Algorithmes d'apprentissage

MP – PC – PSI* - Lycée Leconte de Lisle

12 février 2023

Plan

Intelligence Artificielle

- Branche de l'informatique sous les feux de la rampe.

Intelligence Artificielle

- Branche de l'informatique sous les feux de la rampe.
- Applications récentes :

Intelligence Artificielle

- Branche de l'informatique sous les feux de la rampe.
- Applications récentes :
  - moteurs de recherche

Intelligence Artificielle

- Branche de l'informatique sous les feux de la rampe.
- Applications récentes :
  - moteurs de recherche
  - programmes de jeux battant des êtres humains (`Deep Blue` pour les échecs, `AlphaGo` pour le jeu de go)

Intelligence Artificielle

- Branche de l'informatique sous les feux de la rampe.
- Applications récentes :
  - moteurs de recherche
  - programmes de jeux battant des êtres humains (`Deep Blue` pour les échecs, `AlphaGo` pour le jeu de go)
  - voiture autonome

Intelligence Artificielle

- Branche de l'informatique sous les feux de la rampe.
- Applications récentes :
  - moteurs de recherche
  - programmes de jeux battant des êtres humains (`Deep Blue` pour les échecs, `AlphaGo` pour le jeu de go)
  - voiture autonome
  - reconnaissance vocale

Intelligence Artificielle

- Branche de l'informatique sous les feux de la rampe.
- Applications récentes :
  - moteurs de recherche
  - programmes de jeux battant des êtres humains (`Deep Blue` pour les échecs, `AlphaGo` pour le jeu de go)
  - voiture autonome
  - reconnaissance vocale
  - reconnaissance faciale

Intelligence Artificielle

- Branche de l'informatique sous les feux de la rampe.
- Applications récentes :
  - moteurs de recherche
  - programmes de jeux battant des êtres humains (`Deep Blue` pour les échecs, `AlphaGo` pour le jeu de go)
  - voiture autonome
  - reconnaissance vocale
  - reconnaissance faciale
  - traduction instantanée

Intelligence Artificielle

- Branche de l'informatique sous les feux de la rampe.
- Applications récentes :
  - moteurs de recherche
  - programmes de jeux battant des êtres humains (`Deep Blue` pour les échecs, `AlphaGo` pour le jeu de go)
  - voiture autonome
  - reconnaissance vocale
  - reconnaissance faciale
  - traduction instantanée
  - détection de spams

Intelligence Artificielle

- Branche de l'informatique sous les feux de la rampe.
- Applications récentes :
  - moteurs de recherche
  - programmes de jeux battant des êtres humains (`Deep Blue` pour les échecs, `AlphaGo` pour le jeu de go)
  - voiture autonome
  - reconnaissance vocale
  - reconnaissance faciale
  - traduction instantanée
  - détection de spams
  - `ChatGPT` (*Generative Pre-trained Transformer for Chat*)

Intelligence Artificielle

- Branche de l'informatique sous les feux de la rampe.
- Applications récentes :
  - moteurs de recherche
  - programmes de jeux battant des êtres humains (`Deep Blue` pour les échecs, `AlphaGo` pour le jeu de go)
  - voiture autonome
  - reconnaissance vocale
  - reconnaissance faciale
  - traduction instantanée
  - détection de spams
  - `ChatGPT` (*Generative Pre-trained Transformer for Chat*)
- « Apprentissage Automatique » (*Machine Learning* en anglais) = Big Data + Puissance de calcul (calcul parallèle)

Intelligence Artificielle

- Branche de l'informatique sous les feux de la rampe.
- Applications récentes :
  - moteurs de recherche
  - programmes de jeux battant des êtres humains (`Deep Blue` pour les échecs, `AlphaGo` pour le jeu de go)
  - voiture autonome
  - reconnaissance vocale
  - reconnaissance faciale
  - traduction instantanée
  - détection de spams
  - `ChatGPT` (*Generative Pre-trained Transformer for Chat*)
- « Apprentissage Automatique » (*Machine Learning* en anglais) = Big Data + Puissance de calcul (calcul parallèle)
  - « Réseau de Neurones Profond » (*Deep Neural Networks* en anglais, ou `DNN`) – programme de SII

Intelligence Artificielle

- Branche de l'informatique sous les feux de la rampe.
- Applications récentes :
  - moteurs de recherche
  - programmes de jeux battant des êtres humains (`Deep Blue` pour les échecs, `AlphaGo` pour le jeu de go)
  - voiture autonome
  - reconnaissance vocale
  - reconnaissance faciale
  - traduction instantanée
  - détection de spams
  - `ChatGPT` (*Generative Pre-trained Transformer for Chat*)
- « Apprentissage Automatique » (*Machine Learning* en anglais) = Big Data + Puissance de calcul (calcul parallèle)
  - « Réseau de Neurones Profond » (*Deep Neural Networks* en anglais, ou `DNN`) – programme de SII
  - Algorithmes de classification **supervisée** : au programme « k plus proches voisins » (en anglais k-*Nearest Neigbors* ou k-`NN`)

Intelligence Artificielle

- Branche de l'informatique sous les feux de la rampe.
- Applications récentes :
  - moteurs de recherche
  - programmes de jeux battant des êtres humains (`Deep Blue` pour les échecs, `AlphaGo` pour le jeu de go)
  - voiture autonome
  - reconnaissance vocale
  - reconnaissance faciale
  - traduction instantanée
  - détection de spams
  - `ChatGPT` (*Generative Pre-trained Transformer for Chat*)
- « Apprentissage Automatique » (*Machine Learning* en anglais) = Big Data + Puissance de calcul (calcul parallèle)
  - « Réseau de Neurones Profond » (*Deep Neural Networks* en anglais, ou `DNN`) – programme de SII
  - Algorithmes de classification **supervisée** : au programme « k plus proches voisins » (en anglais k-*Nearest Neigbors* ou k-`NN`)
  - Algorithmes de classification **non supervisée** : au programme « k moyennes » (en anglais k-*means*)

Plan

Présentation de l'algorithme

L'algorithme des k plus proches voisins est un algorithme de classification supervisée : il demande de connaître, pour un nombre important de données, la classe (catégorie) de chacune de ces données, et tente d'en déduire la classe d'une nouvelle donnée en s'intéressant à ses k plus proches voisins, dans un sens à définir.

Présentation de l'algorithme

Définition : Algorithme des k plus proches voisins

Soit $k \in \mathbb{N}^*$, et $\big((x_i, c_i)\big)_{1 \leqslant i \leqslant n}$ une base d'apprentissage : les $x_i \in \mathbb{R}^d$ sont les données, les c_i sont des classes associées à chaque donnée, dans un ensemble $\mathcal{C}$ de classes (de cardinal assez petit).

Présentation de l'algorithme

Définition : Algorithme des k plus proches voisins

Soit $k \in \mathbb{N}^*$, et $\big((x_i, c_i)\big)_{1 \leqslant i \leqslant n}$ une base d'apprentissage : les $x_i \in \mathbb{R}^d$ sont les données, les c_i sont des classes associées à chaque donnée, dans un ensemble $\mathcal{C}$ de classes (de cardinal assez petit).
L'**algorithme des k plus proches voisins** associe à une nouvelle donnée $x \in \mathbb{R}^d$, la classe la plus représentée parmi les classes des k éléments de $(x_i)_{1 \leqslant i \leqslant n}$ les plus proches de x.

Présentation de l'algorithme

Définition : Algorithme des k plus proches voisins

Soit $k \in \mathbb{N}^*$, et $\big((x_i, c_i)\big)_{1 \leqslant i \leqslant n}$ une base d'apprentissage : les $x_i \in \mathbb{R}^d$ sont les données, les c_i sont des classes associées à chaque donnée, dans un ensemble $\mathcal{C}$ de classes (de cardinal assez petit).
L'**algorithme des k plus proches voisins** associe à une nouvelle donnée $x \in \mathbb{R}^d$, la classe la plus représentée parmi les classes des k éléments de $(x_i)_{1 \leqslant i \leqslant n}$ les plus proches de x.

Cet algorithme dépend de deux choix, qui peuvent influer sur ses performances :

- celui de l'entier k ;

Présentation de l'algorithme

Définition : Algorithme des k plus proches voisins

Soit $k \in \mathbb{N}^*$, et $\big((x_i, c_i)\big)_{1 \leqslant i \leqslant n}$ une base d'apprentissage : les $x_i \in \mathbb{R}^d$ sont les données, les c_i sont des classes associées à chaque donnée, dans un ensemble $\mathcal{C}$ de classes (de cardinal assez petit).
L'**algorithme des k plus proches voisins** associe à une nouvelle donnée $x \in \mathbb{R}^d$, la classe la plus représentée parmi les classes des k éléments de $(x_i)_{1 \leqslant i \leqslant n}$ les plus proches de x.

Cet algorithme dépend de deux choix, qui peuvent influer sur ses performances :

- celui de l'entier k ;
- celui de la définition de la distance entre les données.

Présentation de l'algorithme

Définition : Algorithme des k plus proches voisins

Soit $k \in \mathbb{N}^*$, et $\big((x_i, c_i)\big)_{1 \leqslant i \leqslant n}$ une base d'apprentissage : les $x_i \in \mathbb{R}^d$ sont les données, les c_i sont des classes associées à chaque donnée, dans un ensemble $\mathcal{C}$ de classes (de cardinal assez petit).
L'**algorithme des k plus proches voisins** associe à une nouvelle donnée $x \in \mathbb{R}^d$, la classe la plus représentée parmi les classes des k éléments de $(x_i)_{1 \leqslant i \leqslant n}$ les plus proches de x.

Cet algorithme dépend de deux choix, qui peuvent influer sur ses performances :

- celui de l'entier k ;
- celui de la définition de la distance entre les données.

Présentation de l'algorithme

Définition : Algorithme des k plus proches voisins

Soit $k \in \mathbb{N}^*$, et $\left((x_i, c_i)\right)_{1 \leqslant i \leqslant n}$ une base d'apprentissage : les $x_i \in \mathbb{R}^d$ sont les données, les c_i sont des classes associées à chaque donnée, dans un ensemble $\mathcal{C}$ de classes (de cardinal assez petit).
L'**algorithme des k plus proches voisins** associe à une nouvelle donnée $x \in \mathbb{R}^d$, la classe la plus représentée parmi les classes des k éléments de $(x_i)_{1 \leqslant i \leqslant n}$ les plus proches de x.

Cet algorithme dépend de deux choix, qui peuvent influer sur ses performances :

- celui de l'entier k ;
- celui de la définition de la distance entre les données.

Pour ce dernier, le programme stipule que seul le cas de la distance euclidienne habituelle est utilisée ici.

Exemple : la base Iris

Figure – Iris versicolor

- $d = 4$ (longueur et largeur des pétales et des sépales) ; $n = 150$; $|\mathcal{C}| = 3$ (espèces setosa, versicolor ou virginica)
- Classe `iris` de `sklearn.datasets`
- Utilisé dans ce cours avec seulement deux caractéristiques ($d = 2$: longueur et largeur des sépales) et $n = 116$.

Exemple : détection de chiffres

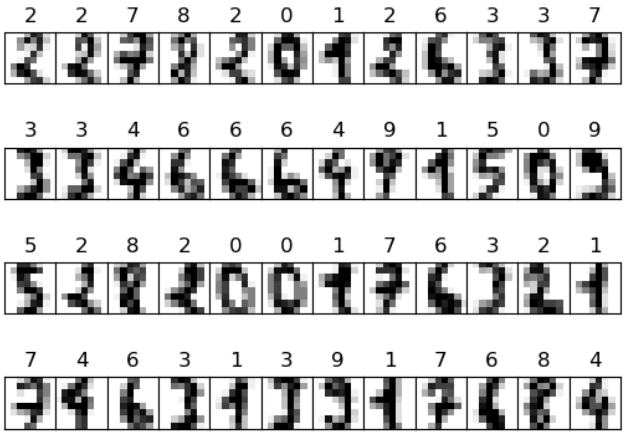


Figure – Chiffres manuscrits

- $d = 64$ (images 8×8) ; $n = 1797$; $|\mathcal{C}| = 10$ (les 10 chiffres)
- Classe `digits` de `sklearn.datasets`
- En TP ?

Exemple : détection de visages

Figure – Visages

- $d = 5828\,;\, n = 13233\,;\, |\mathcal{C}| = 5749$
- Classe `lfw` de `sklearn.datasets`
- Très difficile

Autres exemples de `sklearn.datasets`

- `wine` : détection de type de vin.

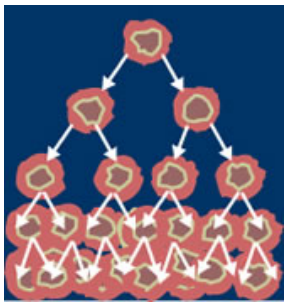
$d = 13$, $n = 178$, $|\mathcal{C}| = 3$ (3 types de vin).

Autres exemples de `sklearn.datasets`

- `wine` : détection de type de vin.

$d = 13$, $n = 178$, $|\mathcal{C}| = 3$ (3 types de vin).

- `breaset_cancer` : diagnostic de cancer du sein.



$d = 30$, $n = 569$, $|\mathcal{C}| = 2$ (bénin ou malin).

Algorithme k-NN

Algorithme k-NN

L'algorithme se découpe alors en trois phases :

1. Calculer la distance $d_i = \|x - x_i\|$ de x à chaque donnée x_i de la base (on peut se contenter de son carré).

Algorithme k-NN

Algorithme k-NN

L'algorithme se découpe alors en trois phases :

1. Calculer la distance $d_i = \|x - x_i\|$ de x à chaque donnée x_i de la base (on peut se contenter de son carré).
2. Trier par ordre croissant ces distances d_i pour obtenir $d_{i_1} \leqslant d_{i_2} \leqslant \cdots \leqslant d_{i_n}$ et ne garder que les k premiers indices $i_1, \ldots, i_k$.

Algorithme k-NN

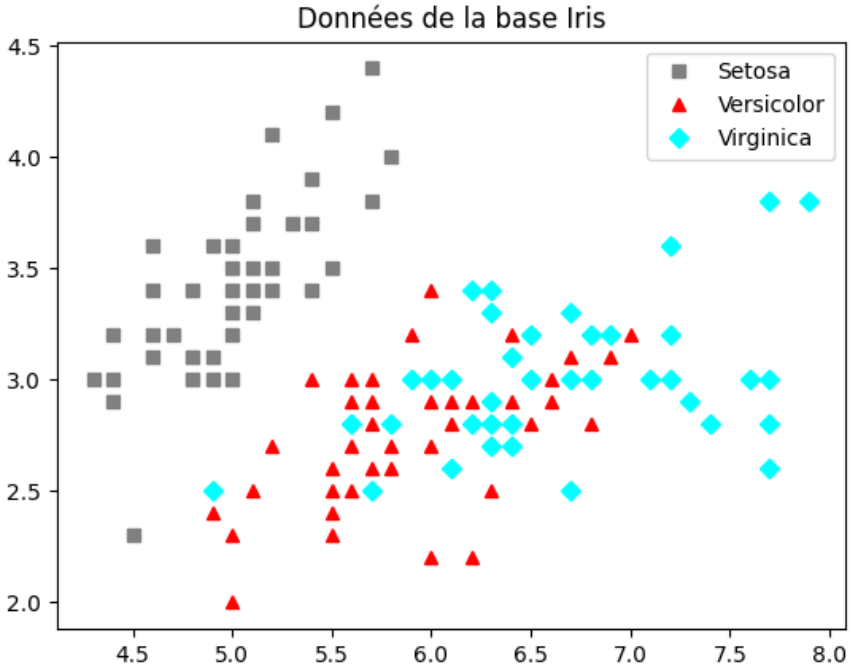
Algorithme k-NN

L'algorithme se découpe alors en trois phases :

1. Calculer la distance $d_i = \|x - x_i\|$ de x à chaque donnée x_i de la base (on peut se contenter de son carré).
2. Trier par ordre croissant ces distances d_i pour obtenir $d_{i_1} \leqslant d_{i_2} \leqslant \cdots \leqslant d_{i_n}$ et ne garder que les k premiers indices $i_1, \ldots, i_k$.
3. Déterminer la classe majoritaire parmi les $(c_{i_j})_{1 \leqslant j \leqslant k}$. En cas d'égalité, il faut décider d'un critère pour départager, par exemple en pondérant par un cœfficient inversement proportionnel à la distance à x. Lorsqu'il n'y a que deux classes, il peut aussi être intéressant de choisir k impair pour éviter les égalités.

Données de la bases Iris

On teste l'algorithme sur la base de donnée Iris : chaque fleur de la base est représentée par un point de coordonnée (longueur des sépales, largeur des sépales), dont la couleur correspond à l'espèce (setosa, versicolor ou virginica).



7-NN

Ici, $d = 2$. On teste, avec une nouvelle fleur, l'algorithme des 7 plus proches voisins. Voici le résultat.

7-NN

Ici, $d = 2$. On teste, avec une nouvelle fleur, l'algorithme des 7 plus proches voisins. Voici le résultat.

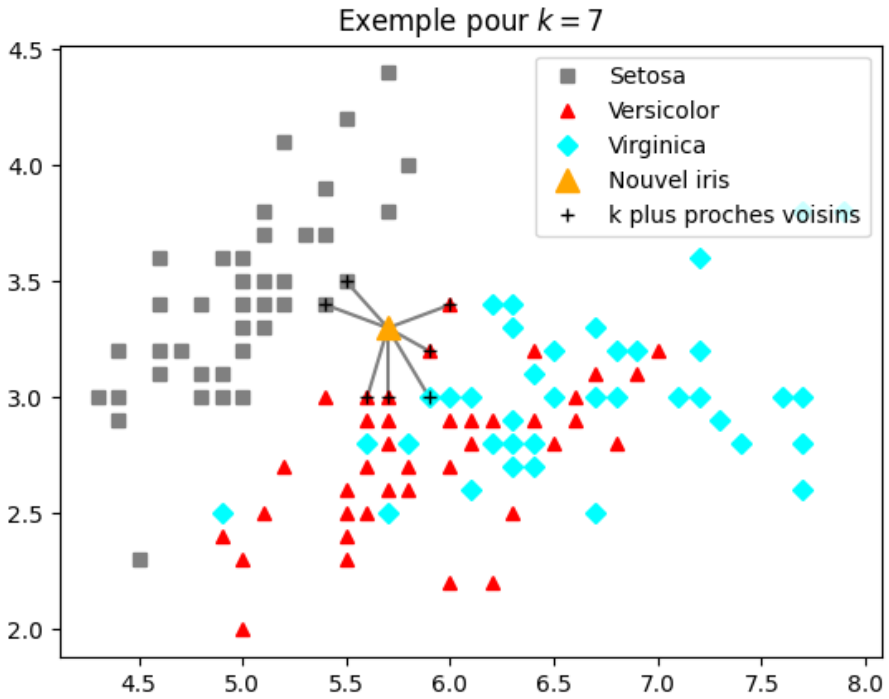
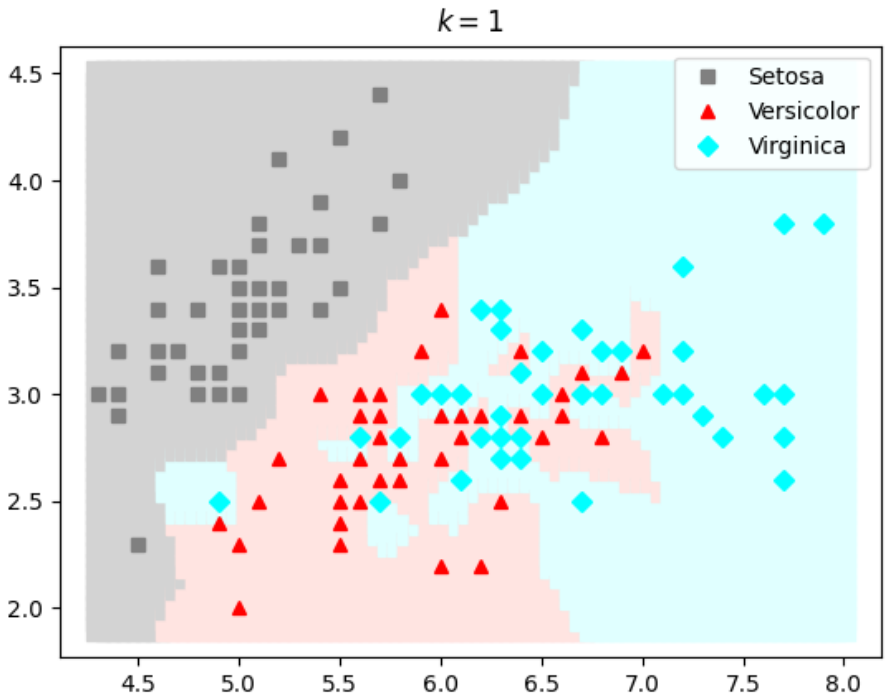


Figure – Application de 7-NN

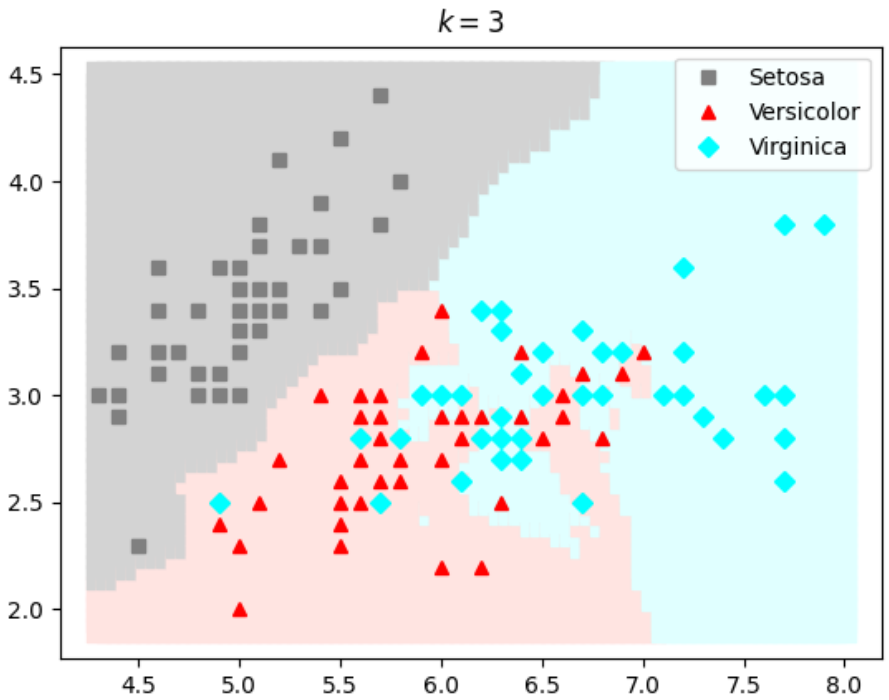
Comparaison pour différentes valeurs de k

Voici, pour différentes valeurs de k $(1, 3, 7, 13, 25, 31)$, le résultat de l'algorithme en chaque point :



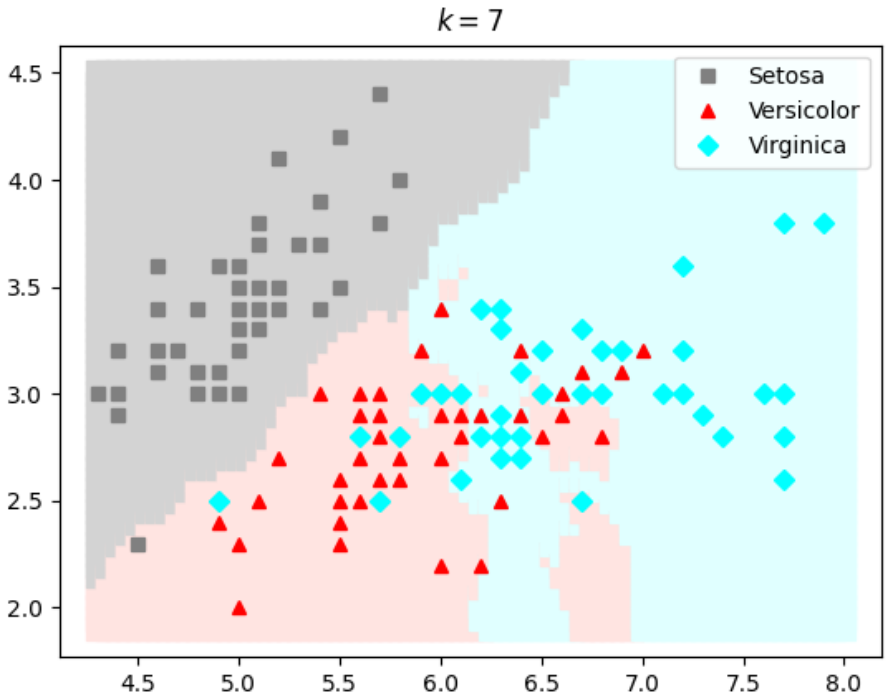
Comparaison pour différentes valeurs de k

Voici, pour différentes valeurs de k $(1, 3, 7, 13, 25, 31)$, le résultat de l'algorithme en chaque point :



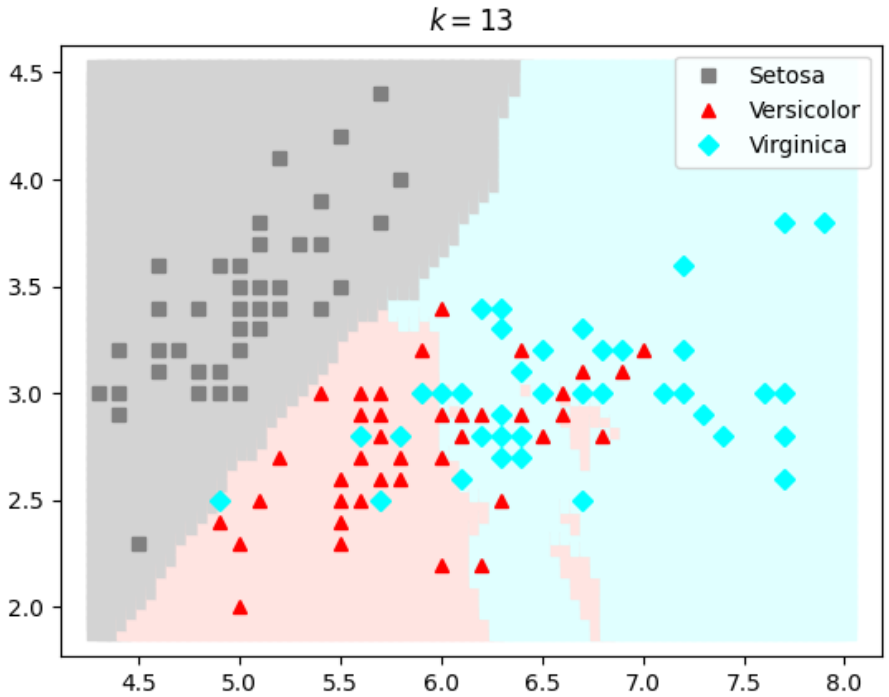
Comparaison pour différentes valeurs de k

Voici, pour différentes valeurs de k $(1, 3, 7, 13, 25, 31)$, le résultat de l'algorithme en chaque point :



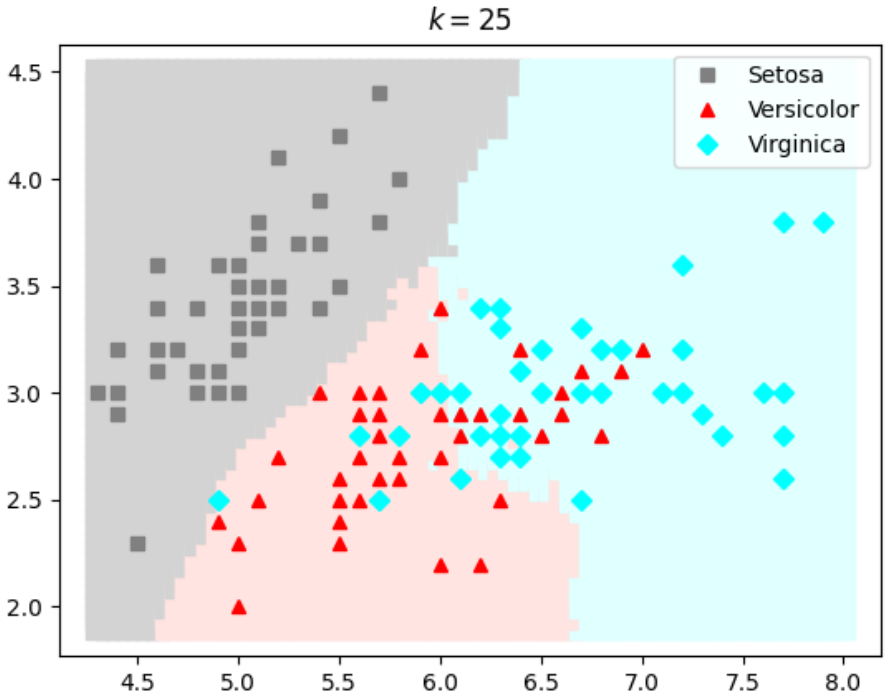
Comparaison pour différentes valeurs de k

Voici, pour différentes valeurs de k $(1, 3, 7, 13, 25, 31)$, le résultat de l'algorithme en chaque point :



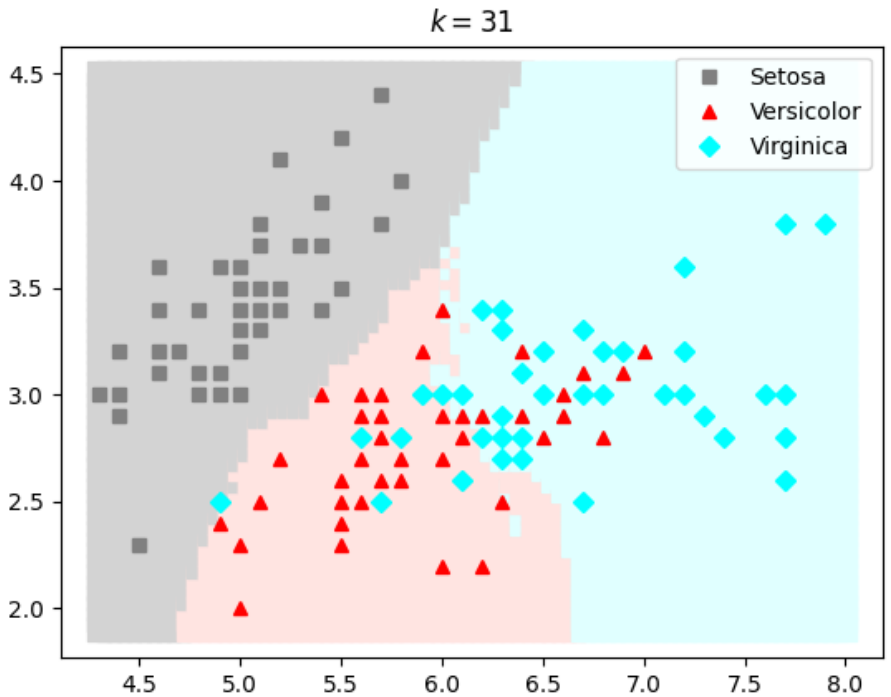
Comparaison pour différentes valeurs de k

Voici, pour différentes valeurs de k $(1, 3, 7, 13, 25, 31)$, le résultat de l'algorithme en chaque point :



Comparaison pour différentes valeurs de k

Voici, pour différentes valeurs de k $(1, 3, 7, 13, 25, 31)$, le résultat de l'algorithme en chaque point :



Évaluation de la qualité de l'algorithme

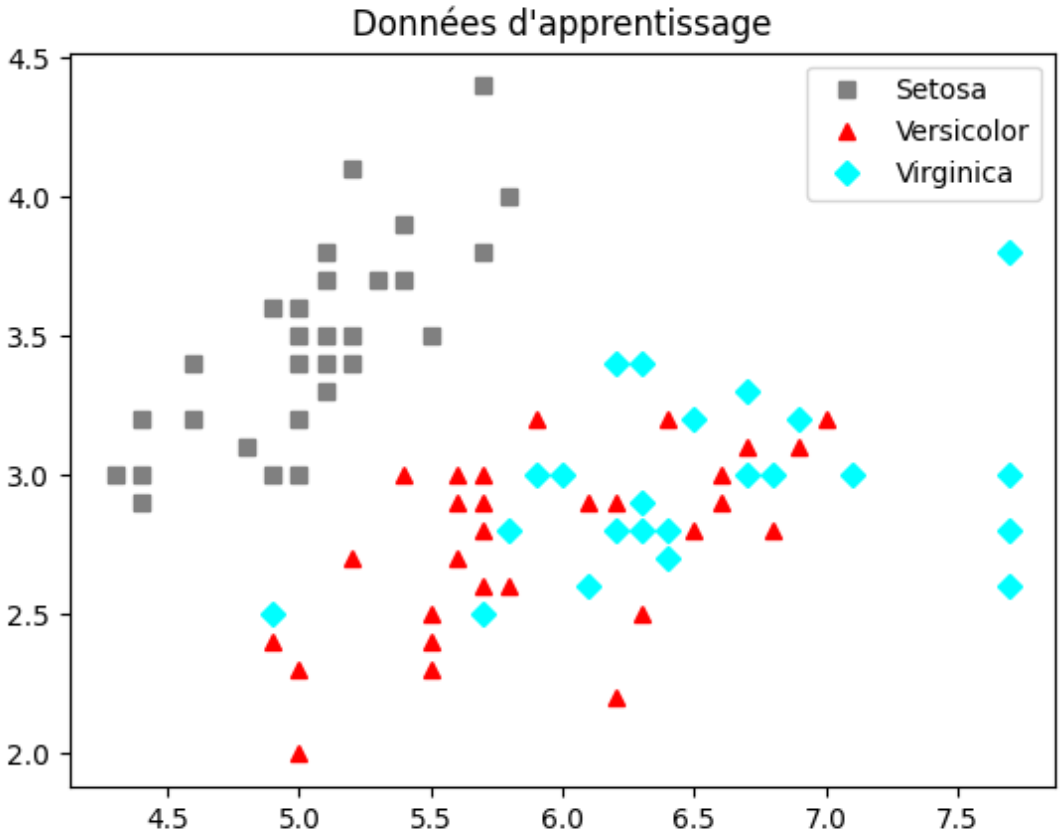
Pour pouvoir évaluer la qualité de l'algorithme, on n'utilise pas l'intégralité des données pour l'apprentissage.

Évaluation de la qualité de l'algorithme

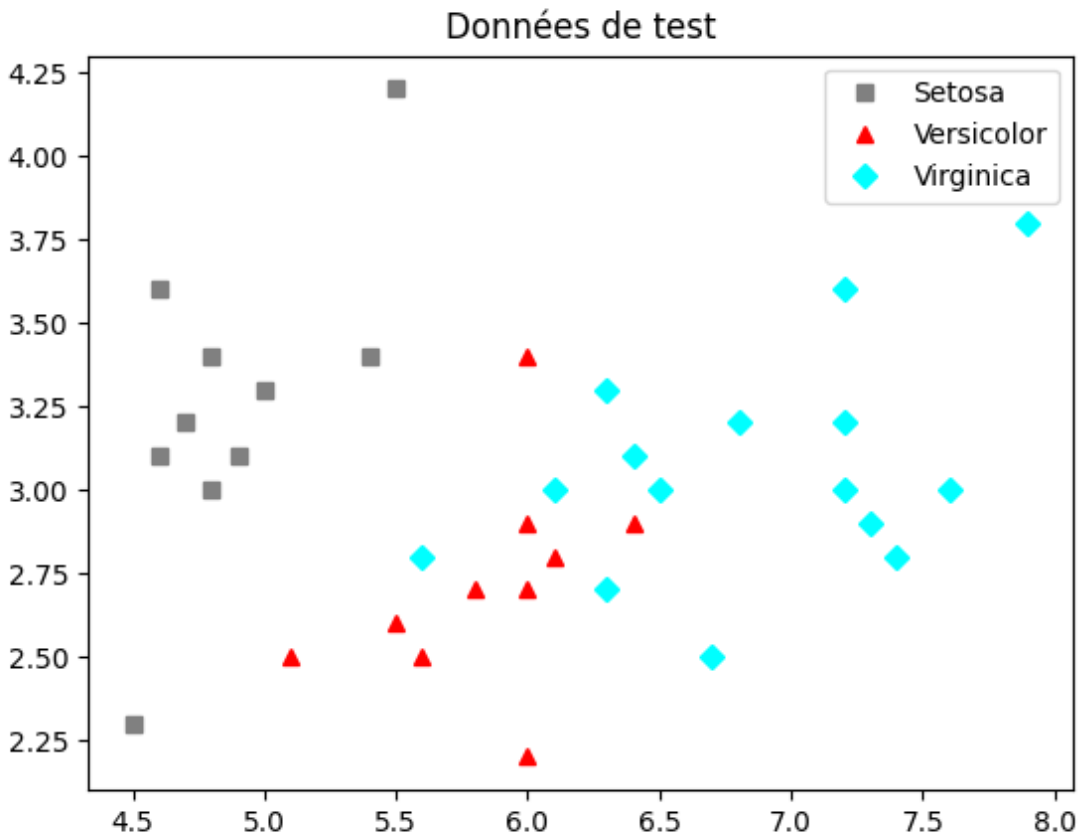
Pour pouvoir évaluer la qualité de l'algorithme, on n'utilise pas l'intégralité des données pour l'apprentissage.

On va scinder ces données en une partie pour l'apprentissage, et une partie pour les tests. La règle est de ne jamais effectuer les tests sur les données d'apprentissage directement. On peut par exemple garder $70\,\%$ des données pour l'apprentissage et $30\,\%$ pour les tests.

Évaluation de la qualité de l'algorithme



Évaluation de la qualité de l'algorithme



Matrice de confusion

Définition : Matrice de confusion

La **matrice de confusion** d'un résultat de classification sur une base de test est une matrice carrée d'ordre le nombre de classes, dont le cœfficient (i, j) compte le nombre d'observation pour laquelle la classe réelle est i et la classe estimée est j.

Matrice de confusion

Définition : Matrice de confusion

La **matrice de confusion** d'un résultat de classification sur une base de test est une matrice carrée d'ordre le nombre de classes, dont le cœfficient (i, j) compte le nombre d'observation pour laquelle la classe réelle est i et la classe estimée est j.

Ainsi, plus la matrice de confusion est proche d'une matrice diagonale, meilleur est l'apprentissage. Cela peut permettre de choisir une valeur de k bien adaptée.

$k=1$	**setosa**	**versicolor**	**virginica**
setosa	9	1	0
versicolor	0	5	5
virginica	0	6	9

$k=3$	**setosa**	**versicolor**	**virginica**
setosa	9	1	0
versicolor	0	5	5
virginica	0	6	9

$k=7$	**setosa**	**versicolor**	**virginica**
setosa	10	0	0
versicolor	0	5	5
virginica	0	4	11

$k = 13$	**setosa**	**versicolor**	**virginica**
setosa	10	0	0
versicolor	0	6	4
virginica	0	2	13

$k = 25$	**setosa**	**versicolor**	**virginica**
setosa	10	0	0
versicolor	0	9	1
virginica	0	3	12

$k = 31$	**setosa**	**versicolor**	**virginica**
setosa	10	0	0
versicolor	0	9	1
virginica	0	3	12

Code python

Python

```python
## Lecture des données dans un fichier csv
import csv
with open('iris.csv', newline='') as fichierIris:
    lecture = csv.reader(fichierIris, delimiter=',')
    # transformation en liste, omission de l'entête
    fleurs = list(lecture)[1:]

# On transforme les 2 premières données en float
for i, (long, larg, espece) in enumerate(fleurs):
    fleurs[i] =  float(long),  float(larg), espece
```

Code python

Python

```python
def distance(iris1, iris2):
    """
iris1 : élément de la liste lecture,
        forme [float, float]
        ou [float, float, espèce]
iris2 : même nature qu'iris1
renvoie la distance euclidienne *au carré*
entre les deux iris."""

    longueur1, largeur1 = iris1[:2]
    longueur2, largeur2 = iris2[:2]
    dx = longueur1 - longueur2
    dy = largeur1 - largeur2
    return dx ** 2 + dy ** 2
```

Code python

Python

```python
def distance(iris1, iris2):
    """
iris1 : élément de la liste lecture,
        forme [float, float]
        ou [float, float, espèce]
iris2 : même nature qu'iris1
renvoie la distance euclidienne *au carré*
entre les deux iris."""

    longueur1, largeur1 = iris1[:2]
    longueur2, largeur2 = iris2[:2]
    dx = longueur1 - longueur2
    dy = largeur1 - largeur2
    return dx ** 2 + dy ** 2
```

Ici, on a choisi d'utiliser les facilités offertes par python. Pas nécessairement possible aux concours !

Code python

Python

```python
def kPlusProchesVoisins(iris, k, fleurs):
    """iris : liste de type [float, float]
k : entier entre 1 et le nombre total de fleurs
fleurs : liste de toutes les fleurs
renvoie la liste des k plus proches voisins de iris"""
    nb_fleurs = len(fleurs)
    # Liste des dist. entre iris et ttes les fleurs
    liste_distances = [distance(iris, fleur)
                                   for fleur in fleurs]
    # Listes des indices de toutes les fleurs
    indices = list(range(nb_fleurs))
    # On la trie suivant les distances croissantes.
    indices.sort(key=lambda i: distance(iris,
                                        fleurs[i]))
    # On récupère k premiers indices.
    I = indices[:k]
    # On renvoie la liste des fleurs corresp.
    return [fleurs[k] for k in I]
```

Code `python`

Python

```python
def classification(iris, k, fleurs):
    """iris : liste de type [float, float]
k : entier entre 1 et le nombre total de fleurs
fleurs : liste de toutes les fleurs
renvoie l'espèce la plus fréquente parmi les k plus
proches voisins de iris."""
    # Dictionnaire de comptage des espèces
    compte = {'Iris-setosa': 0,
              'Iris-versicolor': 0,
              'Iris-virginica': 0}
    # Liste des k plus proches voisins
    ppv = kPlusProchesVoisins(iris, k, fleurs)
    # Mise à jour compteurs
    for _, _, espece in ppv:
        compte[espece] += 1
    # On renvoie lespèce réalisant majoritaire.
    return max(compte, key=lambda espece: compte[espece])
    # ou bien : max(compte, key=compte.get)
```

Plan

Présentation

L'algorithme que nous allons présenter dans cette partie est un algorithme d'apprentissage non supervisé : l'idée est de **partitionner des données** (en anglais : **clustering**) sans avoir de données déjà classées a priori.

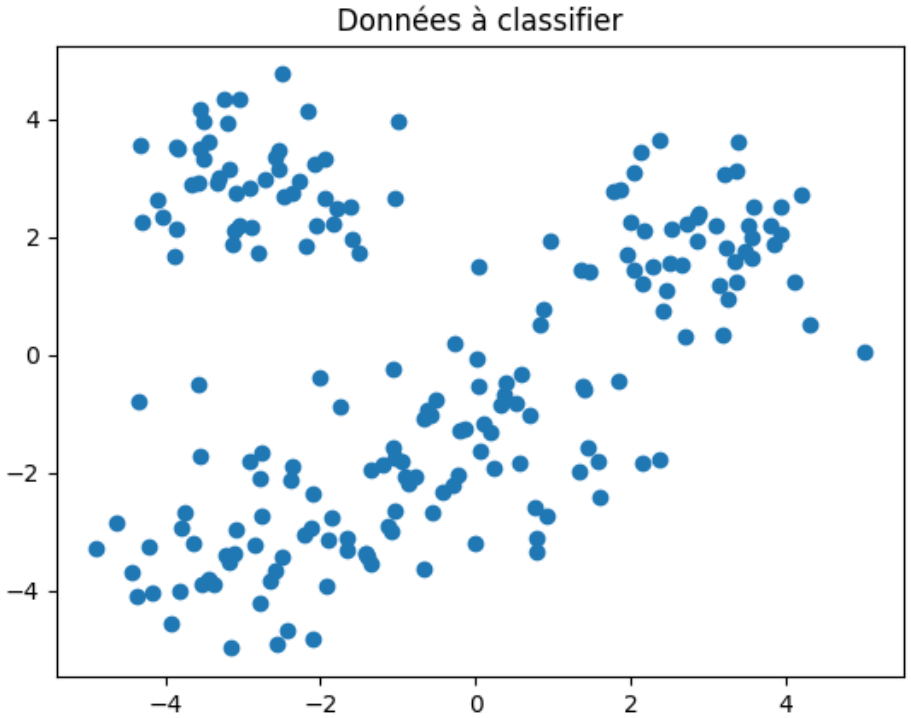
Présentation

L'algorithme que nous allons présenter dans cette partie est un algorithme d'apprentissage non supervisé : l'idée est de **partitionner des données** (en anglais : **clustering**) sans avoir de données déjà classées a priori.

En pratique, il s'agit pour la machine de regrouper les données proches au sein d'une même classe, à charge pour l'humain d'interpréter ensuite ce regroupement.

Présentation

Un exemple où les données (des points de $\mathbb{R}^2$, ici) semblent se regrouper en trois classes :



Centres, variances et inertie

Définition : Centre

Le **centre** ou **isobarycentre** d'un ensemble X de n vecteurs est le vecteur

$$\overline{X} = \frac{1}{n} \sum_{x \in X} x.$$

Centres, variances et inertie

Définition : Centre

Le **centre** ou **isobarycentre** d'un ensemble X de n vecteurs est le vecteur

$$\overline{X} = \frac{1}{n} \sum_{x \in X} x.$$

Exercice :

Écrire une fonction `centre(X)` renvoyant le centre de la liste éventuellement vide de vecteurs `X` (de même dimension `dim` : une variable globale).

Centres, variances et inertie

Dans toute la suite, on note d la distance euclidienne associée à la norme euclidienne $\|\cdot\|$.

Centres, variances et inertie

Dans toute la suite, on note d la distance euclidienne associée à la norme euclidienne $\|\cdot\|$.

Définition : Variance

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

$$V(X) = \sum_{x \in X} \mathrm{d}\left(x, \overline{X}\right)^2 = \sum_{x \in X} \left\|x - \overline{X}\right\|^2.$$

Centres, variances et inertie

Dans toute la suite, on note d la distance euclidienne associée à la norme euclidienne $\|\cdot\|$.

Définition : Variance

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

$$V(X) = \sum_{x \in X} \mathrm{d}\left(x, \overline{X}\right)^2 = \sum_{x \in X} \left\|x - \overline{X}\right\|^2.$$

La variance est un indicateur de dispersion : plus les points de X seront proche de leur centre $\overline{X}$, plus cette variance sera faible.

Centres, variances et inertie

Définition : Inertie

L'**inertie** d'une partition (*clustering*) de X en k parties $X_1, \ldots, X_k$ (classes ou *clusters*) est définie par

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

Centres, variances et inertie

Définition : Inertie

L'**inertie** d'une partition (*clustering*) de X en k parties $X_1, \ldots, X_k$ (classes ou *clusters*) est définie par

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

L'objectif est alors de trouver une partition de X minimisant l'**inertie** I.

Centres, variances et inertie

Définition : Inertie

L'**inertie** d'une partition (*clustering*) de X en k parties $X_1, \ldots, X_k$ (classes ou *clusters*) est définie par

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

L'objectif est alors de trouver une partition de X minimisant l'**inertie** I.

On cherche donc une partition de X permettant d'avoir des données les plus proches possible du centre de leur classe.

Centres, variances et inertie

Définition : Inertie

L'**inertie** d'une partition (*clustering*) de X en k parties $X_1, \ldots, X_k$ (classes ou *clusters*) est définie par

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

L'objectif est alors de trouver une partition de X minimisant l'**inertie** I.

On cherche donc une partition de X permettant d'avoir des données les plus proches possible du centre de leur classe.

Le calcul de la classification optimale, c'est-à-dire minimisant la somme des moments d'inertie, est un problème très coûteux en temps, aussi allons nous employer un algorithme glouton. Ce dernier nous assurera d'obtenir in fine une « bonne » solution, mais pas forcément la meilleure.

L'algorithme des k-moyennes

Algorithme k-MEANS

L'algorithme se déroule avec les étapes suivantes.

1. Soit $c_1, \ldots, c_k$ des vecteurs (par exemple choisis aléatoirement).

L'algorithme des k-moyennes

Algorithme k-MEANS

L'algorithme se déroule avec les étapes suivantes.

1. Soit $c_1, \ldots, c_k$ des vecteurs (par exemple choisis aléatoirement).
2. Associer chaque donnée x à la classe X_i telle que $\mathrm{d}(x, c_i)$ soit minimale.

L'algorithme des k-moyennes

Algorithme k-MEANS

L'algorithme se déroule avec les étapes suivantes.

1. Soit $c_1, \ldots, c_k$ des vecteurs (par exemple choisis aléatoirement).
2. Associer chaque donnée x à la classe X_i telle que $\mathrm{d}(x, c_i)$ soit minimale.
3. Recalculer les centres des classes $c_i = \overline{X_i}$.

L'algorithme des k-moyennes

Algorithme k-MEANS

L'algorithme se déroule avec les étapes suivantes.

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

L'algorithme des k-moyennes

Algorithme k-MEANS

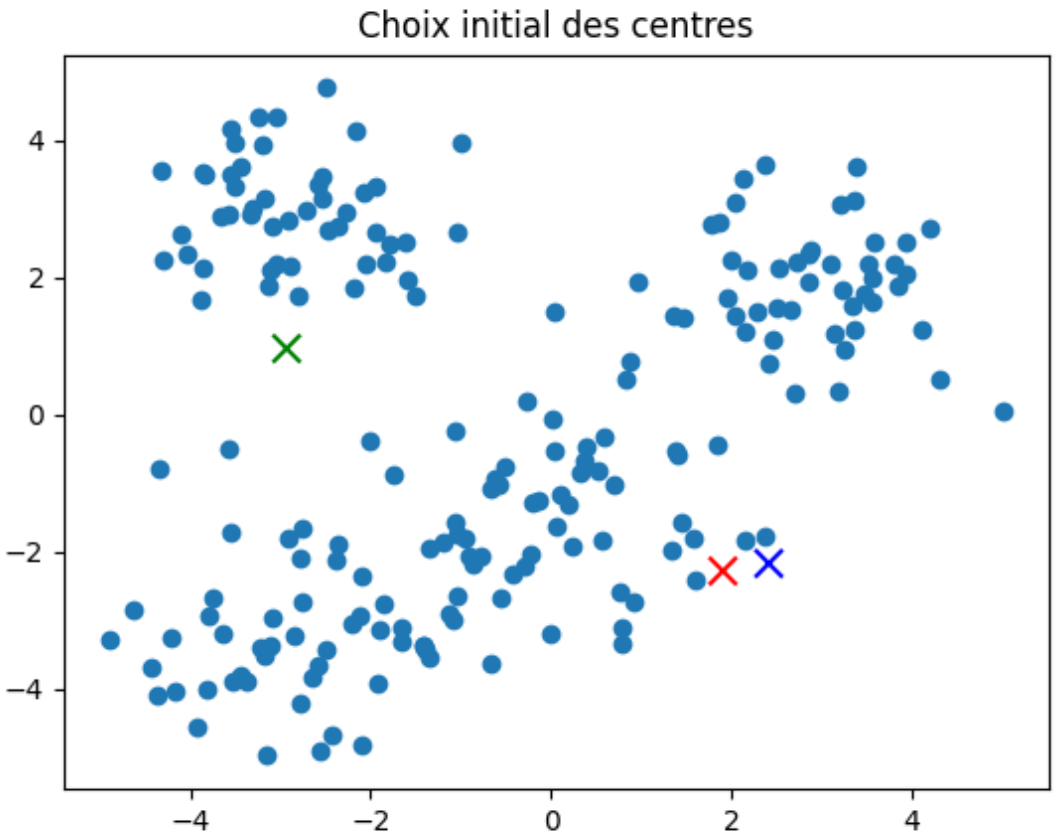
L'algorithme se déroule avec les étapes suivantes.

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

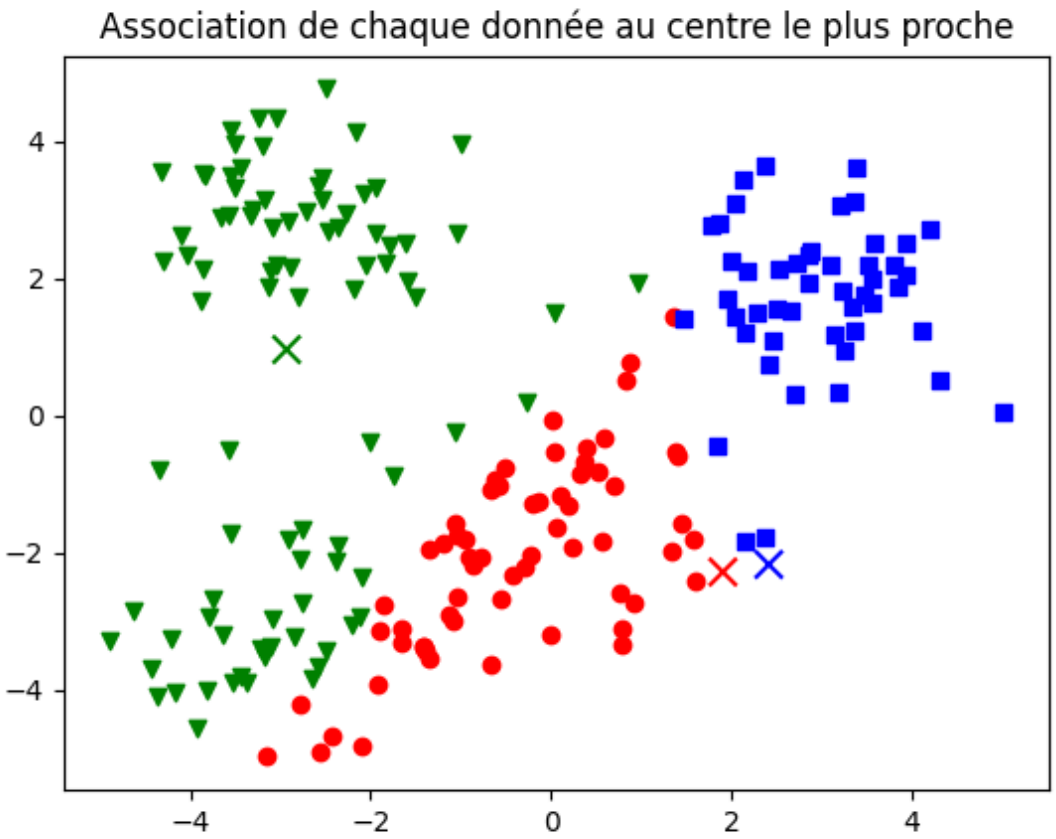
Remarque :

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

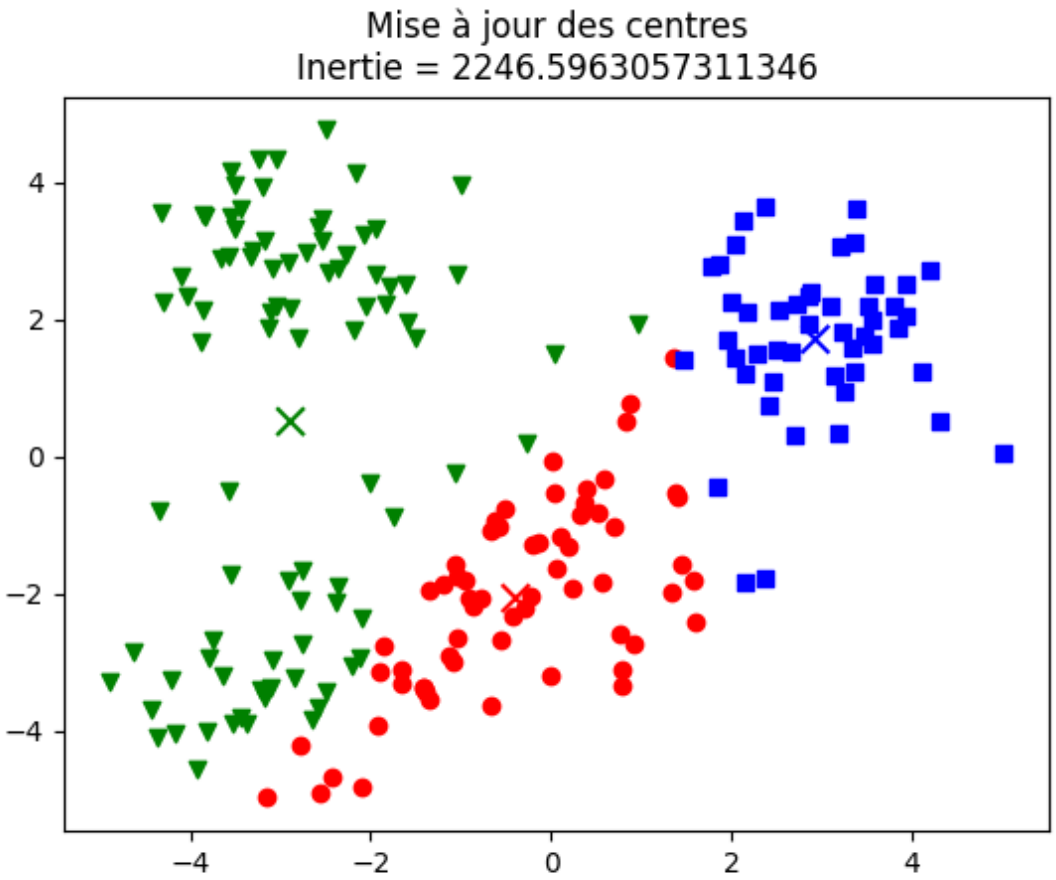
Exemple d'exécution de l'algorithme



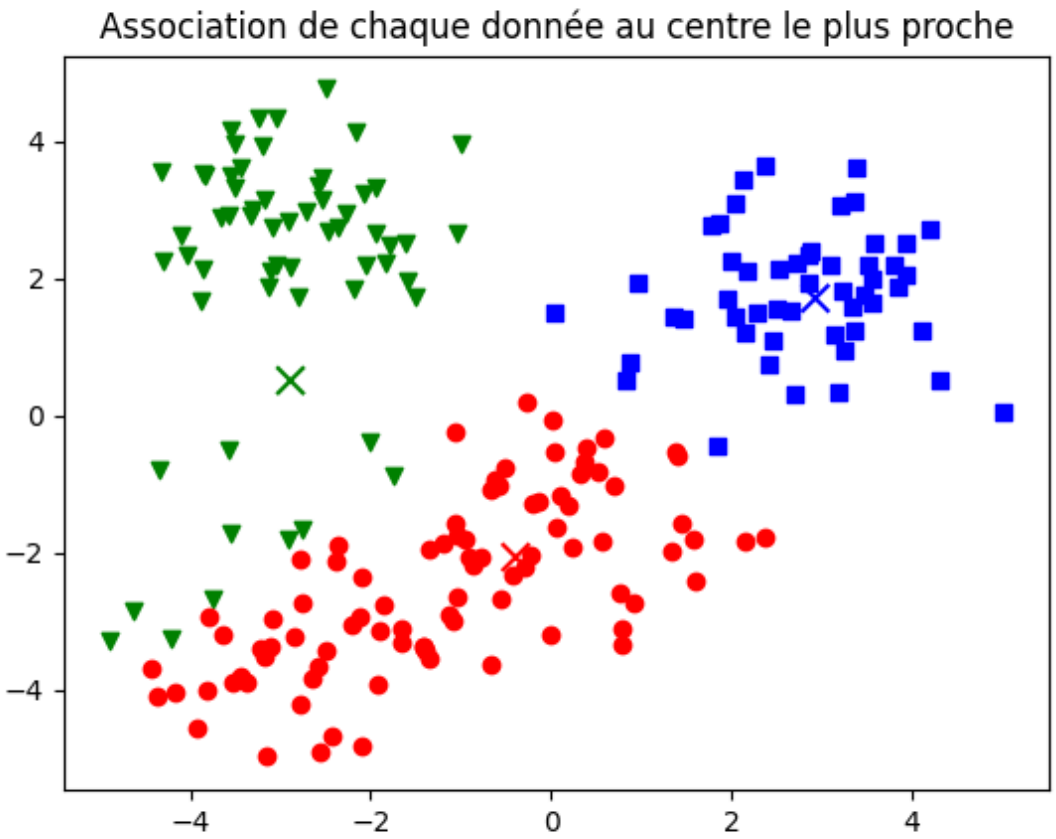
Exemple d'exécution de l'algorithme



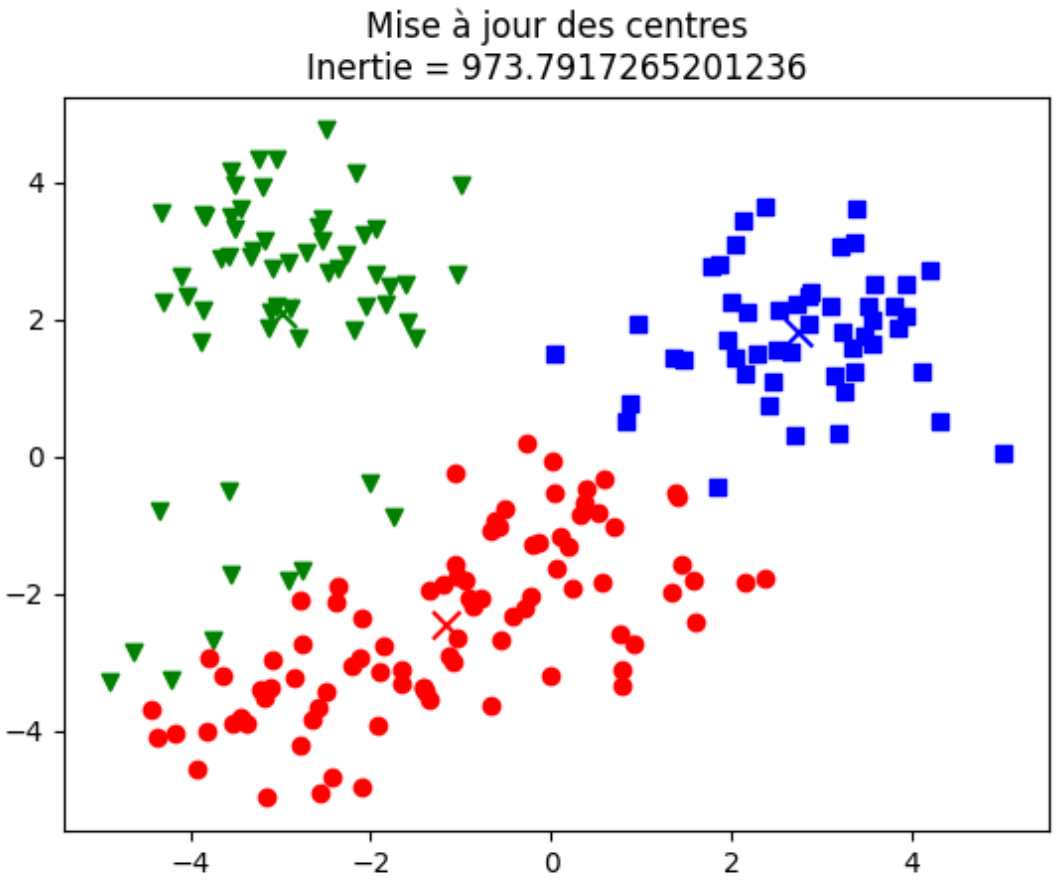
Exemple d'exécution de l'algorithme



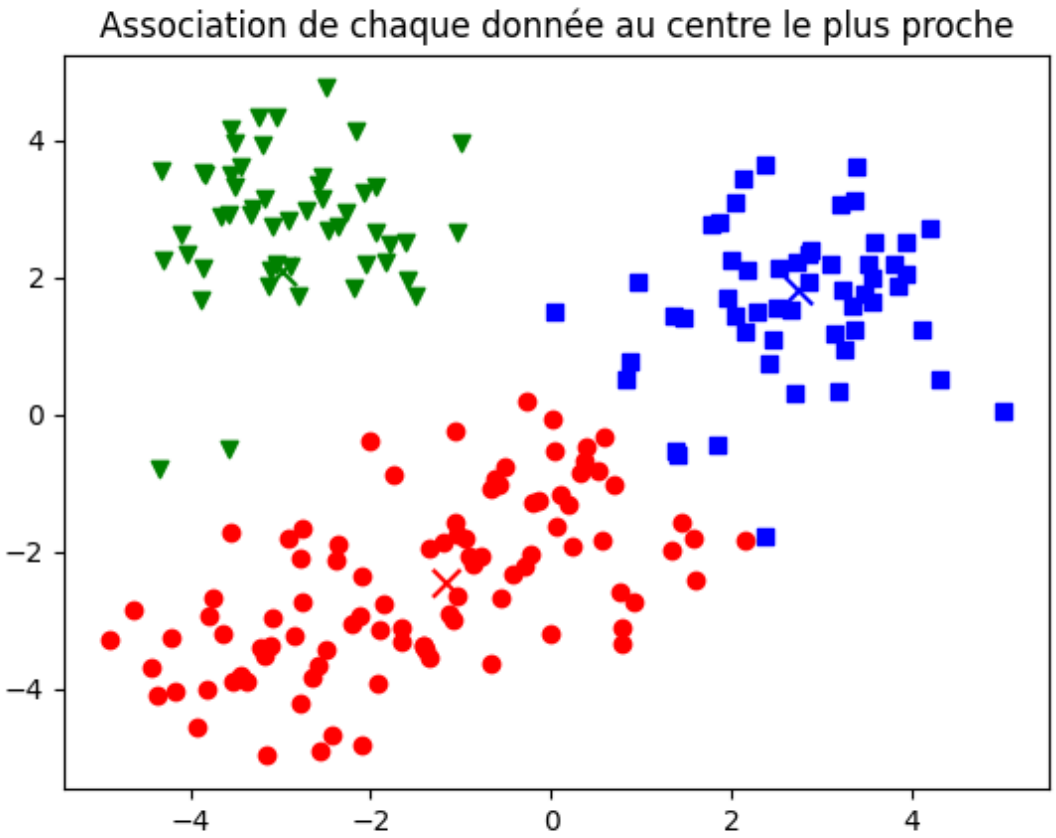
Exemple d'exécution de l'algorithme



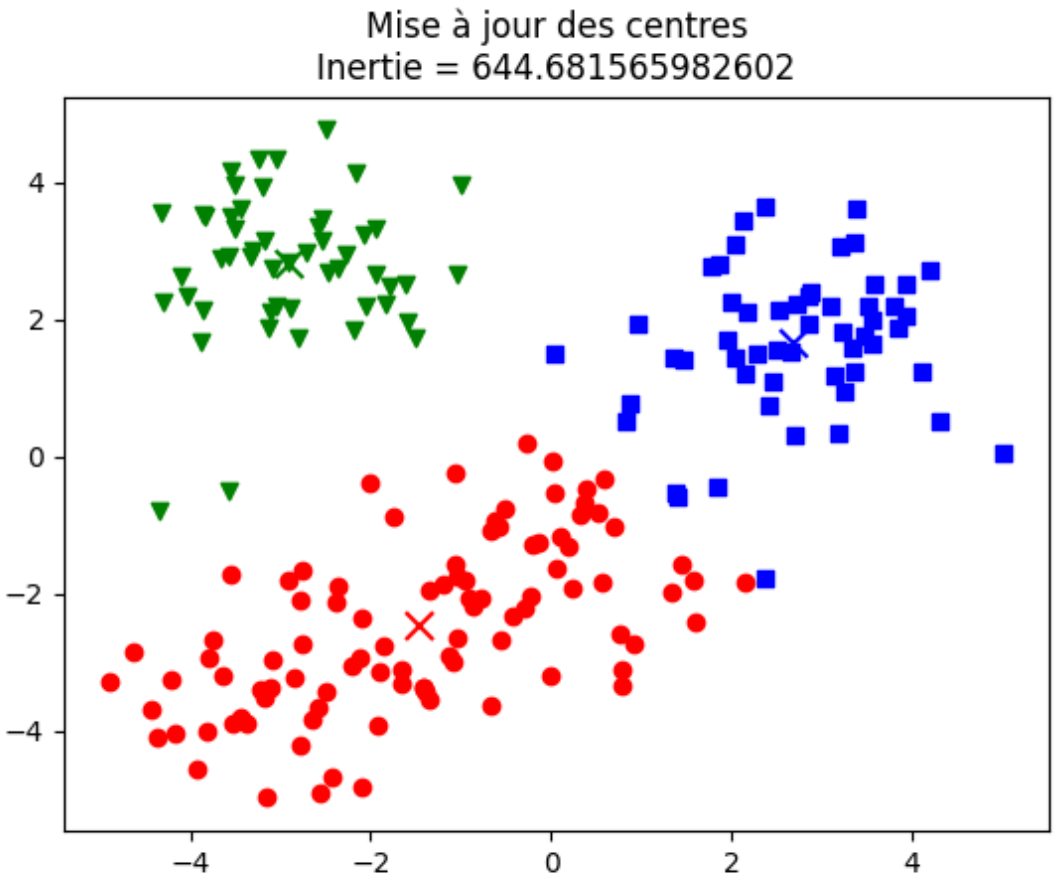
Exemple d'exécution de l'algorithme



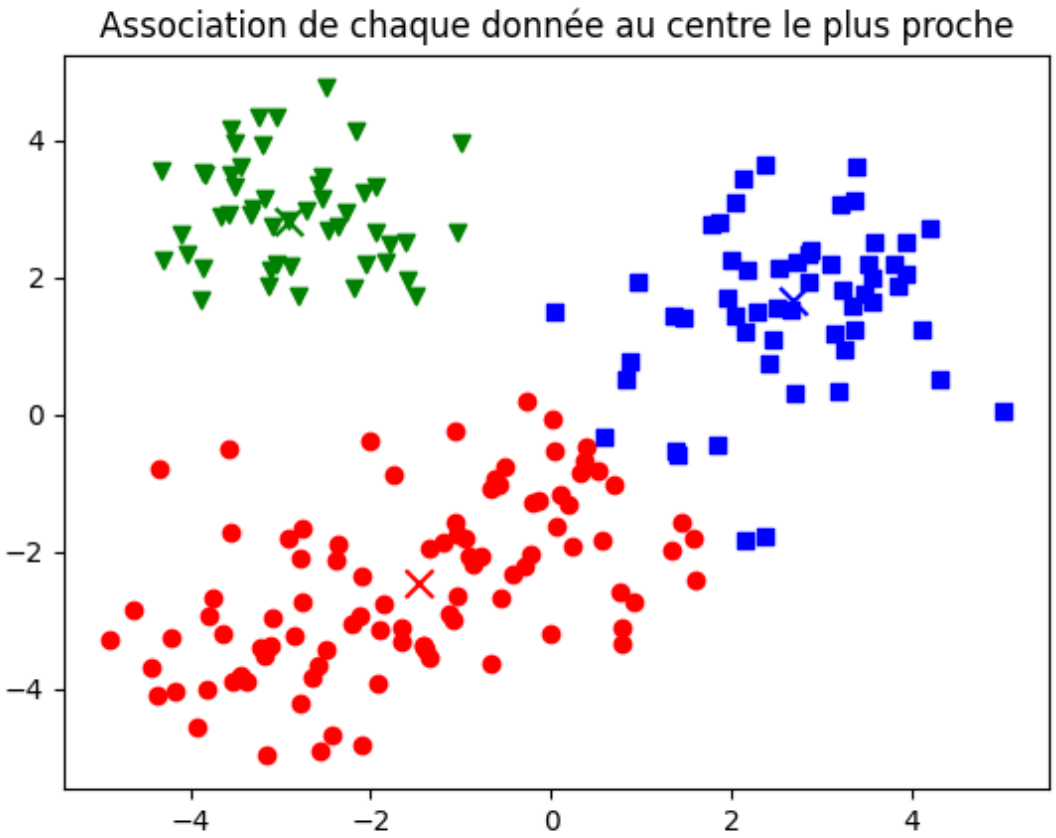
Exemple d'exécution de l'algorithme



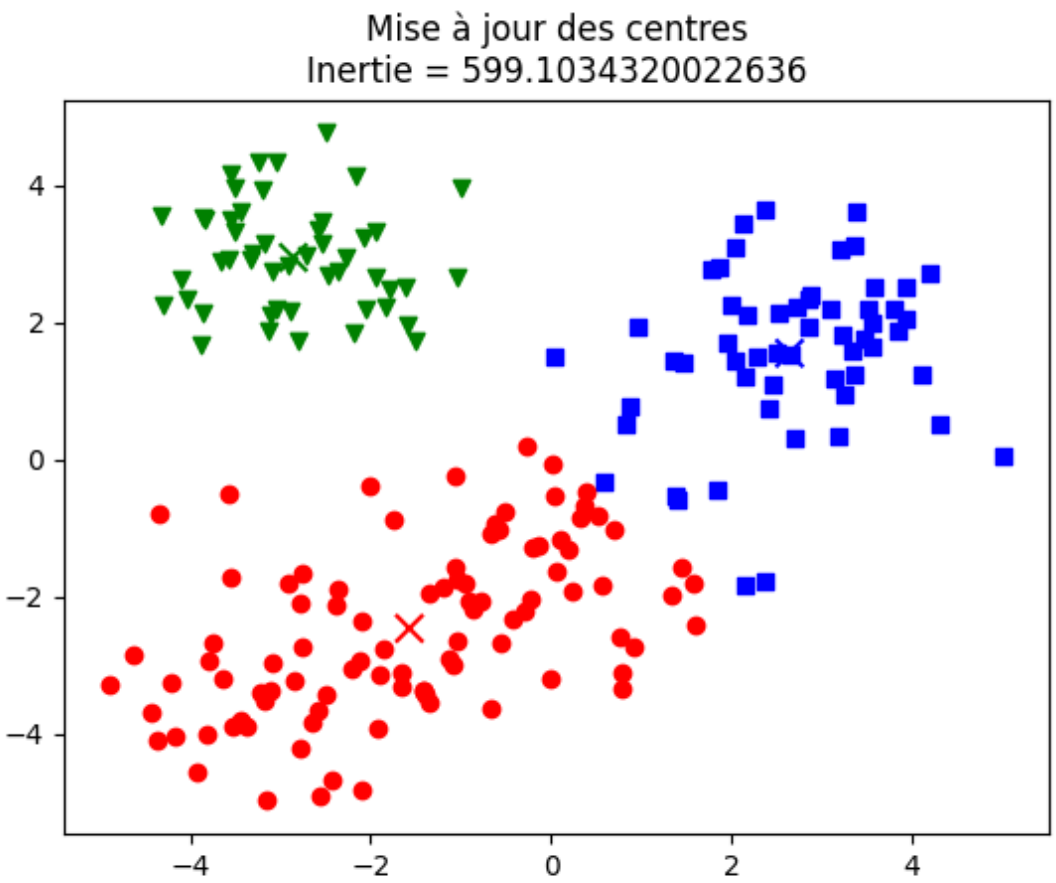
Exemple d'exécution de l'algorithme



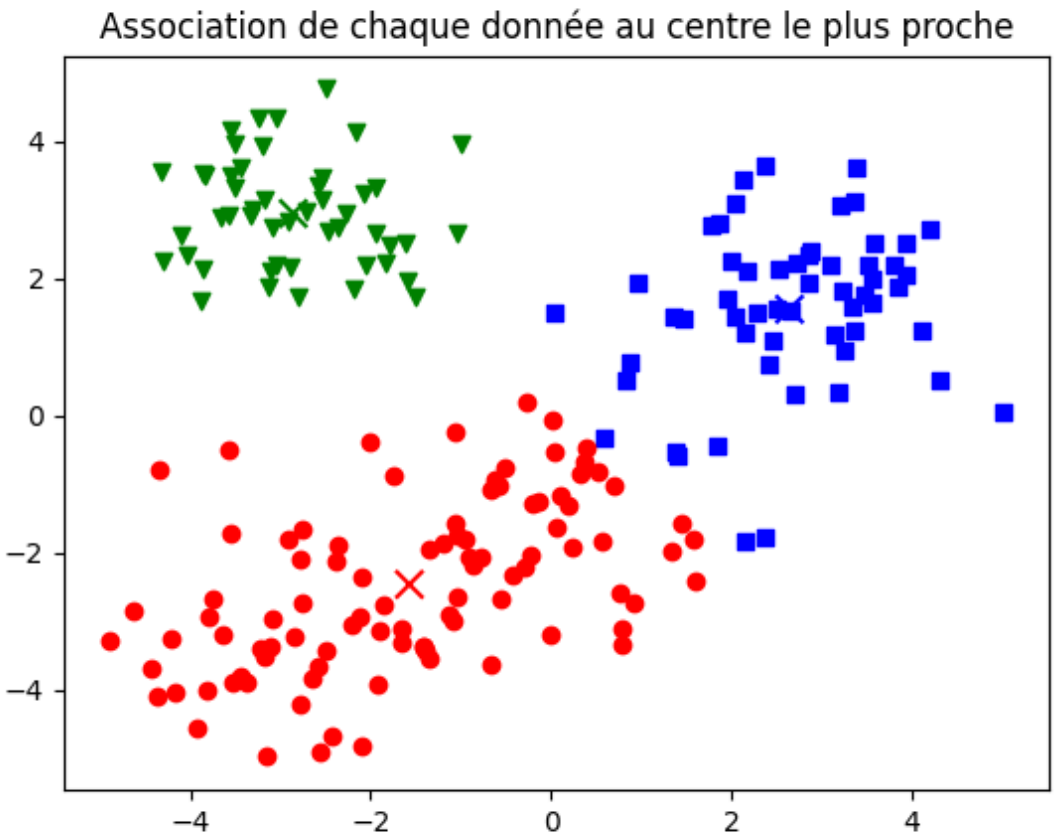
Exemple d'exécution de l'algorithme



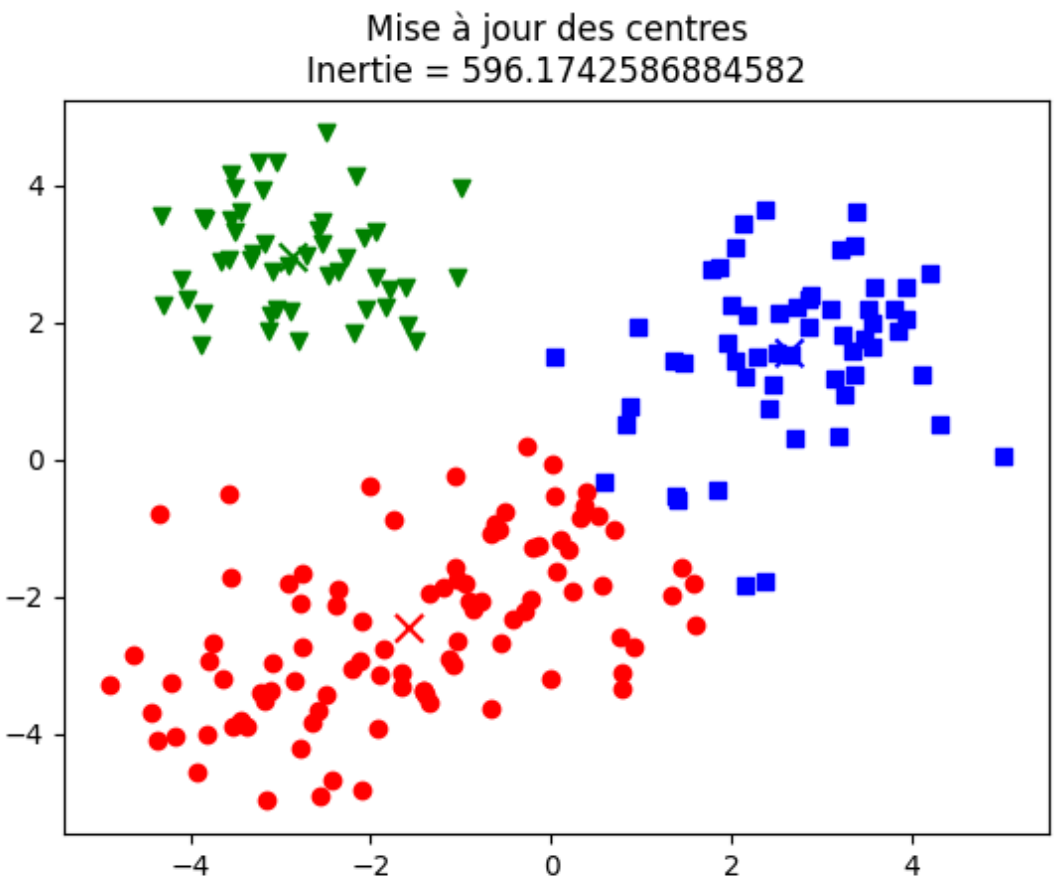
Exemple d'exécution de l'algorithme



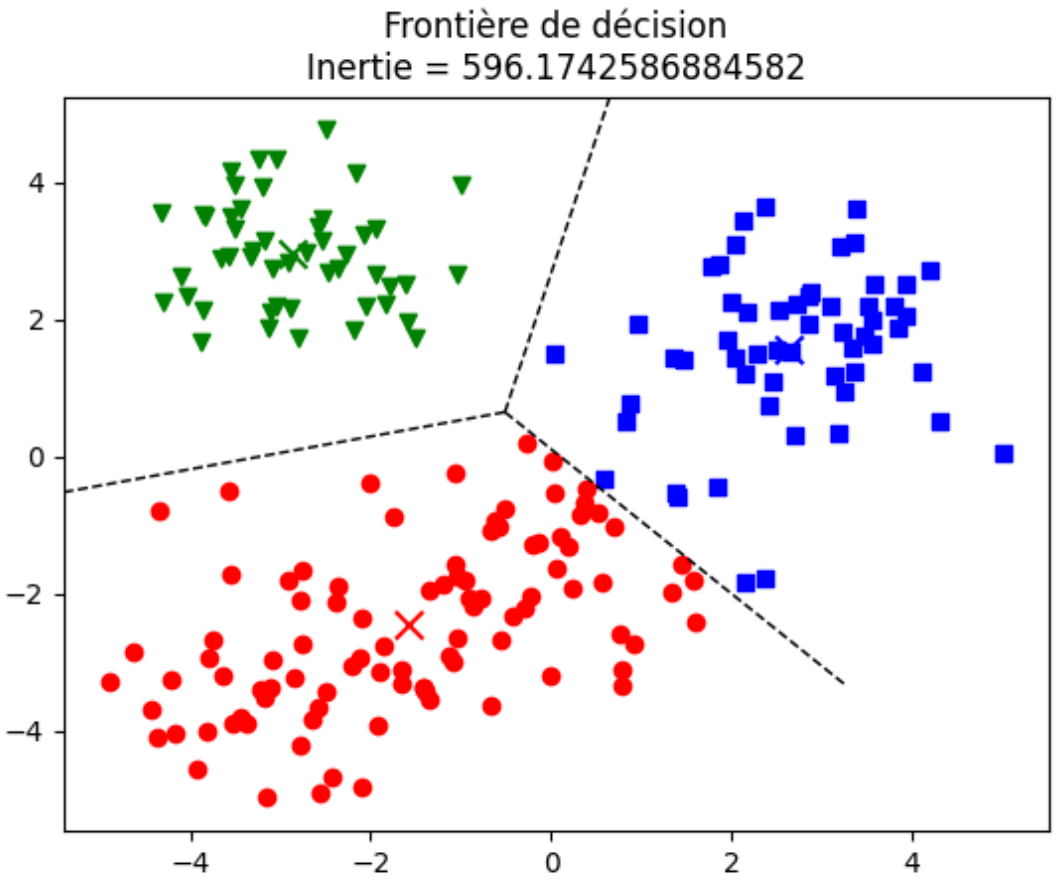
Exemple d'exécution de l'algorithme



Exemple d'exécution de l'algorithme



Résultat de l'exécution



Terminaison

Théorème : Hors programme

L'algorithme des k-moyennes termine.

Terminaison

Théorème : Hors programme

L'algorithme des k-moyennes termine.

Repose sur le fait qu'il n'y a qu'un nombre fini de k-partitions et que l'inertie, positive, décroît strictement.

Terminaison

Théorème : Hors programme

L'algorithme des k-moyennes termine.

Repose sur le fait qu'il n'y a qu'un nombre fini de k-partitions et que l'inertie, positive, décroît strictement.

Cependant, on n'est pas certain d'avoir atteint un minimum global de l'inertie, il s'agit peut-être d'un minimum local seulement.

Implémentation

Exercice :

On utilise une liste `classes` de taille k telle que `classes[i]` est la liste des données de `X` associées à la classe i.
Écrire une fonction `calculer_centre(classes)` renvoyant la liste des centres de chaque classe.

Implémentation

Exercice :

On utilise une liste `classes` de taille k telle que `classes[i]` est la liste des données de `X` associées à la classe i.
Écrire une fonction `calculer_centre(classes)` renvoyant la liste des centres de chaque classe.

Exercice :

Écrire des fonctions `d2(x, y)` calculant le carré de la distance euclidienne du vecteur `x` au vecteur `y` puis `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 que la distance de `x` à `centres[i]` soit minimale.

Exercice :

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 `centre[i]`.

Exercice :

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 `centre[i]`.

Exercice :

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

Choix des centres initiaux

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

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. Par exemple :

- Aléatoirement dans l'espace.

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. Par exemple :

- Aléatoirement dans l'espace.
- Aléatoirement parmi les données.

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. Par exemple :

- 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).

Choix de k

Comment choisir le nombre k de classes ?

Choix de k

Comment choisir le nombre k de classes ?

On peut calculer l'inertie obtenue pour différentes valeurs de k.

Choix de k

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).

Choix de k

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.

On peut utiliser la méthode du coude : on choisit la valeur de k la plus grande pour laquelle l'inertie diminue de façon significative (3 ou 4 ici).

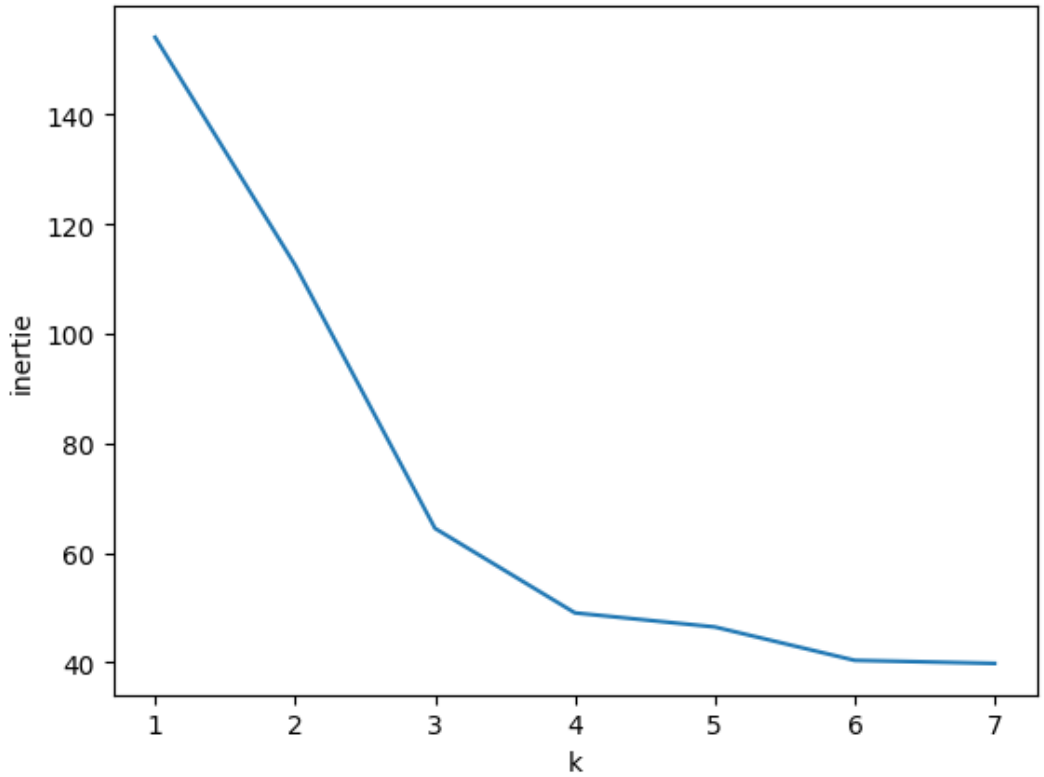


Figure – Méthode du coude (*elbow method*) donnant $k = 3$ ou 4

Limites

L'algorithme des k-moyennes ne fonctionne que sur des données linéairement séparable (c'est-à-dire pouvant être séparées par des hyperplans).