ALGORITHMES POUR L'INTELLIGENCE ARTIFICIELLE
Les problèmes de classification

L'intelligence artificielle (IA) fait référence au comportement "intelligent" d'ue machine, par opposition à l'intelligence naturelle. En tant que discipline scientifique elle est née dans les années 1950 et a connu plusieurs vagues d'enthousiasme et de déception.

Depuis le début du XXI siècle le développement de l'IA et son intégration dans la vie courante a conduit à une reconnaissance importante de cette discipline. Parmis les utilisations actuelles de l'IA on peut citer : les moteurs de recherche, les programmes informatiques surpassant les meilleurs joueurs humains d'échecs et de go, la voiture autonome, la reconnaissance vocale, etc.. Une des techniques qui a le plus contribué à faire connâitre l'IA est l'apprentissage automatique (machine learning) dont l'essor a été considérable ces dernières années.

Cela a été rendu possible par le développement de méthodes statistiques et d'algorithmes plus perfectionnés, la multiplication des quantités de données disponibles, l'amélioration des moyens de calcul et l'augmentation de la puissance des ordinateurs.

Dans ce chapitre nous allons considérer conformément au programme deux techniques d'apprentissage automatique appelées k-plus proches voisins et k-moyennes.

I. Vocabulaire et définitions

1) Partition et classification d'un ensemble

On dispose d'un ensemble E de données, par exemple des photos de chats et de chiens, des images de chiffres de 0 à 9, des échantillons de fleurs, etc ... et nous souhaitons construire des algorithmes permettant de regrouper les éléments de E suivant leurs proximité, à savoir :

- tous les chats ensembles et tous les chiens ensembles ;
- tous les chiffres identiques ensembles ;
- toutes les fleurs regoupées selon leur espèce ; ;
- ...

Ce procédé est s'appelle **classifier** un ensemble, c'est à dire qu'on le divise en sous-ensembles (qu'on apelle aussi **classes**) en regroupant tous les éléments selon leur proximité. Naturellement, on suppose qu'on ne peut pas rencontrer d'animal qui soit à la fois un chat et un chien ou encore de chiffre qui soit à la fois un 3 et un 7.

Le concept mathématique adéquat est celui de **partition** :

Définition 1. *Partition d'un ensemble*

Soit E un ensemble non vide. On appelle **partition** de E tout ensemble $\mathscr{F} \subset \mathscr{P}(E)$ ($\mathscr{F}$ est donc un ensemble de sous-ensembles de E) vérifiant :

1. $\forall C \in \mathscr{F},\ C \neq \varnothing$
2. $\forall (C, D) \in \mathscr{F} \times \mathscr{F},\ C \neq D \implies C \cap D = \varnothing$
3. $\bigcup_{C \in \mathscr{F}} C = E$

Remarques et exemples :

- Étant donné une partition $\mathscr{F}$ d'un ensemble E, chaque élément $x \in E$ appartient à un et un seul ensemble $C \in \mathscr{F}$.
- Si $\mathscr{R}$ est une relation d'équivalence sur E alors l'ensemble des classes d'équivalence forme une partition de E.
- Inversement, si $\mathscr{F}$ est une partition de E, la relation binaire sur E définie par :

$$x \mathscr{R} y \iff \exists C \in \mathscr{F},\ x \in C \quad \text{et} \quad y \in C$$

est une relation d'équivalence sur E dont l'ensemble des classes d'équivalences est justement $\mathscr{F}$.

Définition 2. *Classe d'une partition*

Étant donnée une partition $\mathscr{F}$ d'un ensemble E, tout ensemble $C \in \mathscr{F}$ est appelé **classe** de cette partition.

Définition 3.

Classifier un ensemble E consiste à définir une partition $\mathscr{F}$ de E. Cela consiste donc à regrouper tous les éléments de E dans des sous-ensembles non-vides, disjoints deux à deux et dont la réunion est E.

En pratique ce regroupement se fera selon un degré de ressemblance ou de proximité (notion précisée plus loin) des éléments appartenant à un même classe $C \in \mathscr{F}$.

Dans la suite nous travaillerons sur des ensembles E **finis** et nous noterons $|E|$ leur cardinal (nombre d'éléments). Toute partition $\mathscr{F}$ de E est elle aussi finie et son cardinal sera noté p : $|\mathscr{F}| = p$ ($p > 1$). Les différentes classes de $\mathscr{F}$ seront numérotées par un indice j, appelé indice de la classe, allant de 0 à $p-1$. On aura donc :

$$\mathscr{F} = \{ C_0, \ldots, C_{p-1} \}$$

2) Distances

Afin de pouvoir classifier un ensemble E on a besoin de regrouper ses éléments selon leurs proximité ou leur similitude. Cela peut se faire naturellement si on peut définir une **distance** sur E.

Définition 1. *Distance*

Soit E un ensemble non vide. On appelle distance sur E toute application $d : E \times E \longrightarrow \mathbb{R}_+$ vérifiant les trois propriétés suivantes :

1. $\forall (x, y) \in E^2,\ d(x, y) = 0 \iff x = y$
2. $\forall (x, y) \in E^2,\ d(x, y) = d(y, x)$ (symétrie)
3. $\forall (x, y, z) \in E^3,\ d(x, y) \leqslant d(x, z) + d(y, z)$ (inégalité triangulaire)

Exemple :

- Si E est un espace vectoriel normé (e.v.n.), alors en désignant par $\mathcal{N}$ sa norme on peut définir une distance d associée à $\mathcal{N}$ par :

$$\forall (x, y) \in E^2,\ d(x, y) = \mathcal{N}(x - y)$$

Exercice : montrer qu'il s'agit bien d'une distance

- En particulier dans $\mathbb{R}^n$, pour $x = (x_1, \ldots, x_n)$ et $y = (y_1, \ldots, y_n)$ on peut définir les trois distances :

$$d_1(x, y) = \sum_{i=1}^{n} |x_i - y_i|$$
$$d_2(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \quad \text{(distance euclidienne)}$$
$$d_\infty(x, y) = \max_{1 \leqslant i \leqslant n} |x_i - y_i|$$

Prenons l'exemple classique de la classification de l'ensembles des fleurs iris en trois espèces : iris-setosa, iris-versicolor et iris-virginica. Si on représente une fleur donnée par un point (x, y) de $\mathbb{R}^2$ x étant la longueur et y la largeur de ses pétales, on obtient pour un ensemble de 150 iris le schéma ci-dessous :

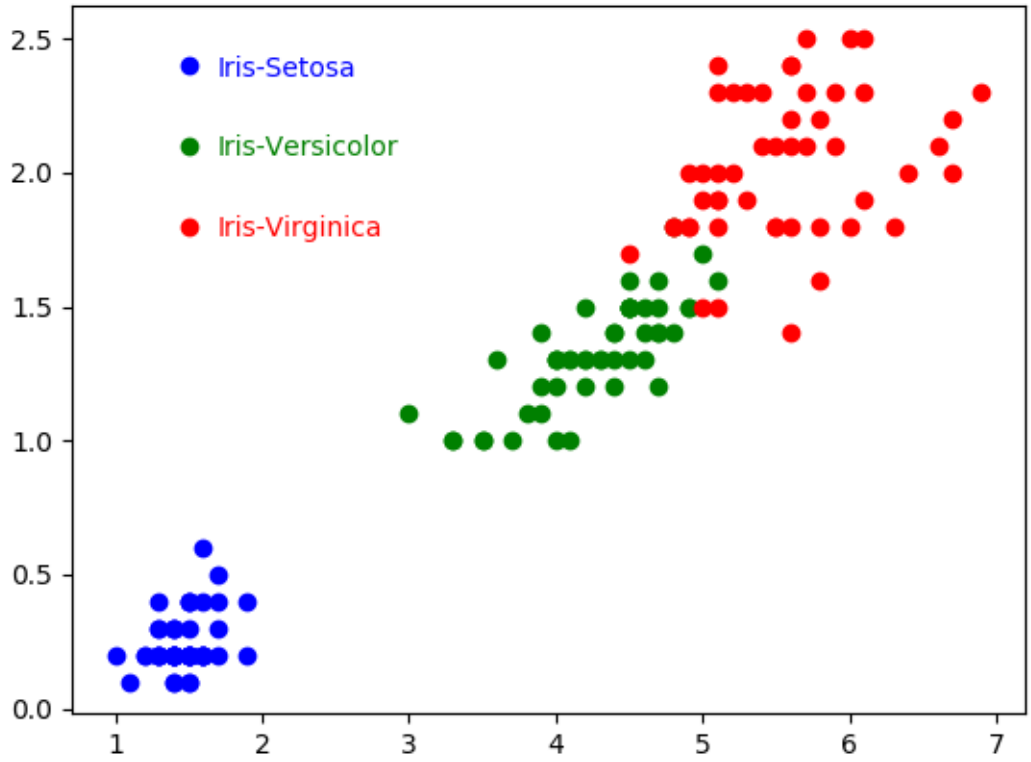


FIGURE 1 – Chaque fleur est représentée par un point dont la couleur correspond à l'espèce.

Les trois espèces ont tendance à créer des amas : il peut donc être pertinent de définir la proximité entre deux fleurs par une des trois distances de $\mathbb{R}^2$.

- **Distance entre deux images** :

Une image est un ensemble de **pixels** qui peuvent être définis comme des zones carrées identiques de dimensions $(a \times a)$. La couleur de chaque pixel est uniforme et l'aspect non

granulaire d'une image n'est dû qu'au fait que la taille des pixels est très petite. La figure2 donne un exemple d'image "binaire" représentant un A. Il n'y a ici que deux couleurs : noir et blanc.

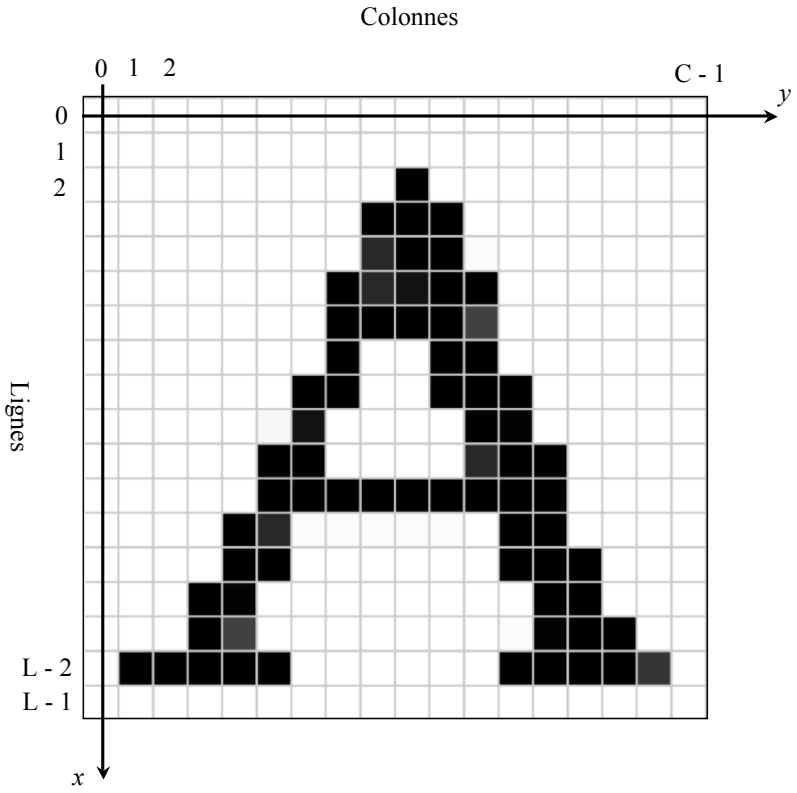


FIGURE 2 – Image de A

Le repérage d'un pixel se fait dans un plan (Oxy), l'axe Ox étant (en général) orienté vers le bas. L'origine O est au centre du pixel situé tout en haut et à gauche et Le centre de chaque pixel est alors repéré par les coordonnées $a \times (i, j)$, i et j étant deux entiers, avec $i \in [\![0, L-1]\!]$ et $j \in [\![0, C-1]\!]$. L est le nombre de lignes de pixels et C le nombre de colonnes.

Encodage d'une image noir et blanc

Dans une image noir et blanc la couleur est encodée par un entier à valeurs dans $[\![0, 255]\!]$: 0 pour un pixel noir, 255 s'il est blanc, toute valeur entre les deux si le pixel est gris plus ou moins foncé.

L'encodage informatique d'une image est alors une matrice M de taille $L \times C$ et dont chaque coefficient M_{ij} est un entier compris entre 0 et 255, qui représente la couleur du pixel de coordonnées $a \times (i, j)$.

On peut alors définir des distances entre deux images de même taille $L \times C$, encodées

par les matrices M et N selon :

$$d_1(M,N) = \sum_{i=0}^{L-1} \sum_{j=0}^{C-1} |M_{ij} - N_{ij}|$$

$$d_2(M,N) = \sqrt{\sum_{i=0}^{L-1} \sum_{j=0}^{C-1} (M_{ij} - N_{ij})^2} \quad \text{(distance euclidienne)}$$

$$d_\infty(M,N) = \max_{\substack{0 \leqslant i \leqslant L-1 \\ 0 \leqslant j \leqslant C-1}} |M_{ij} - N_{ij}|$$

Définition 2. *Espace métrique*

Soit E un ensemble non vide. On appelle **espace métrique** le couple (E, d) où d est une distance définie sur E.

II. Classement supervisé : algorithme des k-plus proches voisins

1) L'algorithme

On veut classer les éléments $x \in E$ d'un ensemble fini E muni d'une distance d (espace métrique). On suppose qu'une partition $\mathscr{F} = \{C_0, ..., C_{p-1}\}$ de E a déjà été créée et qu'on dispose d'un sous-ensemble $A \subset E$ dont on *connaît la classe de chaque élément* : A est **l'ensemble ou base d'apprentissage**.

Le classement est dit **supervisé** car on connaît déjà les classes C_j composant $\mathscr{F}$. L'algorithme fonctionne sur le principe du vote majoritaire c'est à dire qu'il procède de la façon suivante :

1. Pour chaque élément $x \in E \backslash A$ et pour chaque élément $a \in A$, calculer $d(x, a)$.
2. Trier ces distances afin d'organiser les éléments de A selon leur proximité avec x.
3. Identifier les k-plus proches voisins de x dans A
4. Attribuer à x la classe majoritaire parmi celles de ces k-plus proches voisins. Si plusieurs classes sont à égalité : choisir l'une d'entre-elles (au hasard ou la première qui se présente, ...).

Illustrons cela avec l'exemple des iris. Les points colorés représentés sur la Figure 3 sont l'ensemble d'apprentissage A.

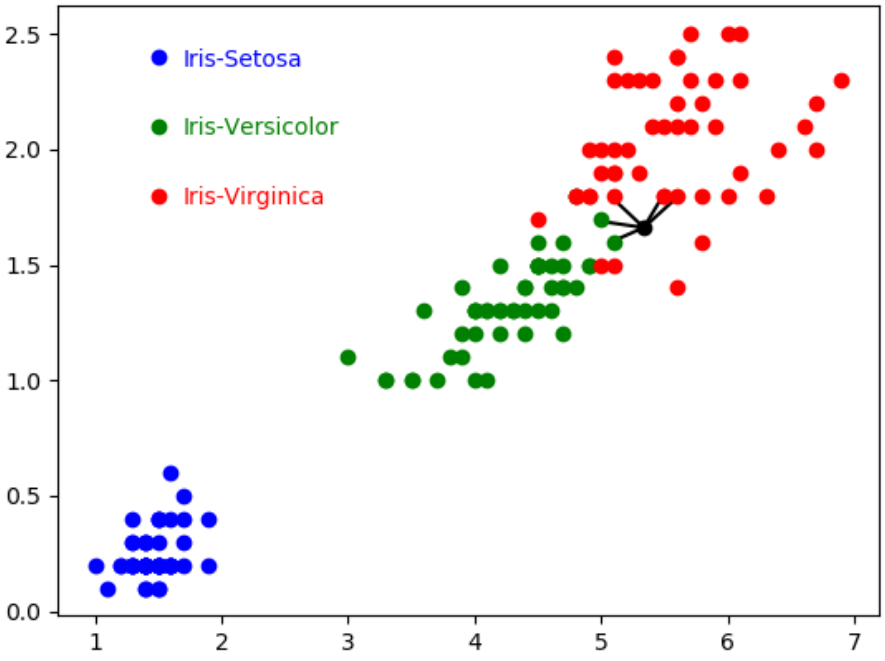


FIGURE 3 –

Considérons une nouvelle fleur d'iris représentée par le point noir. Si on prend $k = 5$, les k plus proches voisins sont indiqués sur la Figure 3 ; on voit qu'il y a 2 iris-Versicolor, 3 iris-Virginica et 0 iris-Setosa : on rangera donc cette nouvelle fleur dans la classe des iris-Virginica.

Une fois cet algorithme appliqué à tous les éléments $x \in E \backslash A$, chaque élément x est placé dans une classe $C_j \in \mathscr{F}$ et on dispose d'une **application prédiction** :

$f_{\text{k,d}} : E \longrightarrow [\![0, p-1]\!],\ x \longmapsto j$ tel que $x \in C_j$ attribué à x par l'algo. avec k voisins et une distance d

On constate que l'algorithme peut produire des résultats contraires à la réalité :

Il s'ensuit que trouver une valeur "optimale" de k est crucial. C'est toutefois un problème difficile.

2) Matrice de confusion

Pour aider à choisir une bonne valeur pour k (la distance d étant fixée) ou une bonne valeur de d (k étant fixé) l'algorithme est testé de façon systématique. On prend un ensemble E fini dont on connaît la classe de tous les éléments. En posant $\mathscr{F} = \{C_0, \ldots, C_{p-1}\}$ et en repérant chaque classe C_j par son indice j, $1 \leqslant j \leqslant p$, on introduit une **matrice confusion** M de taille $p \times p$, définie par :

$$\forall\, (i,j) \in [\![0, p-1]\!],\ M_{ij} = \text{Card}\{\, x \in C_i \mid f_{\text{k,d}}(x) = C_j \,\}$$

Autrement dit :

Définition. *Matrice confusion*

La matrice confusion M construite sur un ensemble E dont la classe de chaque élément est connue est la matrice carrée d'ordre p (p = nombre de classes) dont le coefficient M_{ij} est le nombre d'éléments de E dont la classe vraie est C_i et dont la classe estimée par l'algorithme est C_j.

3) Implémentation en langage Python

On dispose :

- d'un ensemble E représenté par une liste `E` définie comme une variable globale ;
- d'une partition $\mathscr{F} = \{C_0, ..., C_{p-1}\}$ de E représentée par une liste `F` de p listes, chaque sous-liste représentant une classe C_j d'éléments de E ;
- d'une distance sur E encodée comme une fonction `dist(x,y)` qui renvoie la distance entre les éléments `x` et `y` de la liste `E` ;
- d'une liste `A` composée d'une partie des éléments de `E` (la base d'apprentissage) ;
- d'un dictionnaire `classe_A` formé des couples $(a : j)$ où les clés a sont les éléments de `A` et où j est l'indice de la classe C_j a laquelle appartient a.

L'exercice suivant est basé sur l'exemple des iris formé de trois classes d'indice 0, 1 et 2. E sera une liste de tuples `(x0,x1)` où `x0` est la longueur des pétales et `x1` leur largeur. Vous vous reporterez au script TD_IA1 dont le début construit les ensembles E et A ainsi que le dictionnaire `classe_A`

1. Vous disposez du module `numpy` qui a été importé grâce à :

   ```
   import numpy as np
   ```

   Écrire une fonction `dist(x,y)` qui renvoie la distance euclidienne entre deux éléments `x` et `y` de E.

2. On se donne un entier $k > 0$ tel que $k < |A|$. Écrire une fonction `k_ppv(x,A,k,dist)` qui prend un élément `x` de `E` et qui renvoie une liste constituée de ses k plus proches voisins dans `A`. On ne s'inquiètera pas de savoir si `x` est oui ou non un élément de `A`.

   Indication : on pourra utiliser une liste intermédiaire `L` pour stocker les tuples `(a,dist(x,a))` lorsque `a` parcourt `A`. On pourra trier cette liste selon les valeurs croissantes de `dist(x,a)` grâce à des algorithmes de tri que vous connaissez (si vous ne savez pas lequel prendre optez pour l'algorithme de tri rapide).

3. Écrire une fonction `classe_majoritaire(L, classe_A)` qui prend en argument une liste `L` formée de k éléments de A et le dictionnaire `classe_A`. Cette fonction renvoie l'indice j associé à la classe majoritaire des éléments de `L`. Si plusieurs classe C_j obtiennent le même score maximal, la fonction renvoie l'indice de la première d'entre elles.

   Indication : cette fonction pourra utiliser un dictionnaire `dico` dont les clés sont les indices j (indice de la classe C_j) et les valeurs le nombre de fois où l'indice j apparaît pour les éléments de `L`.

4. Écrire une fonction `algo_kppv(E,A,classe_A,k,dist)` qui parcourt tous les éléments de $x \in E$ (y compris ceux de A) et qui affecte à chaque élément x l'indice j de la classe C_j donné par l'algorithme des k-plus proches voisins : la fonction renvoie un dictionnaire formé des couples $(x : j)$. On utilisera les deux fonctions précédentes.

5. Le script écrit au début du fichier TD_IA1 a construit un dictionnaire `classe_E` formé des couples $(x : j)$ où $x \in E$ et où j est l'indice de la classe de x.

   Écrire une fonction `M_conf(A,class_A,E,classe_E,k,dist)` qui applique l'algorithme des k-plus proches voisins aux éléments de E et qui renvoie la matrice confusion construite sur E. La matrice sera une matrice numpy de taille $p \times p$

III. Algorithme des k moyennes

Dans tout ce qui suit un vecteur de $\mathbb{R}^n$ sera noté en lettres majuscules : X par exemple ; X est donc un n-uplet de nombres réels de la forme $X = (x_1, ..., x_n)$. De plus :

- $\|.\|$ désigne la norme euclidienne de $\mathbb{R}^n$, c'est à dire :

$$\|X\| = \sqrt{x_1^2 + ... + x_n^2}$$

  On dispose donc de la distance euclidienne entre deux vecteurs X et Y : $d(X, Y) = \|X - Y\|$.

- On pourra dans un des exercices proposés utiliser le produit scalaire canonique de $\mathbb{R}^n$ associé à la norme euclidienne :

$$X.Y = x_1y_1 + ... + x_ny_n$$

Enfin pour toute partie **finie** A de $\mathbb{R}^n$, on note $|A|$ le cardinal de A (nombre d'éléments de A).

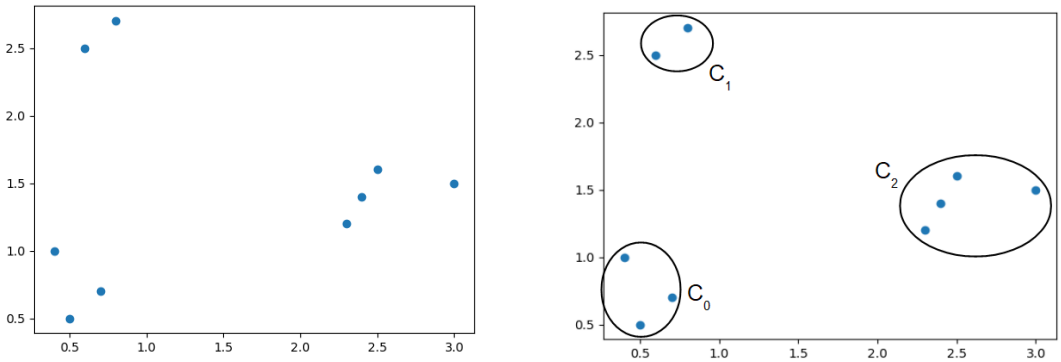
1) Ce que l'on veut obtenir avec cet algorithme

Considérons des données appartenant à un ensemble E fini et supposons qu'on puisse représenter chaque donnée par un vecteur de $\mathbb{R}^n$: E devient donc une partie finie de $\mathbb{R}^n$.

Supposons en outre que ces données (vecteurs $X \in E$) puissent se regouper selon leur proximité et que cette proximité soit mesurable par la distance euclidienne de $\mathbb{R}^n$: plus $\|X-Y\|$ est petit, plus les données représentées par X et Y sont semblables.

Exemple :

Considérons 9 données représentées par des vecteurs de $\mathbb{R}^2$ (Figure 1).



On constate que les données s'aggrègent naturellement selon leur proximité, ce qui permet de définir naturellement une partition de trois classes $\mathscr{F} = \{C_0, C_1, C_2\}$.

Est-on capable de produire un algorithme automatique permettant de trouver ces trois classes ? L'algorithme des k-moyennes apporte une réponse positive à cette question.

2) Classification non supervisée

L'algorithme des k-moyennes n'est pas supervisé : cela signifie qu'on ne possède au départ aucune partition de E. Seul le nombre k de classes qu'on veut obtenir est précisé. On aura besoin de :

Définition *Barycentre d'une partie de* $\mathbb{R}^n$

Soit A une partie *finie et non vide* de $\mathbb{R}^n$ de cardinal $|A| > 0$. Le **barycentre** de A est le point G_A de $\mathbb{R}^n$ défini par :

$$G_A = \frac{1}{|A|} \sum_{X \in A} X$$

L'algorithme des k-moyennes fonctionne de la façon suivante :

Entrée : une partie finie E de $\mathbb{R}^n$ représentant des données à classer. Un entier naturel $k > 0$.

Sortie : une partition $\mathscr{F} = \{ C_0, C_1, ..., C_{k-1} \}$ de E de cardinal k.

Version basique de l'algorithme :

Initialisation :

1. Choisir k éléments de $\mathbb{R}^n$ notés $G_0^{(0)}$, ..., $G_{k-1}^{(0)}$.
2. Créer une variable `F` qui est une liste de k listes (future partition) : chaque sous-liste de `F` représente la classe C_j. Au début, les C_j sont des listes vides.

Déroulement étape numéro $p \geqslant 1$:

DÉBUT

3. Pour chaque $X \in E$, calculer la distance $\left\| X - G_j^{(p-1)} \right\|$ pour $0 \leqslant j \leqslant k-1$.

   — Placer X dans la classe numéro j réalisant $\left\| X - G_j^{(p-1)} \right\|$ minimale ;

   — Si deux classes (ou plus) conviennent, choisir l'une d'entre elles et affecter X à cette classe.

   À l'issue de 3. on obtient k classes $C_j^{(p)}$, $0 \leqslant j \leqslant k-1$.

4. Calculer les k barycentres $G_j^{(p)}$ de chaque classe $C_j^{(p)}$.

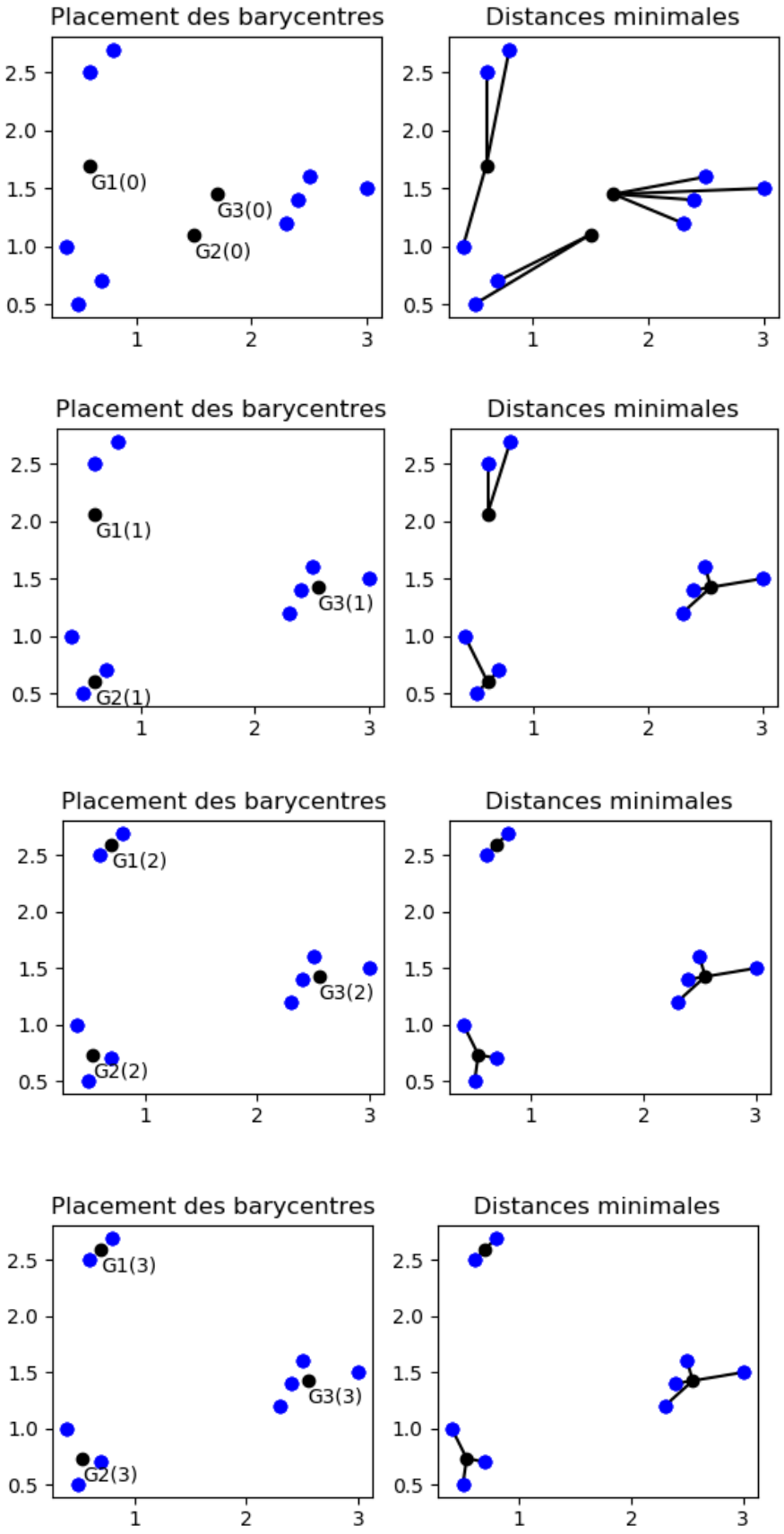
FIN

5. Recommencer 3. et 4. autant de fois que nécessaire jusqu'à ce que les $G_j^{(p)}$ ne varient plus c'est à dire qu'ils vérifient :

$$\max_{0 \leqslant j \leqslant k-1} \| G_j^{(p)} - G_j^{(p-1)} \| < \varepsilon$$

où ε est une précision que l'on se donne, par exemple 10^{-4}.

Exemple de fonctionnement


Placement des barycentres
Distances minimales
G1(0)
G3(0)
G2(0)
Placement des barycentres
Distances minimales
G1(1)
G3(1)
G2(1)
Placement des barycentres
Distances minimales
G1(2)
G3(2)
G2(2)
Placement des barycentres
Distances minimales
G1(3)
G3(3)
G2(3)

Exercice : soient $C_j^{(p)}$, $0 \leqslant j \leqslant k-1$ les classes d'éléments de E obtenus à la fin de l'étape numéro p ($p \geqslant 1$) de l'algorithme et $\mathscr{F}_p = \{ C_0^{(p)}, ..., C_{k-1}^{(p)} \}$. L'ensemble $\mathscr{F}_p$ obtenu à l'itération p est-il toujours une partition de E ?

Au cours des itérations successives la distance entre chaque point d'une classe et le barycentre de la classe diminue et finit par ne plus changer. Cela justifie les définitions suivantes :

3) Inertie d'une partition

Définition 2. *Inertie d'une partie de $\mathbb{R}^n$*

Soit A une partie *finie et non vide* de $\mathbb{R}^n$ de cardinal $|A| > 0$. **L'inertie** de A est définie par :

$$I_A = \sum_{X \in A} \|X - G_A\|^2$$

où G_A est le barycentre de A.

Définition 3. *Inertie d'une partition $\mathscr{F}$ de E*

Soit $\mathscr{F} = \{ C_0, ..., C_{k-1} \}$ une partition d'une partie **finie** E de $\mathbb{R}^n$. On appelle **inertie de la partition** $\mathscr{F}$ la somme des inerties de chacune des classes qui composent $\mathscr{F}$:

$$I_{\mathscr{F}} = \sum_{j=0}^{k-1} I_{C_j}$$

On voit donc que l'algorithme des k-moyennes "minimise" les inerties de chaque classe créée ou, ce qui est équivalent puisque les inerties sont positives, "minimise" l'inertie de la partition $\mathscr{F}$ créée.

Exercice (hors programme) :

Posons $\mathscr{F}_p = \{ C_0^{(p)}, ..., C_{k-1}^{(p)} \}$ la partition obtenue à la fin de l'étape numéro p et notons $G_j^{(p)}$ le barycentre de $C_j^{(p)}$, avec $0 \leqslant j \leqslant k-1$. On définit pour $p \geqslant 1$:

$$I_p = \sum_{j=0}^{k-1} \sum_{X \in C_j^{(p)}} \|X - G_j^{(p)}\|^2 \quad \text{et} \quad Q_p = \sum_{j=0}^{k-1} \sum_{X \in C_j^{(p)}} \|X - G_j^{(p-1)}\|^2$$

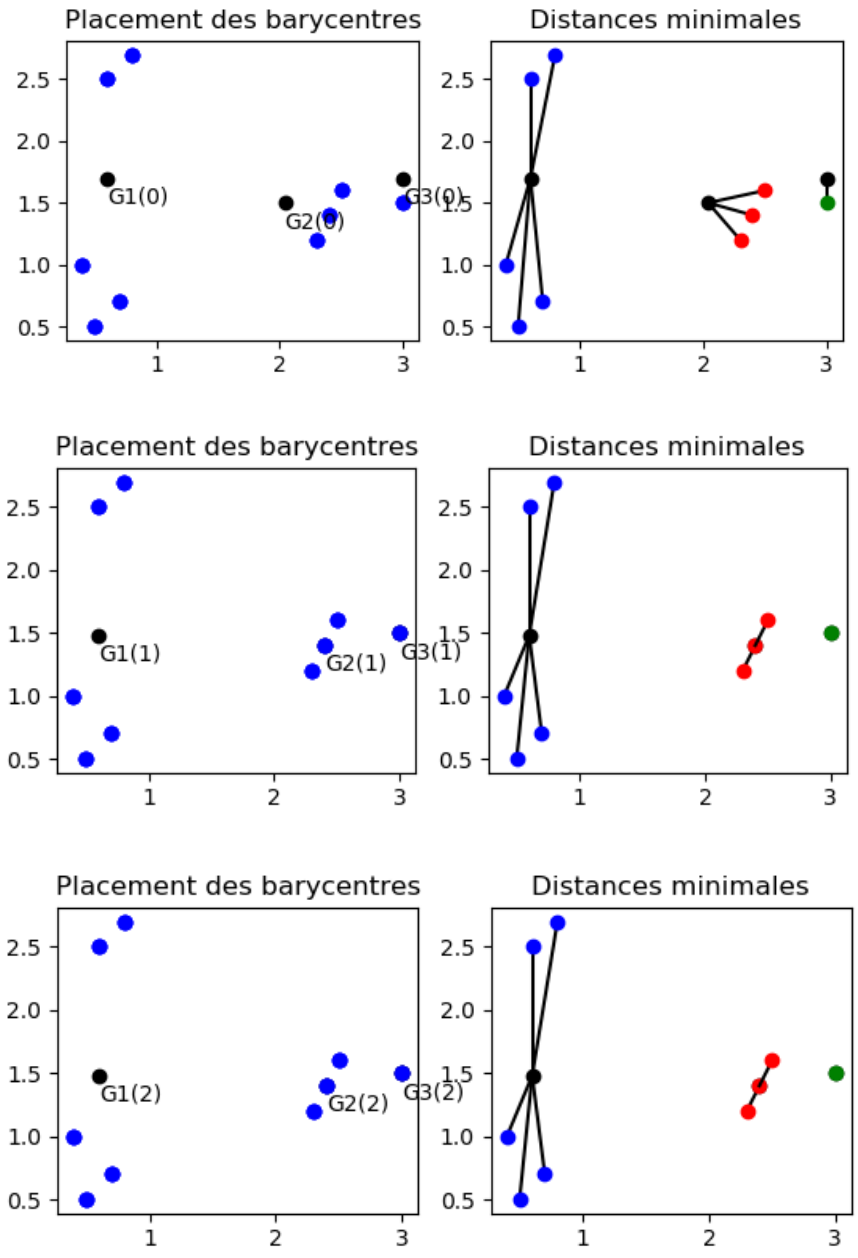
1. Que représente I_p ?
2. Montrer que pour tout $j \in [\![0, k-1]\!]$ et tout $p \geqslant 1$:

$$\sum_{X \in C_j^{(p)}} \|X - G_j^{(p-1)}\|^2 = \sum_{X \in C_j^{(p)}} \|X - G_j^{(p)}\|^2 + \sum_{X \in C_j^{(p)}} \|G_j^{(p)} - G_j^{(p-1)}\|^2$$

   On pourra utiliser la relation entre la norme euclidienne et le produit scalaire de $\mathbb{R}^n$, ainsi que la définition des barycentres.
3. En déduire que $\forall\, p \geqslant 1,\ Q_p \geqslant I_p$.
4. En raisonnant sur les distances aux barycentres, montrer d'autre part que, pour $p \geqslant 2$, $I_{p-1} \geqslant Q_p$.
5. En déduire que la suite $(I_p)_{p \geqslant 1}$ est décroissante, puis qu'elle est convergente.

4) Limites de l'algorithme

L'exercice précédent montre que l'algorithme va forcément converger vers une partition dont l'inertie est un minimum **compte-tenu du choix initial des** $G_j^{(0)}$. Rien ne prouve cependant qu'on obtient une partition $\mathscr{F}$ conforme à ce qui est souhaité. L'exemple suivant illustre ce point.



La partition limite atteinte dépend donc grandement de l'initialisation ! La seule garantie de l'algorithme est de converger vers un **point fixe**, c'est à dire vers une configuration invariante par pplication successive de l'algorithme.

Beaucoup de points fixes ne correspondent pas à un minimum absolu de $I_{\mathscr{F}}$ (nous supposons qu'il existe une partition $\mathscr{F}$ qui réalise ce minimum absolu). C'est pour cela qu'il est souhaitable de lancer plusieurs fois l'algorithme avec des initialisations aléatoires différentes, puis à sélectionner parmi les solutions trouvées celle qui conduit à la plus petite valeur de $I_{\mathscr{F}}$.

En pratique il y a deux méthodes pour choisir les $G_j^{(0)}$:

Méthode 1 :

Créer une partition aléatoire de E formée de k classes et prendre pour $G_j^{(0)}$ les k barycentres de chacune de ces classes ;

Méthode 2 :

Choisir aléatoirement k points $G_j^{(0)}$ de E.

5) Exemple de réalisation

Vous vous reporterez au fichier TD_IA2 qui se base à nouveau sur l'exemple des iris. L'ensemble E est un ensemble de tableaux `numpy` à deux éléments de la forme `np.array([x0,x1])` où `x0` et `x1` sont respectivement les longueurs et largeurs des pétales. E est donc une partie finie de $\mathbb{R}^2$

Une partition finie $\mathscr{F}$ de E sera représentée par une liste `F` de k listes, chaque sous-liste `F[j]` de `F` représentant une classe C_j d'éléments de E.

1. Étant donnés deux tableaux numpy `X` et `Y` représentant deux vecteurs de $\mathbb{R}^2$ et `a` un nombre flottant, que représente `X+Y` ? `a*X` ?
2. *On commence par une fonction utile pour la suite* : écrire une fonction `norme(X)` qui renvoie la norme euclidienne du vecteur `X` de $\mathbb{R}^2$.
3. Écrire une fonction `barycentre_partie(A)` qui prend comme argument une liste `A` de vecteurs de $\mathbb{R}^2$ et qui renvoie le barycentre de ces vecteurs.
4. Écrire une fonction `barycentres(F)` qui prend comme argument une partition finie `F` de E qui renvoie une liste de la forme $[G_0, ..., G_{k-1}]$ où G_j est le barycentre de la classe `F[j]`.
5. Écrire une fonction `inertie(F)` qui prend comme argument une partition `F` d'un ensemble fini E et qui renvoie l'inertie $I_{\mathscr{F}}$ de cette partition.
6. a) Écrire une fonction `IndiceMinimum(L)` qui renvoie *l'indice* du premier élément minimal de la liste `L`
   b) Écrire une fonction `classe_min(X,Gliste)` dont les paramètres sont un vecteur `X` de E et une liste de barycentres tel que celle renvoyée par `barycentres(F)`. Cette fonction renvoie le numéro `j` de la classe telle que $\|X - G_j\|$ est minimale. Si plusieurs classes conviennent on choisit (pour des raisons de simplicité de la fonction) celle dont le numéro est le plus petit.
7. Écrire une fonction `algo_k_moyennes(E:liste,k:int,epsilon:float)` qui renvoie une partition de l'ensemble E en k classes. Cette fonction :
   - choisit au hasard k points initiaux $G_j^{(0)} \in E$ distincts deux à deux (on utilise donc la **méthode 2** du 4)). Vous utiliserez la fonction `randint(a,b)` (du module `random`) qui renvoie un entier choisi au hasard entre les entiers `a` et `b` inclus ;
   - puis utilise l'algorithme des k-moyennes pour construire une partition de E. On pourra utiliser `max(L)` qui renvoie le plus grand élément de la liste `L`.

   `epsilon` est la précision (par exemple 10^{-6}) qui permet de terminer l'algorithme (point 5. de la description de l'algorithme).

8. Test : vous pourrez tester l'algorithme précédent en dessinant une figure représentant les différents points de E colorés selon la classe attribuée par l'algo. On rappelle que :
   `plt.plot([x],[y],"ro")`
   dessine un point de coordonnées (x, y) et le colorie en rouge. Si on veut une couleur bleue on utilisera `"bo"` comme troisième paramètre de la fonction. De même un troisième paramètre égal à `"go"` trace un point vert. Comparer à la figure réelle jointe au fichier.