Introduction à l'apprentissage automatique - Polytech
Unsupervised learning

Caio Corro, Guillaume Wisniewski

Université Paris-Saclay, CNRS, Laboratoire Interdisciplinaire des Sciences du Numérique, 91400, Orsay, France

Context

Supervised Learning

- X = set of observations (generally $X = \mathbb{R}^d$)
- Y = set of labels
- functional dependency f between X and Y
- oracle can label each example x
- supervised learning : infer f knowing a finite sample of labeled data

Unsupervised Learning

Definition

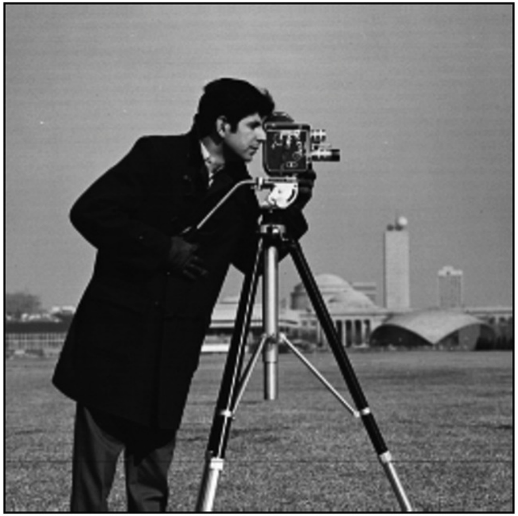
unsupervised learning = no information about the output, *i.e.* no label $y \in Y$

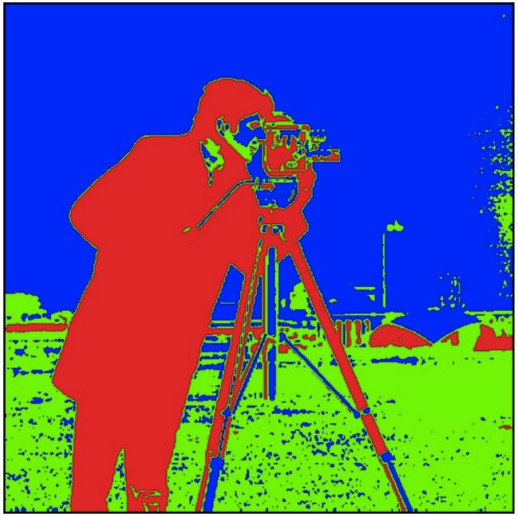
- labeling is too expensive
- datasets/tasks that change over time (e.g. topic detection in newsfeed $\Rightarrow$ new classes appear all the time)
- exploratory data analysis : summarize the main characteristics of a dataset

What we have access to ?

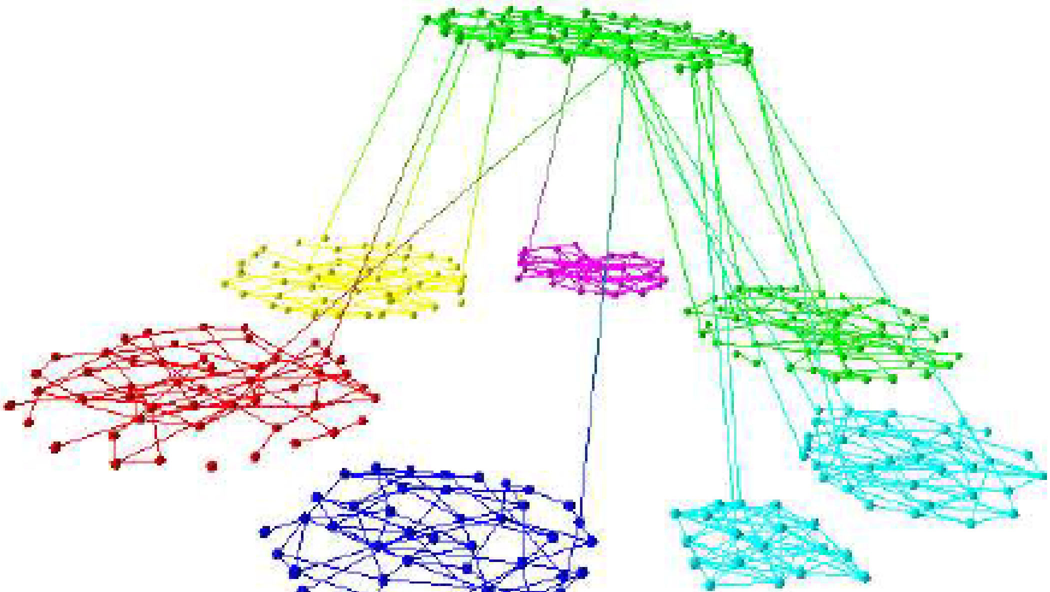
- a dataset $D = \{ \boldsymbol{x}^{(i)} \}_{i=1}^{n}$
- (maybe) information on the number of classes k
- the parametric form of the probability distribution of each class

Example n° 1 : Image segmentation

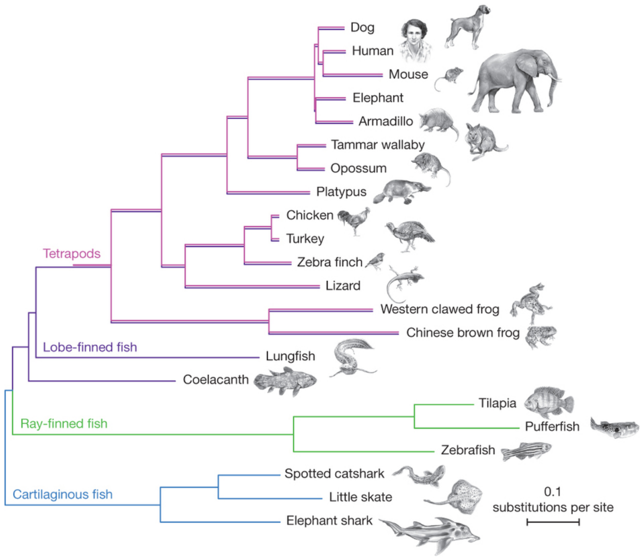




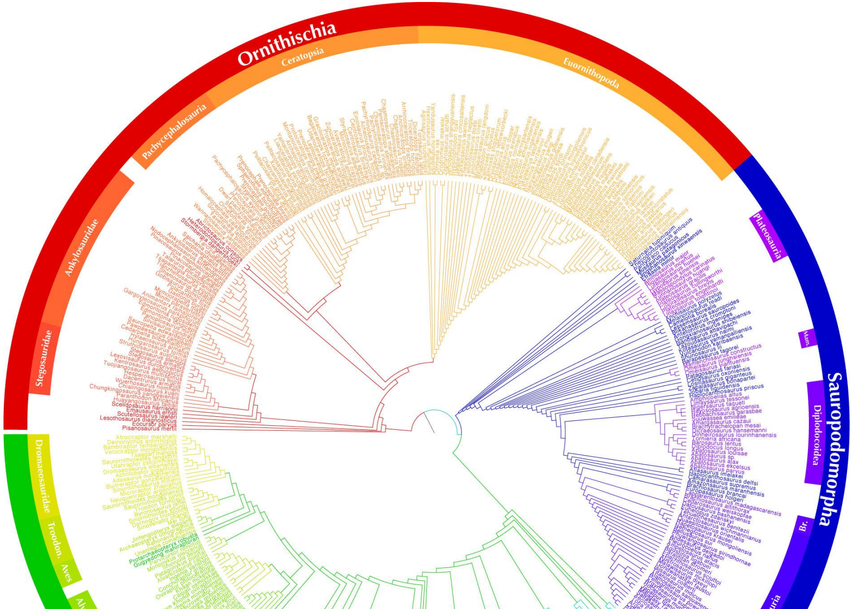
Example n° 2 : community identification



Example n° 3 : Phylogenetic Tree (toy example)

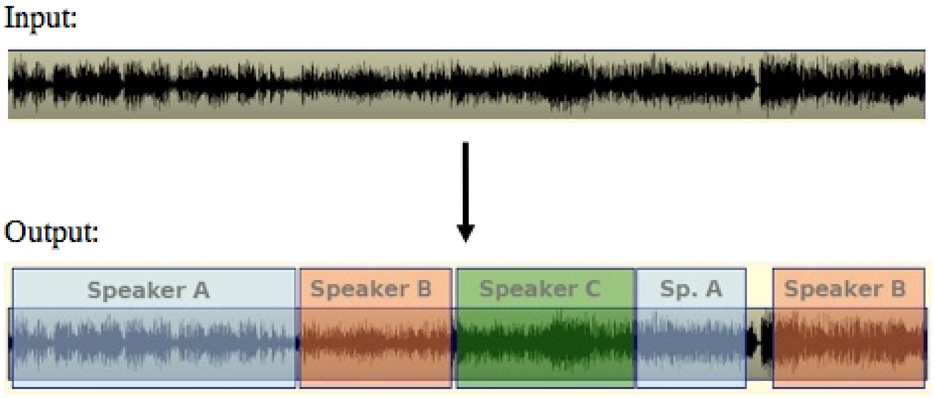


Example n° 3 : Phylogenetic Tree (true example)



Speaker diarization

Who spoke when ?



Example n° 5 : ad targeting

https://medium.com/@jeffgould/the-natural-history-of-gmail-data-mining-be115d196b10

« The Natural History of Gmail Data Mining »

- Texarkana case : Google was compelled to reveal many information about the data it collects about users to select the ad it displays
- many information are collected about users :
  - email content (e.g. order confirmation)
  - websites that have been visited
  - user profile
- Google is able to complete them (e.g. zip code $\Rightarrow$ average income) easy when you have access to many user information
- users are gathered/clustered into millions of « buckets »
- Gmail isn't really about email – it's a gigantic profiling machine

Task definition

Definition

Given two observations $\boldsymbol{x}, \boldsymbol{x}'$ decide whether they belong to same class / cluster or not

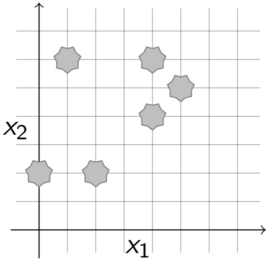
Differences

- no ‘good’ answer
- no guarantee that a solution exists
- quality of a cluster = arbitrary / subjective
- no loss function

Clustering Methods

Intuition

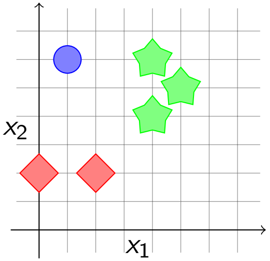
- examples can be seen as vectors in an Euclidean space



- vectors that are ‘close’ should have the ’same behavior’
- ‘gather’ similar vectors together

Intuition

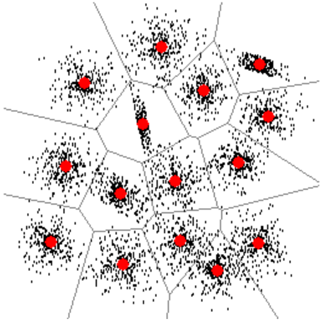
- examples can be seen as vectors in an Euclidean space



- vectors that are ‘close’ should have the ’same behavior’
- ‘gather’ similar vectors together

Principles

- clustering : gather similar observations together
  - how to define the similarity between points
  - how to measure the quality of a clustering
- similarity = distance in an Euclidean space $\oplus$ threshold on the distance
  - be careful : distance is sensitive to feature scale

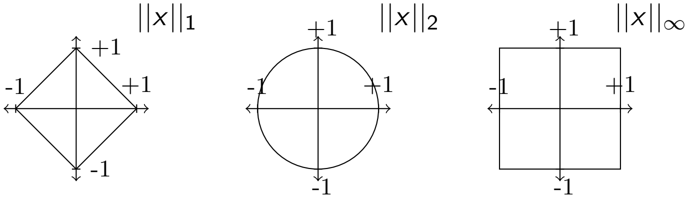


Distances

p-norm in $\mathbb{R}^n$

$$d_p(\boldsymbol{x}, \boldsymbol{x}') = \left[\sum_{i=1}^{d} |x_i - x_i'|^p\right]^{\frac{1}{p}}$$

- $p = 1$ Manhattan distance
- $p = 2$ Euclidean distance
- $p \to \infty$ Chebyshev distance



Evaluating a clustering

Notations

- n observations $D = \left\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(n)}\right\}$
- D clustered in k disjoint sets $D_1, \ldots, D_k$
- $\ell(D_1, \ldots, D_k) =$ quality of $D_1, \ldots, D_k$ (not really a loss in this case)

Minimum variance clustering

- $\boldsymbol{\mu}^{(i)}$ mean of cluster D_i : $\boldsymbol{\mu}^{(i)} = \frac{1}{|D_i|} \sum_{\boldsymbol{x} \in D_i} \boldsymbol{x}$
- sum of distances to the mean within the i-th cluster $r_i = \sum_{\boldsymbol{x} \in D_i} \|\boldsymbol{x} - \boldsymbol{\mu}^{(i)}\|^2$
  $\Rightarrow$ the smaller r_i, the 'denser' the cluster
- final criterion $\ell(D_1, \ldots, D_c) = \sum_{i=1}^{k} \sum_{\boldsymbol{x} \in D_i} \|\boldsymbol{x} - \boldsymbol{\mu}^{(i)}\|^2$

Finding the best partition

- combinatorial problem : $\sim \frac{c^n}{n!}$ different clusterings
- iterative optimization
  - starting from an initial configuration
  - modify this configuration to improve the objective function
  - no guarantee that we reach a global optimum
- in practice : k-means

k-means : notations

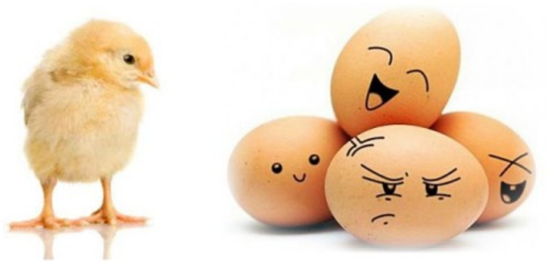
Input

- D : set of vectors in $\mathbb{R}^d$
- k : number of clusters to find
- d : distance / similarity metric

Output

- k vectors of $\mathbb{R}^d$: centers/mean/representer of a cluster
- each observation $\mathbf{x} \in D$ is assigned to a cluster

k-means : principle



- minimal distance classifier
- given the centers : easy to cluster the observations (assign each observation to the closest center)
- if the clustering is known : easy to estimate the centers

  (compute the mean of the observation of the cluster)
- iterate between these two steps

k-means : algorithm

1. choose k centers randomly $\boldsymbol{\mu}^{(1)}, \dots, \boldsymbol{\mu}^{(k)}$
2. assign each observation $\boldsymbol{x} \in D$ to the cluster $\omega(\boldsymbol{x})$ so that

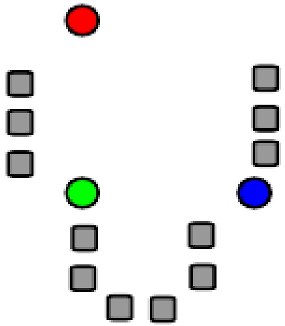
$$\omega(\boldsymbol{x}) = \underset{i \in \{1,\dots,k\}}{\arg\min} \|\boldsymbol{x} - \boldsymbol{\mu}^{(i)}\|^2$$

3. recalculer la moyenne à partir des points associés à la classe

$$\boldsymbol{\mu}^{(i)} = \frac{1}{|\{\boldsymbol{x} \in D | \omega(\boldsymbol{x}) = i\}|} \sum_{\substack{\boldsymbol{x} \in D \\ \text{s.t. } \omega(\boldsymbol{x}) = i}} \boldsymbol{x}$$

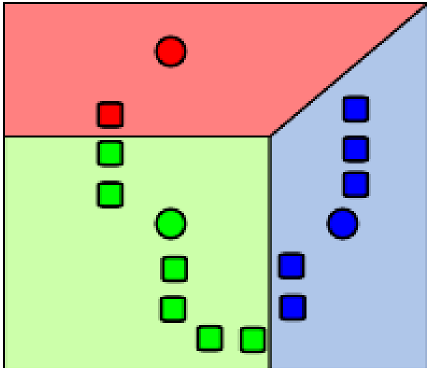
4. iterate steps 2 & 3

k-means picture



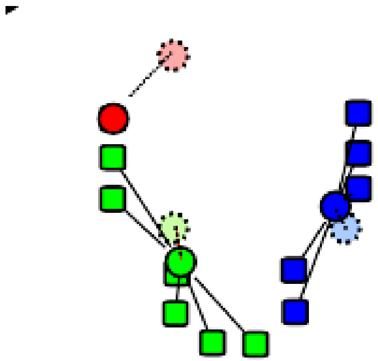
Pictures from Wikipedia

k-means picture



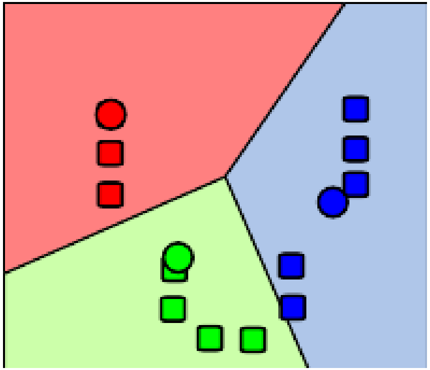
Pictures from Wikipedia

k-means picture



Pictures from Wikipedia

k-means picture



Pictures from Wikipedia

Analysis

k-means : questions

Does it converge ?

- YES ! (sketch of the proof on next slides)

How fast does it converge ?

- in practice : quite quickly ($\simeq$ 20 iterations)
- in theory : smoothed analysis $\rightarrow$ we can prove the fast convergence

Does it converge to the 'right' answer ?

- what is the 'right' answer ?
- local optimum : sensitive to initialization

Convergence

- quality measure :

$$\ell(D_1, ..., D_k) = \sum_{i=1}^{k} \sum_{\boldsymbol{x} \in D_i} \|\boldsymbol{x} - \boldsymbol{\mu}^{(i)}\|^2 \tag{1}$$

- Notice that :
  1. if the old clustering is the same as the new, then the next clustering will again be the same.
  2. if the new clustering is different from the old then the newer one has a lower cost
- only a finite number of assignments $\Rightarrow$ the iteration must eventually enter a cycle.
- The cycle can not have length greater than 1 because otherwise by (2) you would have some clustering which has a lower cost than itself which is impossible. Hence the cycle must have length exactly 1.

Initialization : furthest-first heuristic

- goal : select k initial centers
- algorithm :
  1. pick a random example $\boldsymbol{x} \in D$ and set $\boldsymbol{\mu}^{(1)} = \boldsymbol{x}$
  2. for $i = 2..k$: set $\boldsymbol{\mu}^{(i)}$ to the example m that is as far as possible from all previously selected centers :

$$\boldsymbol{\mu}^{(i)} \in \arg\max_{\boldsymbol{x} \in D} \left(\min_{j<i} \|\boldsymbol{x} - \boldsymbol{\mu}^{(j)}\|^2 \right) \tag{2}$$

- heuristics : intuitive + works well in practice
- no proof !

k-means++

- randomized version of the furthest-first heuristic
- choose a data point at random, with probability proportional to its distance to a center

```
$\boldsymbol{\mu}^{(1)} \leftarrow \boldsymbol{x}$ for a given $\boldsymbol{x} \in D$ chosen uniformly at random
for $i \in \{2, ..., k\}$ do
    // compute distances between each point and its closest center
    $d_m \leftarrow \min_{j \in \{1,...,i-1\}} ||\boldsymbol{x}^{(m)} - \boldsymbol{\mu}^{(k)}||^2, \quad \forall i \in \{1, ..., n\}$
    $\mathbf{p} \leftarrow \frac{1}{\sum_i d_i} \mathbf{d}$
    $m \leftarrow$ random sample from $\mathbf{p}$
    $\boldsymbol{\mu}^{(i)} \leftarrow \boldsymbol{x}^{(m)}$
end for
```

How to choose k ?

- increasing k will always decrease $\ell \Rightarrow$ no learning criteria
- main idea : penalize the model if it becomes too complex (*i.e.* too many clusters)
- other scenario :
  - sometimes we know in advance the number of classes we need to find
  - we can evaluate on a downstream task (*e.g.* clustering can be used to extract features)