Algorithme des k-moyennes

Quentin Fortier

January 18, 2023

Classification supervisée/non supervisée

Définition

Un algorithme de classification est un algorithme qui permet d'associer à chaque donnée une classe (une espèce de fleur, un chiffre...)

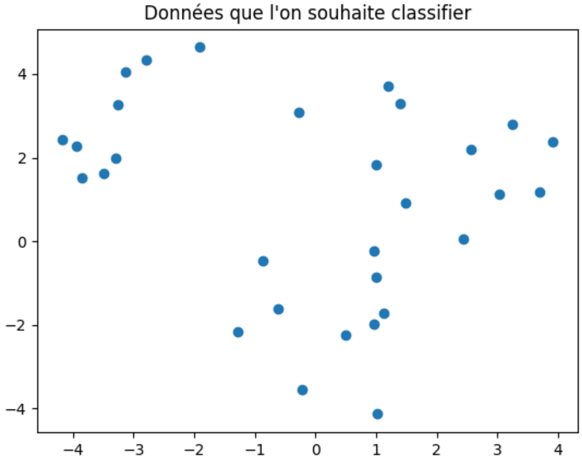
Définition

On distingue deux types d'algorithmes de classification :

1. **Classification supervisée** : on connaît les classes de certaines données (données d'entraînement) qui permettent de prédire la classe d'une nouvelle donnée.
   Exemple : algo. des plus proches voisins
2. **Classification non supervisée** : aucune classe n'est connue (pas de données d'entraînement).
   Exemple : algo. des k-moyennes

Algorithme des k-moyennes : Principe général

Exemple : il semble y avoir $k = 3$ classes de données parmi ces points.



Algorithme des k-moyennes : Principe général

Définition

Le **centre** (ou : **isobarycentre**) d'un ensemble de vecteurs $x_1, \dots, x_n$ est le vecteur

$$\overline{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$$

Question

Écrire une fonction `centre(X)` renvoyant le centre de la liste de vecteurs `X`.

Algorithme des k-moyennes : Principe général

Dans tout le cours, on note d la distance euclidienne.

Définition

La **variance** (ou : **moment d'inertie**) $V(X)$ d'un ensemble de vecteur X est définie par

$$V(X) = \sum_{x \in X} d(x, \overline{X})^2$$

La variance mesure la variation par rapport à la moyenne : plus $V(X)$ est petit, plus les vecteurs de X sont proches du barycentre $\overline{X}$.

Algorithme des k-moyennes : Principe général

Soit X un ensemble de données et un entier k.

Objectif

On veut trouver un partitionnement (*clustering*) de X en k sous-ensembles $X_1, \ldots, X_k$ (classes ou *clusters*) minimisant l'**inertie** I :

$$I = \sum_{i=1}^{k} V(X_i)$$

Dit autrement : on veut associer à chaque donnée x une classe k telle que l'inertie I soit minimum.

Plus l'inertie est petite, plus les données sont proches du centre de leur classe et plus le partitionnement est bon.

Algorithme des k-moyennes : Principe général

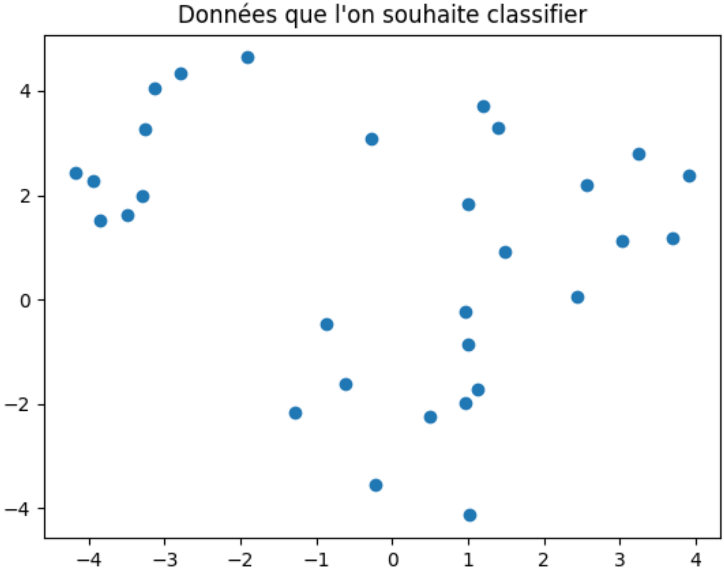
Algorithme des k-moyennes (k-*means*)

Objectif : partitionnner X en classes X_1, ..., X_k.

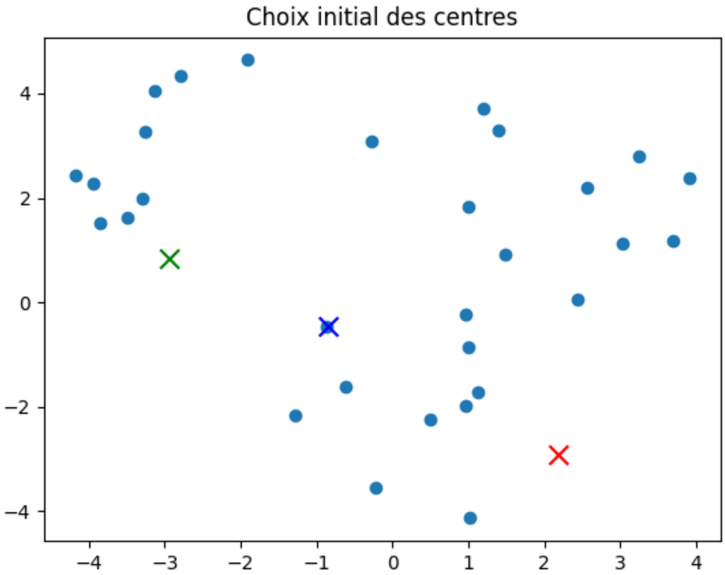
1. Soit $c_1, ..., c_k$ des vecteurs choisis aléatoirement.
2. Associer chaque donnée x à la classe X_i telle que $d(x, c_i)$ soit minimale.
3. Recalculer les centres des classes $c_i = \overline{X_i}$.
4. Si les centres ont changé, revenir à l'étape 2.

Attention : dans l'algorithme des k-moyennes, k est le nombre de classes alors que dans l'algorithme des plus proches voisins, k est le nombre de voisins.

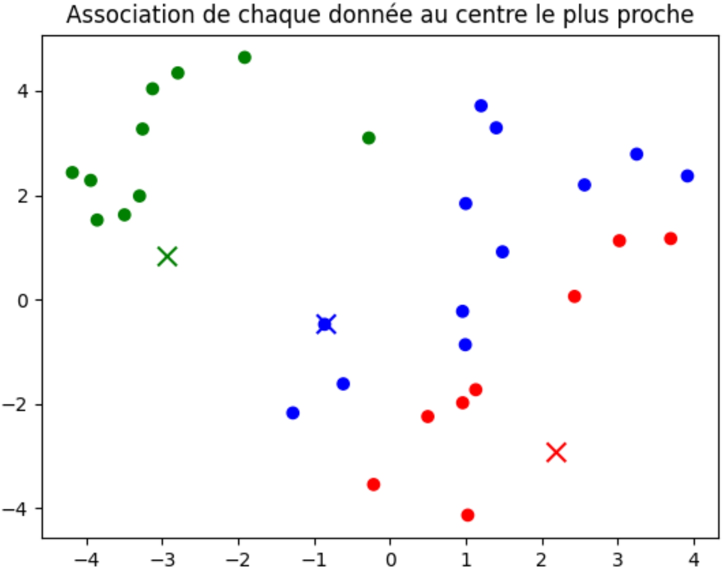
Algorithme des k-moyennes : Exemple



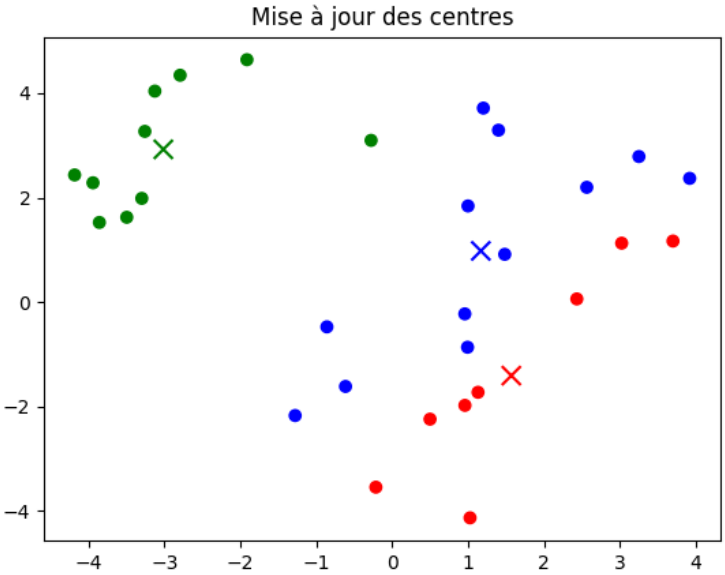
Algorithme des k-moyennes : Exemple



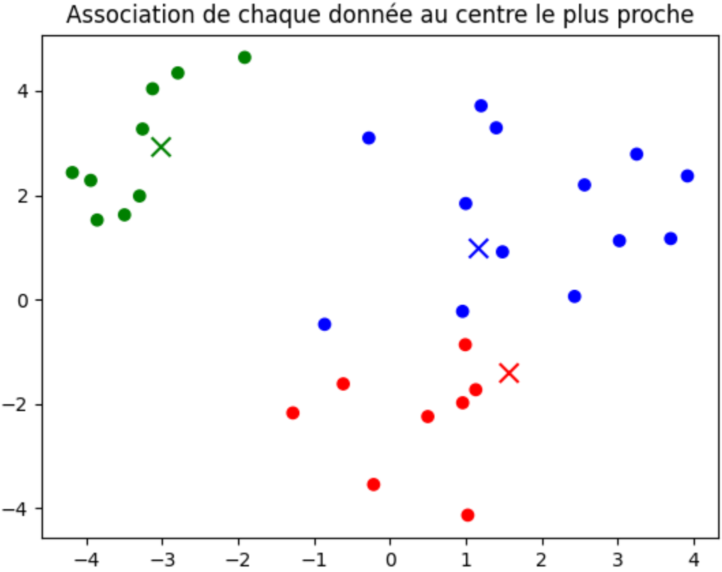
Algorithme des k-moyennes : Exemple



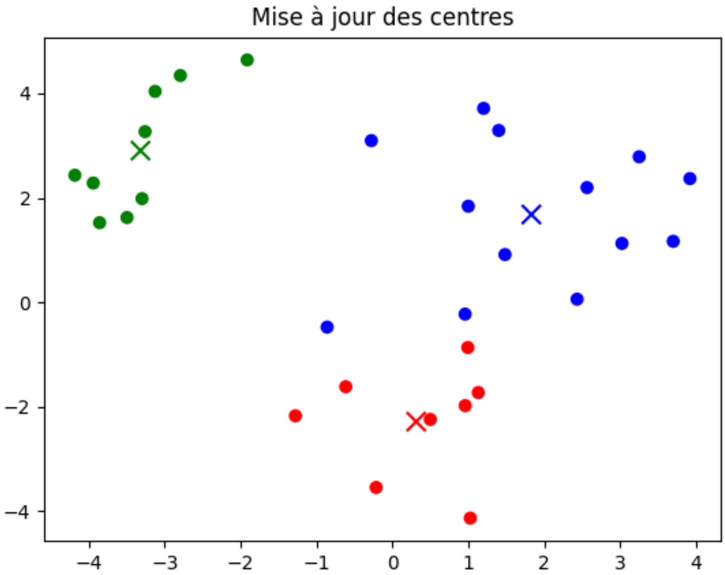
Algorithme des k-moyennes : Exemple



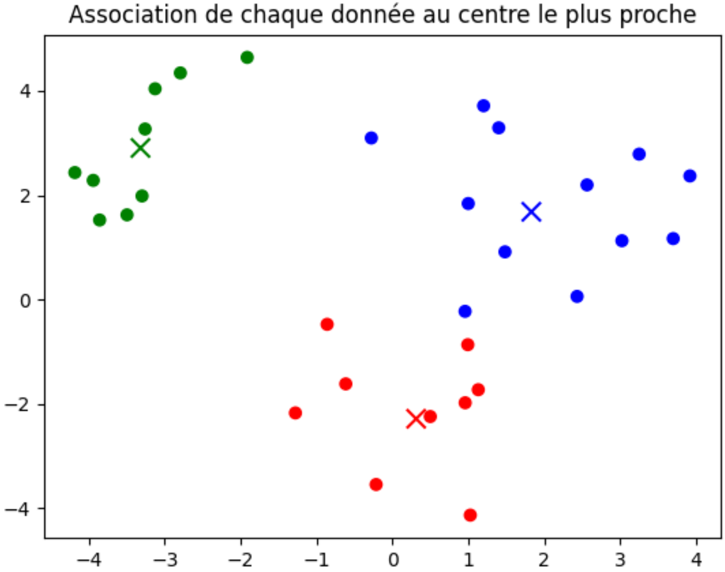
Algorithme des k-moyennes : Exemple



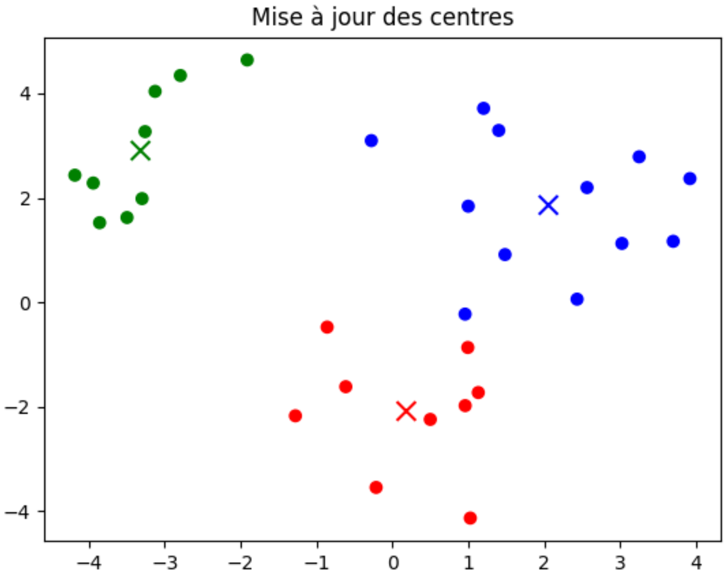
Algorithme des k-moyennes : Exemple



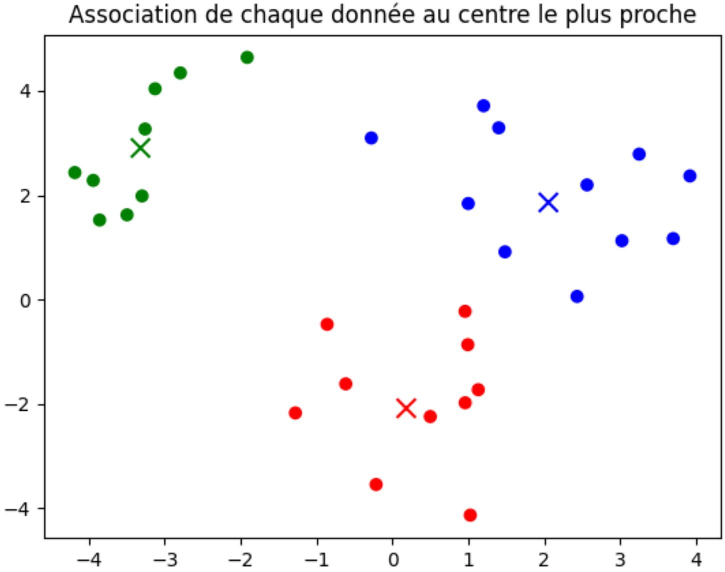
Algorithme des k-moyennes : Exemple



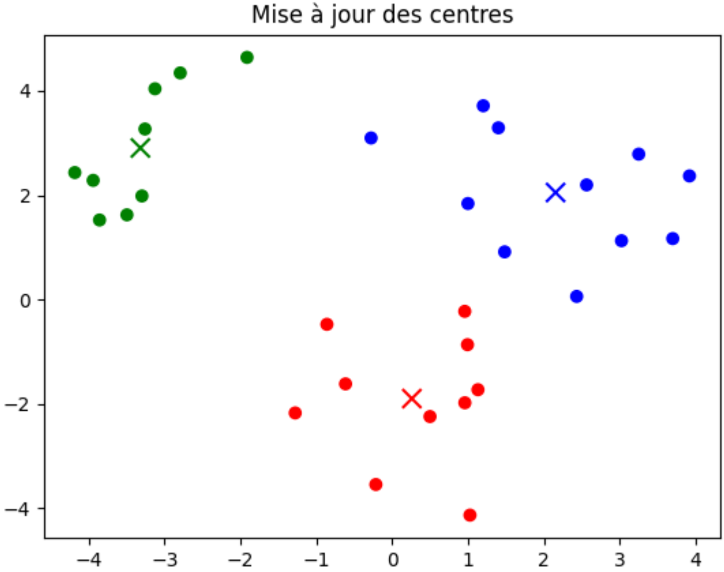
Algorithme des k-moyennes : Exemple



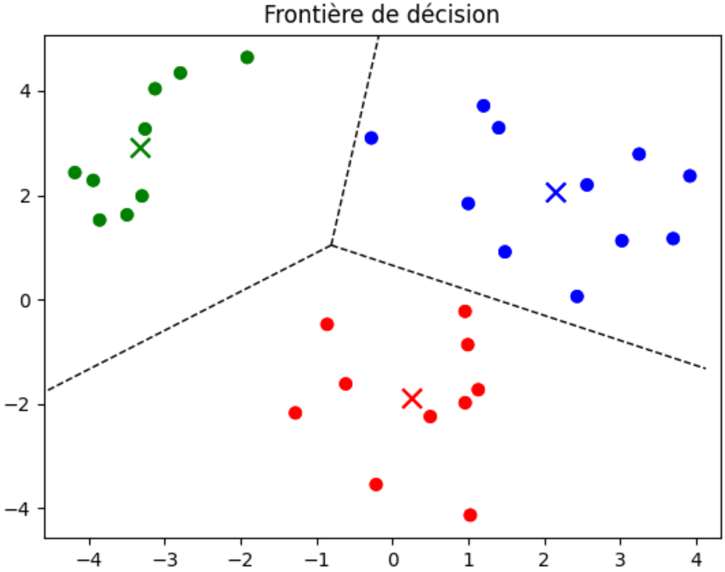
Algorithme des k-moyennes : Exemple



Algorithme des k-moyennes : Exemple



Algorithme des k-moyennes : Exemple



Algorithme des k-moyennes : Terminaison

Théorème (HP)

L'algorithme des k-moyennes termine (pas de boucle infinie).

Preuve :
Il existe un nombre fini de partitions de X en k classes, donc l'inertie I peut prendre qu'un nombre fini de valeurs.
Il suffit donc de montrer que I décroît strictement à chaque itération.

Algorithme des k-moyennes : Terminaison

Preuve (suite) :

Il suffit donc de montrer que I décroît strictement à chaque itération :

- Réassigner x de X_i à X_j si $d(x, c_i) > d(x, c_j)$ fait diminuer I.
- Recalculer les centres des classes fait diminuer I, d'après le lemme suivant :

Lemme

Si X est un ensemble de vecteurs alors $f : y \mapsto \sum_{x \in X} d(x, y)^2$ est minimum pour $y = \overline{X}$.

Algorithme des k-moyennes : Implémentation

On utilise une liste `classes` de taille k telle que `classes[i]` est la liste des données de `X` associées à la classe i.

Question

Écrire une fonction `calculer_centres(classes)` renvoyant la liste des centres de chaque classe.

Algorithme des k-moyennes : Implémentation

Question

Écrire une fonction `plus_proche(x, centres)` renvoyant le numéro `i` de la classe la plus proche de `x` parmi `centres`, c'est-à-dire la classe telle la distance de `x` à `centres[i]` soit minimum

Algorithme des k-moyennes : Implémentation

Question

Écrire une fonction `calculer_classes(X, centres)` renvoyant une liste `classes` telle que `classes[i]` soit la liste des données de `X` dont le centre le plus proche est `centres[i]`.

Algorithme des k-moyennes : Implémentation

Question

Écrire une fonction `kmeans(X, centres)` appliquant l'algorithme des k-moyennes à `X` en partant des centres `centres` et renvoyant la liste des classes obtenues.

Algorithme des k-moyennes : Choix des centres initiaux

L'inertie du clustering obtenu à la fin de l'algorithme dépend des centres initiaux.

Il y a plusieurs façons de les choisir :

- Aléatoirement dans l'espace.
- Aléatoirement parmi les données.
- Lancer l'algorithme plusieurs fois avec des centres initiaux différents et conserver la meilleure solution (celle d'inertie minimum).
- Autres méthodes : *k-means++*...

Algorithme des k-moyennes : Choisir k

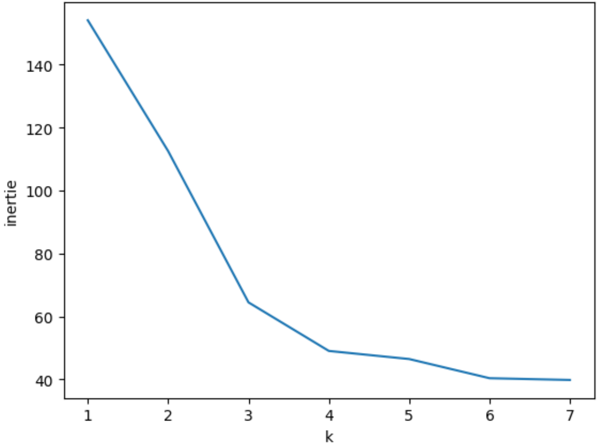
Question

Comment choisir le nombre k de classes ?

On peut calculer l'inertie obtenue pour différentes valeurs de k.
Cependant, plus k est grand, plus l'inertie diminue jusqu'à valoir 0 si k est égal au nombre de données (ce qui n'a aucun intérêt.
On choisit donc la plus grande valeur de k pour laquelle l'inertie diminue de façon significative.

Algorithme des k-moyennes : Choisir k

Méthode du coude (*elbow method*) : On choisit la plus grande valeur de k pour laquelle l'inertie diminue de façon significative (ci-dessous : 3 ou 4).



Algorithme des k-moyennes : Non optimalité

L'algorithme des k-moyennes converge toujours, mais pas forcément vers un minimum global de l'inertie (seulement vers un minimum local).

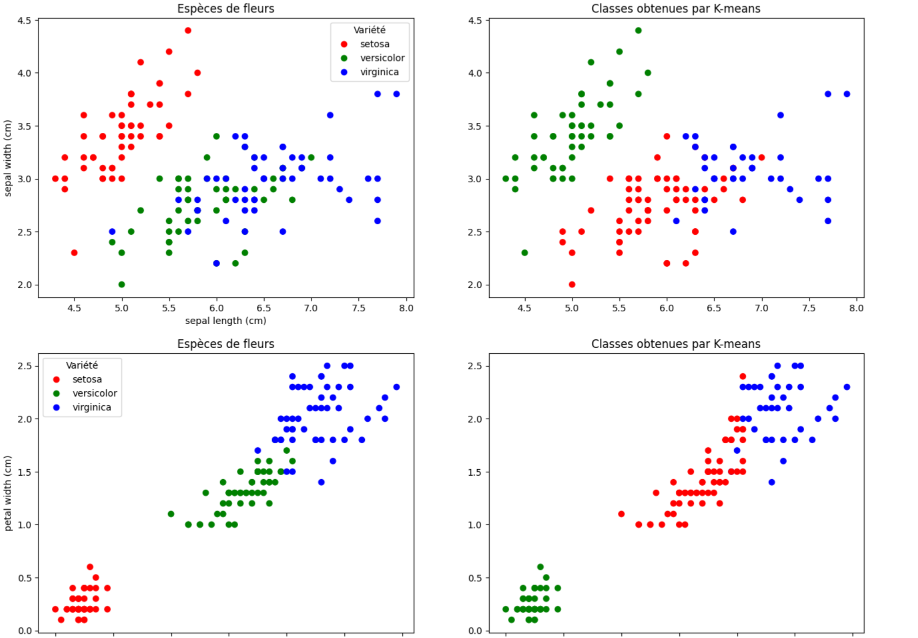
Question

Donner un exemple d'exécution de l'algorithme des k-moyennes qui ne donne pas un clustering d'inertie optimale.

Algorithme des k-moyennes : Classification de nouvelles données

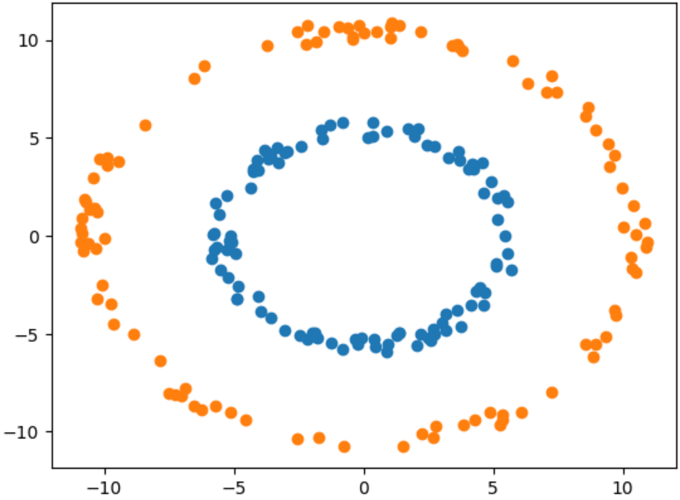
On peut utiliser l'algorithme des k-moyennes pour classifier une nouvelle donnée x : on associe x à la classe dont le centre est le plus proche de x.

Algorithme des k-moyennes : Application aux iris



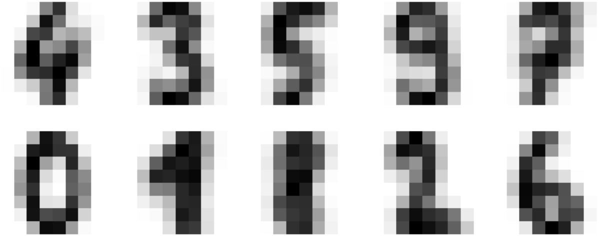
Algorithme des k-moyennes : Limites

L'algorithme des k-moyennes ne marche que sur des données linéairement séparables (pouvant être séparées par un hyperplan).



Algorithme des k-moyennes : Interprétations

Les centres obtenus à la fin de l'algorithme donnent des informations sur les constituants des classes.
Voici par exemple les centres obtenus avec $k = 10$ sur des chiffres manuscrits :



Algorithme des k-moyennes : Interprétations

Il est également intéressant de regarder l'équation des frontières de décision, pour savoir quels sont les attributs qui permettent de discriminer les données.

Si par exemple l'équation de la frontière de décision entre les classes 1 et 2 est $ax + by = 0$ avec $a \gg b$, alors l'attribut x est plus discriminant que y pour pouvoir distinguer les classes 1 et 2.

Application à la compression d'images

Une image est souvent représentée par une matrice dont chaque élément (pixel) est un triplet de valeurs entre 0 et 255 (rouge, vert, bleu).

Question

Combien y a t-il de couleurs différentes possibles ?

Question

Écrire une fonction `nombre_couleurs(img)` qui renvoie le nombre de couleurs différentes présentes dans l'image `img`.

Application à la compression d'images

On peut souhaiter limiter le nombre de couleurs différentes :

- Sur un écran avec un nombre plus limité de couleurs (console...)
- Pour diminuer la taille de l'image : si on utilise k couleurs, on peut utiliser stocker un entier entre 1 et k pour chaque pixel au lieu de trois entiers entre 0 et 255.

Application à la compression d'images

On applique l'algorithme des k-moyennes sur les pixels pour obtenir k couleurs différentes (ici $k = 8$) et on remplace chaque pixel par la couleur la plus proche.