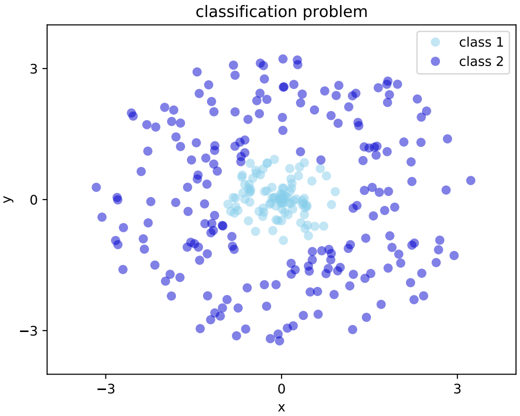
Fondamentaux théoriques du machine learning



Overview

Feature maps

Clustering

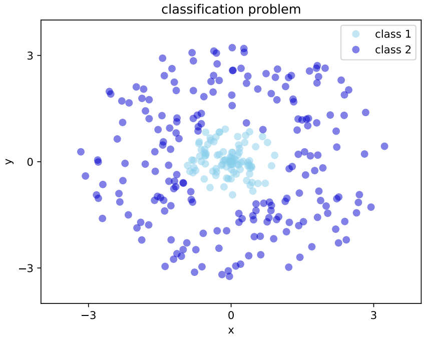
Feature maps

Often, we do not work with the $x_i \in \mathcal{X}$, but with **representations** $\phi(x_i)$, with $\phi : \mathcal{X} \rightarrow \mathbb{R}^d$. Possible motivations :

- $\mathcal{X}$ need not be a vector space.
- $\phi(x)$ can provide more useful **features** for the considered problem (classification, regression).
- The prediction function is then allowed to depend **non-linearly** on x. (But there can still be linear dependence in $\phi(x)$, this will often be the case).

Feature map

Exercice 1 : Finding a feature map
What feature map could be used to be able to **linearly separate** these two classes ?



Application to OLS and ridge

Instead of

$$X = \begin{pmatrix} x_1^T \\ \dots \\ x_i^T \\ \dots \\ x_n^T \end{pmatrix} = \begin{pmatrix} x_{11}, \dots, x_{1j}, \dots x_{1d} \\ \dots \\ x_{i1}, \dots, x_{ij}, \dots x_{id} \\ \dots \\ x_{n1}, \dots, x_{nj}, \dots x_{nd} \end{pmatrix}$$

The design matrix is

$$\phi = \begin{pmatrix} \phi(x_1)^T \\ \dots \\ \phi(x_i)^T \\ \dots \\ \phi(x_n)^T \end{pmatrix}$$

Application to OLS and ridge

The statistical results are maintained, as a function of d, the dimension of $\phi(x)$.

Linear estimator

We often encounter estimators of the form

$$f(x) = h(\langle \phi(x), \theta \rangle) = h(\phi(x)^T \theta) \tag{1}$$

- They are often called "linear models"
- Being linear in θ is not the same as being linear in x.

Linear estimator

We often encounter estimators of the form

$$f(x) = h(\langle \phi(x), \theta \rangle) = h(\phi(x)^T \theta) \tag{2}$$

- regression : $h = Id$
- classification : $h = \text{sign}$.

Linear estimator

Interpretation of a linear model as a vote, in the case of classification.

$$f(x) = h(\langle \phi(x), \theta \rangle) = h(\phi(x)^T \theta) \tag{3}$$

Kernel methods

The topic of feature maps is very rich and important in machine learning

- **kernel methods** : ϕ is **chosen**. Many famous choices are available (gaussian kernels, polynomial kernels, etc).
- **neural networks** : ϕ is **learned**.

We will have dedicated exercises on both these methods.

Unsupervised learning

From a number of samples x_i, you want to retrieve information on their structure : **modelisation**. The three main unsupervised learning problems are :

- clustering
- density estimation
- dimensionality reduction

Clustering

Clustering consists in partitioning the data. $\forall i, x_i \in \mathcal{X}^n$.

$$D_n = \{(x_i)_{i \in [1,\dots,n]}\} \tag{4}$$

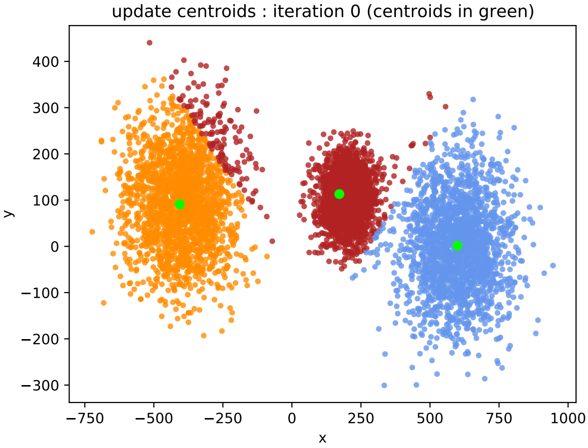
A **partition** is a set of K subsets $A_k \subset D_n$, such that

- $$\cup_{k \in [1,\dots,K]} A_k = D_n \tag{5}$$
- $$\forall k \neq k', A_k \cap A_{k'} = \emptyset \tag{6}$$

Partitions

- **Example 1 :** A is the set of even integers, B the set ot odd integers. Is (A, B) a partition of $\mathbb{N}$?
- **Example 2 :** C is the set of multiples of 2, D the set of multiples of 3. Is (C, D) a partition of $\mathbb{N}$?

Example : partition of data



Applications of clustering

Example applications :

- spam filtering [Sharma and Rastogi, 2014,]
- fake news identification [Hosseinimotlagh and Papalexakis, 2018,]
- marketing and sales
- document analysis [Zhao and Karypis, 2002,]
- traffic classificaiton [Woo et al., 2007,]

Some of these applications can be considered to be semi-supervised learning.

Vector quantization

Vector quantization consists in computing **prototypes** $\Omega = (\omega_k)_{k \in [1,\dots,K]} \in \mathcal{X}^K$ that represent the data well.
This implies that a **metric** is defined on $\mathcal{X}$.
Most often, this is interesting / useful if $K << n$.

Voronoï subsets

We assume a loss (we can think of it as a distance here) L is defined on $\mathcal{X}$. The **Voronoï subset** of ω is defined as

$$V(\omega) = \{x \in D_n, \underset{\omega' \in \Omega}{\arg\min}\, L(\omega', x) = \omega\} \tag{7}$$

- We assume that arg min returns one single element.
- The Voronoï subsets form a partition of D_n.

Distortion

To measure the quality of a Voronoï partition, we introduce the **distortion** $R(\Omega)$.
For each x, we note $h_\Omega(x) = \arg\min_{\omega' \in \Omega} L(\omega', x)$.

$$R(\Omega) = \frac{1}{n} \sum_{i=1}^{n} L(x_i, h_\Omega(x_i)) \tag{8}$$

Distortion

For each x, we note $h_\Omega(x) = \arg\min_{\omega' \in \Omega} L(\omega', x)$.

$$\begin{aligned} R(\Omega) &= \frac{1}{n} \sum_{i=1}^{n} L(x_i, h_\Omega(x_i)) \\ &= \frac{1}{n} \sum_{\omega \in \Omega} \sum_{x \in V(\omega)} L(x_i, h_\Omega(x_i)) \\ &= \frac{1}{n} \sum_{\omega \in \Omega} V_\Omega(\omega) \end{aligned} \tag{9}$$

with

$$V_\Omega(\omega) = \sum_{x \in V(\omega)} L(x_i, h_\Omega(x_i)) \tag{10}$$

Minimum of distortion

We want to find the prototypes for which the distortion is **minimal**.

- The set of prototypes minimizing distorsion might not be unique.
- We need to tune K (number of prototypes).

Vector quantization techniques

- K-means
- Growing neural gas (GNG)
- Self-organizing maps

K-means clustering

- $\mathcal{X} = \mathbb{R}^d$.
- $L(x, y) = ||x - y||^2$.

Objective function

With

- $\Omega = \{\omega_1, \dots, \omega_K\} \in \mathbb{R}^{K,d}$.
- $z_i^k = 1$ if x_i is assigned to ω_k, $z_i^k = 0$ otherwise. $z = (z_i^k) \in \mathbb{R}^{n,K}$.

we define the objective function

$$J(\Omega, z) = \sum_{i=1}^{n} \sum_{k=1}^{K} z_i^k ||x_i - \omega_k||^2 \tag{11}$$

It is also called the **inertia**.

K-means algorithm

Result: $\Omega \in \mathbb{R}^{K,d}$
$\Omega \leftarrow$ Random initialization;
$z = M$ where M is the $n \times K$ matrix with 0's;
while *Convergence criteria is not satisfied* **do**
    a] Greedily minimize J with respect to z;
    b] Minimize J with respect to Ω;
end
return Ω

Algorithm 1: K-means (Loyd algorithm)

Stopping criterion

To stop the algorithm, the norm of the difference between Ω_t and Ω_{t+1} must be smaller than a given tolerance. (e.g. $1e^{-4}$). Here it is a norm between matrix (Frobenius norm) :

$$||A||_F = \sqrt{\sum_{i=1}^{n} A_{ij}^2} \tag{12}$$

Minimization

We focus on step b]. How can we minimize J with respect to Ω ?

Minimization

Exercice 2 : **Convexity :**
Show that $J(\Omega, z)$ is convex with respect to Ω.

$$J(\Omega, z) = \sum_{i=1}^{n} \sum_{k=1}^{K} z_i^k ||x_i - \omega_k||^2 \tag{13}$$

Hence, to minimize J with respect to Ω, we just need to cancel the gradient.

Minimization

Exercice 3 : **Gradient :**
Compute the gradient of $J(\Omega, z)$ with respect to Ω and deduce the minimizer Ω^*.

- z is fixed
- we can see Ω has a vector of $\mathbb{R}^{Kd}$.

Convexity

Is $J(\Omega, z)$ convex in z ?

Convexity

Is $J(\Omega, z)$ convex in z ?
No, as z is not even defined on a convex set, so convexity can not be defined anyways !
Hence, the function might have **local minima**.

Suboptimal clustering

Exercice 3 : **Local minima** : propose a setting (dataset, initialization) for which the algorithm outputs a bad set of centroids.

Suboptimal clustering

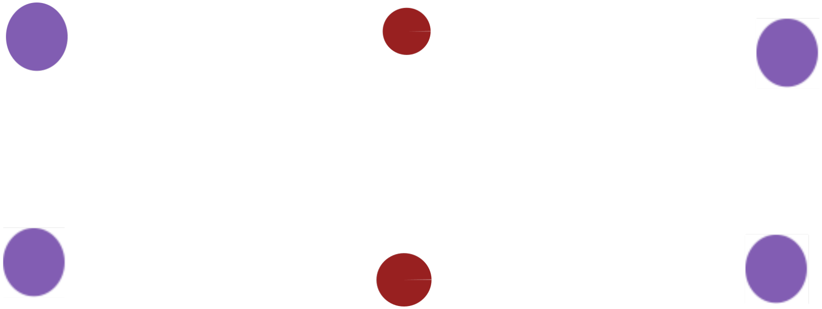


Figure – Initialization of centroids in red.

Random initialization

Hence, the result of K-means strongly depends on the initial position of the centroids.
A common approach is to restart the algorithm several times and select the result with lowest inertia.

Drawbacks of inertia minimization

K-means is based on the minimization of the inertia and hence on the euclidean distance. **However**, in some contexts, the euclidean distance is not the adapted metric.
`https://scikit-learn.org/stable/modules/clustering.html`

References I

Hosseinimotlagh, S. and Papalexakis, E. E. (2018).
Unsupervised content-based identification of fake news articles with tensor decomposition ensembles.
Proceedings of the WSDM MIS2 : Misinformation and Misbehavior Mining on the Web Workshop, pages 1–8.

Sharma, A. and Rastogi, V. (2014).
Spam Filtering using K mean Clustering with Local Feature Selection Classifier.
International Journal of Computer Applications, 108(10) :35–39.

References II

Woo, D. M., Park, D. C., Song, Y. S., Nguyen, Q. D., and Tran, Q. D. N. (2007).
Terrain classification using clustering algorithms.
Proceedings - Third International Conference on Natural Computation, ICNC 2007, 1 :315–319.

Zhao, Y. and Karypis, G. (2002).
Evaluation of hierarchical clustering algorithms for document datasets.
International Conference on Information and Knowledge Management, Proceedings, (August 2002) :515–524.