, \dots, \hat{C}_K) \leftarrow$ output of K -means algorithm based on (Y_1, \dots, Y_n)

Ensure: $(\hat{C}_1, \dots, \hat{C}_K)$.

Idea

- 1 Dimension reduction
- 2 K-means

Reformulation for normalized cut

Proposition

Let W and L be respectively the adjacency matrix and the Laplacian of the similarity graph of (X_1, \dots, X_n) . For any positive integer k and for all partitioning (C_1, \dots, C_K) of (X_1, \dots, X_n) , we have

$$\hat{D}_n(C_1, \dots, C_K) = \text{tr}(H^\top L H),$$

where $H = \left(\frac{1}{\sqrt{\text{vol}(C_j)}} \mathbb{1}_{i \in C_j} \right)_{\substack{1 \leq i \leq n \\ 1 \leq j \leq K}}$.

In addition, the columns of $D^{\frac{1}{2}} H$ are orthonormal to each other ($H^\top D H = I$).

Reformulation for normalized cut

Proposition

Let W and L be respectively the adjacency matrix and the Laplacian of the similarity graph of (X_1, \dots, X_n) . For any positive integer k and for all partitioning (C_1, \dots, C_K) of (X_1, \dots, X_n) , we have

$$\hat{D}_n(C_1, \dots, C_K) = \text{tr}(H^\top L H),$$

where $H = \left(\frac{1}{\sqrt{\text{vol}(C_j)}} \mathbb{1}_{i \in C_j} \right)_{\substack{1 \leq i \leq n \\ 1 \leq j \leq K}}$.

In addition, the columns of $D^{\frac{1}{2}} H$ are orthonormal to each other ($H^\top D H = I$).

Proof: similar to the one of the previous Proposition except that we have for all $j \in [K]$,

$$h_j^\top L h_j = \frac{1}{\text{vol}(\hat{C}_j)} \sum_{\substack{i \in C_j \\ \ell \notin C_j}} W_{i,\ell}.$$

Relaxation for normalized-cut problem

The normalized cut problem

$$\min_{(\hat{C}_1, \dots, \hat{C}_K) \in \mathcal{P}(\{X_1, \dots, X_n\})} \sum_{j=1}^K \frac{1}{\text{vol}(\hat{C}_j)} \sum_{\substack{i \in I_j \\ \ell \notin I_j}} W_{i,\ell}$$

is equivalent to

$$\begin{aligned} \min_{H \in \mathbb{R}^{n \times K}} \quad & \text{tr}(H^T L H) \\ \text{s.t.} \quad & H^T D H = I \\ & \forall j \in [K], \forall i \in [n]: H_{i,j} \in \left\{ 0, \frac{1}{\sqrt{\text{vol}(\hat{C}_j)}} \right\}, \end{aligned}$$

and can be relaxed to

$$\min_{H \in \mathbb{R}^{n \times K}} \text{tr}(H^T L H) \tag{1}$$

$$\text{s.t. } H^T D H = I. \tag{2}$$

Relaxation for normalized-cut problem

$$\min_{H \in \mathbb{R}^{n \times k}} \operatorname{tr}(H^T L H) \quad (3)$$

$$\text{s.t. } H^T D H = I. \quad (4)$$

can be reformulated

$$\begin{aligned} \min_{H \in \mathbb{R}^{n \times k}} \operatorname{tr}(U^T L_s U) \\ \text{s.t. } H = D^{-\frac{1}{2}} U \\ U^T U = I, \end{aligned}$$

where $L_s = D^{-\frac{1}{2}} L D^{-\frac{1}{2}}$.

Relaxation for normalized-cut problem

$$\min_{H \in \mathbb{R}^{n \times k}} \text{tr}(H^T L H) \quad (3)$$

$$\text{s.t. } H^T D H = I. \quad (4)$$

can be reformulated

$$\begin{aligned} \min_{H \in \mathbb{R}^{n \times k}} \text{tr}(U^T L_s U) \\ \text{s.t. } \quad H = D^{-\frac{1}{2}} U \\ \quad \quad U^T U = I, \end{aligned}$$

where $L_s = D^{-\frac{1}{2}} L D^{-\frac{1}{2}}$.

- Solved by U for which the columns are minor eigenvectors of L_s
- corr. to H for which columns are minor eigenvectors of $L_w = D^{-1} L$
- resulting algorithm: **normalized spectral clustering**.

Normalized spectral clustering (with L_w)

Require: $W \in \mathbb{R}^{n \times n}$ (adjacency matrix).

$L_w \leftarrow$ Laplacian of W

$H \leftarrow K$ minor eigenvectors of L_w as columns {similar to the generalized eigenproblem $Lu = \lambda Du$ }

$Y_i \leftarrow i^{\text{th}}$ row of H (for all $i \in [n]$) $\{Y_i \in \mathbb{R}^K\}$

$(\hat{C}_1, \dots, \hat{C}_K) \leftarrow$ output of k-means algorithm based on (Y_1, \dots, Y_n)

Ensure: $(\hat{C}_1, \dots, \hat{C}_K)$.

Remark: $\lambda \in \mathbb{R}_+$ is eigenvalue of L_w with eigenvector u iff λ and u solve the generalized eigenvalue problem $Lu = \lambda Du$.

Ratio cut vs. normalized cut

- Both objective functions: points separated into different clusters should be dissimilar
- Both take into account the *importance* of the clusters (by their size or their volume)

Different behavior on cluster importance:

- it is easy to see that, for all $j \in [K]$:

$$\sum_{\substack{i \in I_j \\ \ell \in I_j}} W_{i,\ell} = \text{vol}(\hat{C}_j) - \sum_{\substack{i \in I_j \\ \ell \notin I_j}} W_{i,\ell}.$$

In other words, the intra-cluster similarity is maximized as soon as the volume of the cluster is maximized and the cut with the rest

of the vertices is minimized; which is what is achieved by normalized cut minimization.

- On the other hand, the size $|\hat{C}_j|$ of a cluster is not necessarily related to the intra-cluster similarity.

↪ **normalized cut minimization** addresses both parts of clustering

- (+) Normalized spectral clustering: L_w behaves as expected when $n \rightarrow \infty$
- (-) L can lead to completely unreliable results, even for small sample size Von Luxburg, 2007

- (+) Normalized spectral clustering: L_w behaves as expected when $n \rightarrow \infty$
- (-) L can lead to completely unreliable results, even for small sample size Von Luxburg, 2007

There exists another spectral algo:

Normalized spectral clustering (with L_s)

Require: $W \in \mathbb{R}^{n \times n}$ (adjacency matrix).

$L_s \leftarrow$ Laplacian of W

$H \leftarrow K$ minor eigenvectors of L_s as columns

$Y_i \leftarrow i^{\text{th}}$ row of H normalized to 1 (for all $i \in [n]$) $\{Y_i \in \mathbb{R}^K,$

$\sum_{j=1}^K (Y_i)_j^2 = 1\}$

$(\hat{C}_1, \dots, \hat{C}_K) \leftarrow$ output of k-means algorithm based on (Y_1, \dots, Y_n)

Ensure: $(\hat{C}_1, \dots, \hat{C}_K)$.

- There is no theoretical guarantees concerning the “quality” of these two relaxations.
- Second, there exist many other relaxations: relying on semidefinite programming
- Spectral relaxations are not appealing for the quality of the solutions they provide but for the simplicity of the problem in which they result (standard linear algebra – eigenvalue – problems).

Properties of graph Laplacian

- $W \in \mathbb{R}_+^{n \times n}$ a symmetric adjacency matrix
- $D \in \mathbb{R}^{n \times n}$ its degree matrix

So far, we have seen three Laplacians:

Definition

Unnormalized Laplacian: $L = D - W$;

Normalized Laplacian 1: $L_s = D^{-\frac{1}{2}} L D^{-\frac{1}{2}} = I - D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$;

Normalized Laplacian 2: $L_w = D^{-1} L = I - D^{-1} W$.

Indexed by s and w : they are respectively symmetrically normalized by $D^{-\frac{1}{2}}$ (on left and right) and whitened by D .

Properties of graph Laplacian

- Bridging gap between graph cut and eigenvalue dec.
- Discover that 0 is an eigenvalue of L and L_w with eigenvector $\mathbb{1}$.

Proposition

1 $\forall u \in \mathbb{R}^n$:

$$u^\top L u = \frac{1}{2} \sum_{1 \leq i, \ell \leq n} W_{i, \ell} (u_i - u_\ell)^2$$

$$u^\top L_s u = \frac{1}{2} \sum_{1 \leq i, \ell \leq n} W_{i, \ell} \left(\frac{u_i}{\sqrt{D_{i,i}}} - \frac{u_\ell}{\sqrt{D_{\ell, \ell}}} \right)^2.$$

- 2 0 is eigenvalue of L and L_w with eigenvector $\mathbb{1}$. 0 is eigenvalue of L_s with eigenvector $D^{\frac{1}{2}} \mathbb{1}$.
- 3 $\lambda \in \mathbb{R}_+$ is eigenvalue of L_w with eigenvector u if and only if λ is eigenvalue of L_s with eigenvector $D^{\frac{1}{2}} u$.
- 4 L , L_s and L_w are symmetric SDP matrices.

Properties of graph Laplacian

Proof :

- 1 See above.
- 2 Obvious.
- 3 $\lambda u = L_w u \iff \lambda u = D^{-1} L u \iff \lambda(D^{\frac{1}{2}} u) = D^{-\frac{1}{2}} L D^{-\frac{1}{2}} (D^{\frac{1}{2}} u)$.
- 4 Symmetry comes from symmetry of W . SDPness comes from Point 1 and Point 3.

Properties of graph Laplacian

Proof :

- 1 See above.
- 2 Obvious.
- 3 $\lambda u = L_w u \iff \lambda u = D^{-1} L u \iff \lambda(D^{\frac{1}{2}} u) = D^{-\frac{1}{2}} L D^{-\frac{1}{2}} (D^{\frac{1}{2}} u)$.
- 4 Symmetry comes from symmetry of W . SDPness comes from Point 1 and Point 3.

Proposition

Let G be an undirected graph with non-negative weights.

- the multiplicities of the eigenvalue 0 of L , L_s and L_w are the same and equal the number k of connected components (A_1, \dots, A_k) in G .
- the eigenspace of 0 for both L and L_w is spanned by $\{\mathbb{1}_{A_1}, \dots, \mathbb{1}_{A_k}\}$ and the eigenspace of 0 for L_s is spanned by $\{D^{-\frac{1}{2}} \mathbb{1}_{A_1}, \dots, D^{-\frac{1}{2}} \mathbb{1}_{A_k}\}$.

Hierarchical clustering

Drawback of K-means

Lack of hierarchy in clusters (i.e. decreasing K does not lead to merging clusters)

↪ Hierarchical clustering to address this issue

Drawback of K-means

Lack of hierarchy in clusters (i.e. decreasing K does not lead to merging clusters)

↪ Hierarchical clustering to address this issue

How?

- Introduce very simple methods based on measuring the similarity (or linkage) between clusters.

Drawback of K-means

Lack of hierarchy in clusters (i.e. decreasing K does not lead to merging clusters)

↪ Hierarchical clustering to address this issue

How?

- Introduce very simple methods based on measuring the similarity (or linkage) between clusters.

What?

- Focus on **agglomerative** approaches (which is based on merging clusters) ↪ bottom-up
- We put **divisive** ones aside (based on splitting clusters) ↪ top-down

Linkage-based methods are probably the simplest and most intuitive paradigm of clustering.

Agglomerative version

- start from the partitioning of training set (X_1, \dots, X_n) in which each cluster is a unit set $\{X_i\}$ (for $i \in [n]$)
- merge successively the *closest* clusters

Straightforwardly,

- the number of clusters decreases at each iteration
- clusters are nested
- each cluster \hat{C}^t at iteration t is either the same $\hat{C}^t = \hat{C}^{t-1}$ or the union of two previous clusters $\hat{C}^t = \hat{C}_1^{t-1} \cup \hat{C}_2^{t-1}$.

Two parameters need to be defined in such a procedure:

- the (dis)similarity (or linkage) between two clusters
- the merging stopping rule

Some dissimilarities

Let $d : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}_+$ be a dissimilarity and consider two subsets A and B of (X_1, \dots, X_n) .

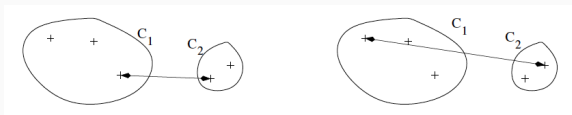
Here are some cluster dissimilarities $D : P(\{X_1, \dots, X_n\})^2 \rightarrow \mathbb{R}_+$.

Single linkage

$$D(A, B) = \min_{x \in A, y \in B} d(x, y).$$

Complete linkage

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



Some dissimilarities

Average linkage

$$D(A, B) = \frac{1}{|A||B|} \sum_{x \in A, y \in B} d(x, y).$$

Ward's minimum variance

Given the intraclass inertia for a generic subset $C \subset (X_1, \dots, X_n)$:

$$I(C) = \sum_{x \in C} d(x, m_C)^2,$$

where $m_C = \frac{1}{|C|} \sum_{y \in C} y$, the cluster distance in Ward's method is

$$D(A, B) = I(A \cup B) - I(A) - I(B),$$

which is the increase of intraclass inertia when merging A and B .

For the Euclidean distance,

$$D(A, B) = \frac{|A||B|}{|A| + |B|} \|m_A - m_B\|^2.$$

Since Ward's method merges clusters by minimizing the increase in the total intraclass inertia, it is very similar k-means but greedy procedure

Linkage methods can be used with a variety of distances (or affinities), in particular:

- Euclidean distance (or l2);
- Manhattan distance (or Cityblock, or l1);
- cosine distance;
- any precomputed affinity matrix.

Stopping criterion

- If the agglomerative procedure runs until the end, all points share the same large cluster.
- The resulting sequence of partitioning can be represented as a **tree**, called a **dendrogram**
 - the root = unique cluster that gathers all points (the final cluster)
 - the leaves = the unit set clusters (algorithm initialization)

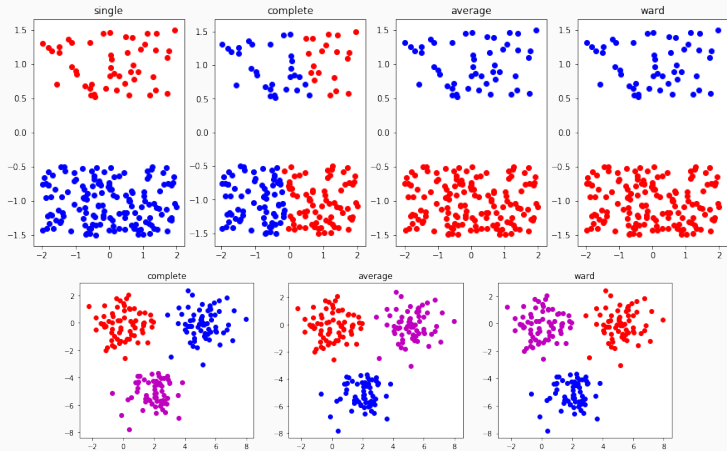
Stopping rules

- a fixed number of clusters
- a distance upper bound \bar{D} (or alternatively a *scaled distance upper bound* $\alpha \in \mathbb{R}_+$ such that $\bar{D} = \alpha \max_{1 \leq i, j \leq n} d(X_i, X_j)$ for single, complete and average linkages)

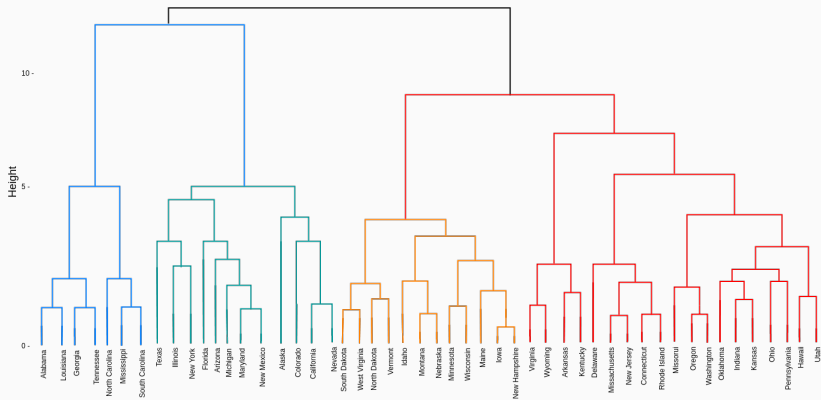
Complexity

- $O(n^3)$ if no restriction on the merging possibilities
- $O(n^2)$ if only a bounded number of merging is possible for a given cluster

Agglomerative clustering on some pictures



Cluster Dendrogram



Concluding remarks

K-Means and GMM don't work for "embedded" cluster structures

