5. (Apprent)i(ss)a(ge)

Pierre Karpman

Lycée Champollion MP Tronc Commun

Avec plein d'images et d'exemples de Jérémy Larochette

https://membres-ljk.imag.fr/Pierre.Karpman/CPGE/2025/

Table des matières

Table des matières

« Intelligence artificielle »

« Intelligence artificielle »

Concept flou faisant référence à des traitements informatiques :

- Dans certains champs d'application auto-déclarés
  - jeux (vidéo)
  - traitement automatique des langues
  - reconnaissance de formes, d'objets
    - plein de thématiques en traitement du signal en général
  - résolution de systèmes booléens « SAT »
  - génération de texte (blabla), d'images, de programmes…
  - …
- Avec certains types d'approches
  - *Machine learning* (apprentissage)
    - approche statistique (régression linéaire (très) améliorée)
    - *k*-plus-proches-voisins ; *k*-moyennes ; …
  - « Réseaux de neurones »
  - …

Une famille de problèmes en IA (sous-branche : apprentissage) : classification

Problème de classification supervisée

- Pré-entrée : $\mathcal{C}$ ensemble de *classes* sur un certain type
  - Par ex. type « image » : {*images de chat*, *images de chien*}
- Pré-entrée : un ensemble d'« entraînement » $\mathcal{E}$ d'objets assortis de leur classe
- Entrée : x (du bon type)
- Sortie : un élément de $\mathcal{C}$, éventuellement avec degré de confiance
  - Par ex. « x est une image de chat avec confiance 0.97 »

Problème de classification non-supervisée

- Entrée : Un ensemble de données $\mathcal{D}$
- Entrée (optionnelle) : Un nombre de *clusters* (« groupes »)
- Sortie : une partition des éléments de D en plusieurs groupes

Table des matières

Algorithme des k-plus-proches-voisins

Algorithme $k - NN$

Soit :

- $\mathcal{C}$ des classes sur $\mathcal{T}$
- $\mathcal{E} = \{(t, c)\} \subseteq \mathcal{T} \times \mathcal{C}$ un ensemble d'entraînement
- δ une distance sur $\mathcal{T}$
- $k \in \mathbb{N}$
- $x \in \mathcal{T}$

L'algorithme « $k - NN$ » des k-plus-proches-voisins classe x comme la classe majoritaire parmi les k éléments de $\mathcal{E}$ les plus proches de x (pour δ)

Remarque

Famille d'algorithme plutôt qu'*un* algorithme

- Quelle distance ???
- Quelle valeur de k ?
- Besoin de $\mathcal{E}$

… ou plutôt famille d'*heuristiques* ?

Instanciation d'un $k - NN$

Quelques possibilités…

Distance

- Distance euclidienne ; euclidienne au carré…
- Avec prétraitement 🙈
  - Par ex. $\mathcal{T}$ image : appliquer une détection de contour ; une transformation de Fourier discrète… avant mesure de la distance
  - Projection dans un sous-espace
    - Utile pour $\mathcal{T}$ en grande dimension, avec des dimensions « corrélées »
    - Ex. *analyse en composante principale* (ACP) (projection sur les vecteurs propres les plus significatifs)
  - De façon générale : isoler des caractéristiques jugées « pertinentes »

Choix de k

- Par évaluation expérimentale des résultats (cf. ci-dessous)
- Variable (on ne se compare qu'aux voisins « suffisamment proches »)
  - Décale la question : quel seuil de proximité ?

Coût d'un $k - NN$

Coût « basique »

- $\#\mathcal{E}$ calculs de distance
- Plus éventuellement coûts annexes (par ex. calcul de DFT)

Utilisation d'une *locality-sensitive hash function* (LSH) 🙈

Idée :

- construire une fonction de hachage qui regroupe des éléments proches dans un même *bucket*
- seulement comparer x aux éléments du même *bucket*

Exemple de construction en grande dimension :

- Hash (identifiant d'un *bucket*) : projection sur un sous-espace aléatoire
  - Même principe qu'une ACP, mais applicable pour d'autres métriques ; sans analyse préalable

Exemple : reconnaissance de fleur

Une base Iris

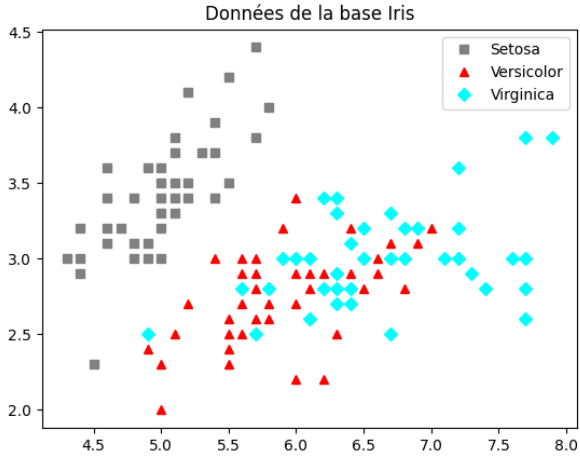
Objectif : distinguer trois types d'iris sur la base de 4 critères

- $\mathcal{C} = \{setosa, versicolor, virginica\}$
- $\mathcal{T} = \mathbb{R}^4$: longueur & largeur des pétales et des sépales

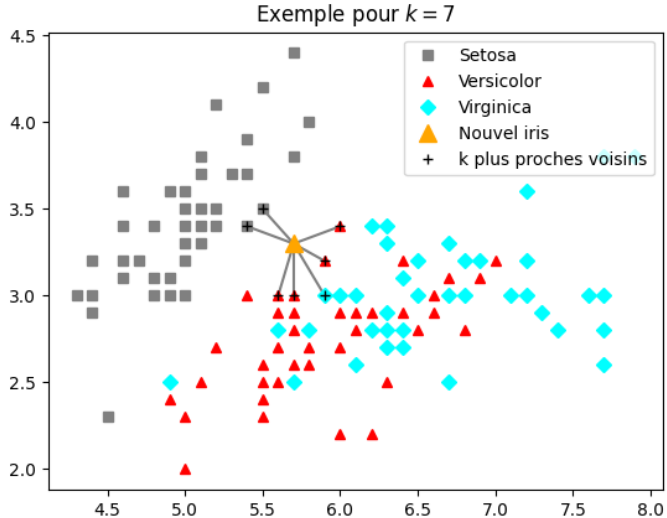
Figure: *iris virginica* (Frank Mayfield)

Visualisation : iris & $k - NN$ en deux dimensions

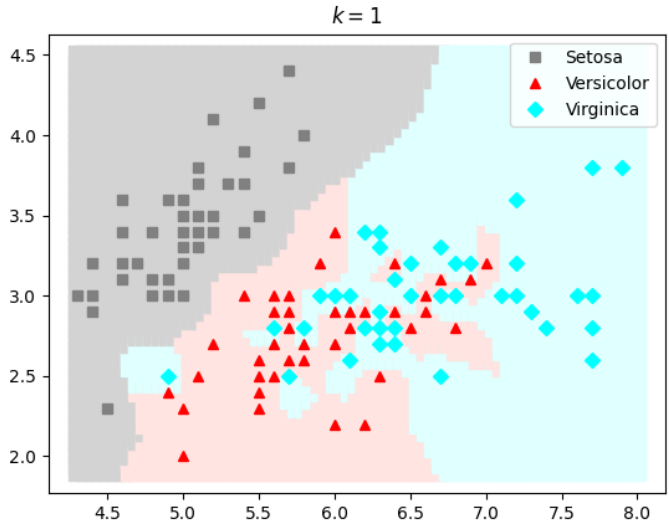
Représentation des éléments de $\mathcal{E}$ projetés sur les longueur (abscisse) & largeur (ordonnée) des sépales



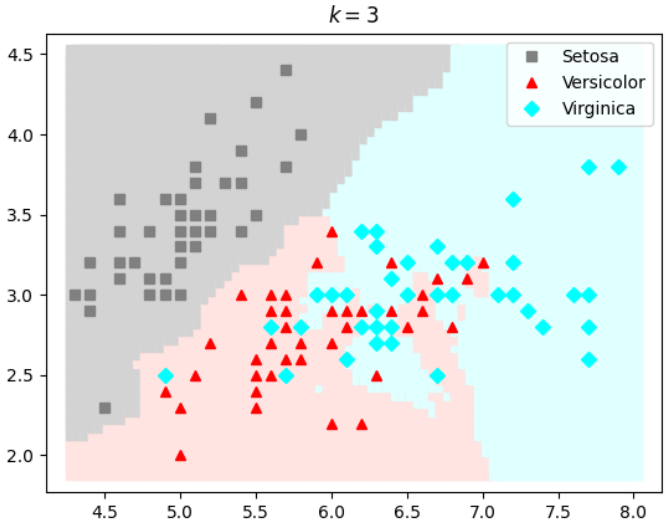
Exécution de 7 — *NN* avec une nouvelle fleur



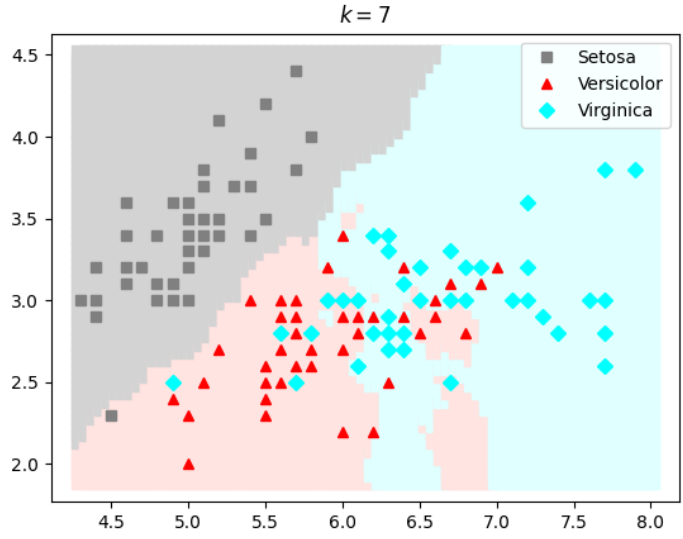
Partition de l'espace par $k - NN$ pour $k = 1$ (« diagramme de Voronoï »)



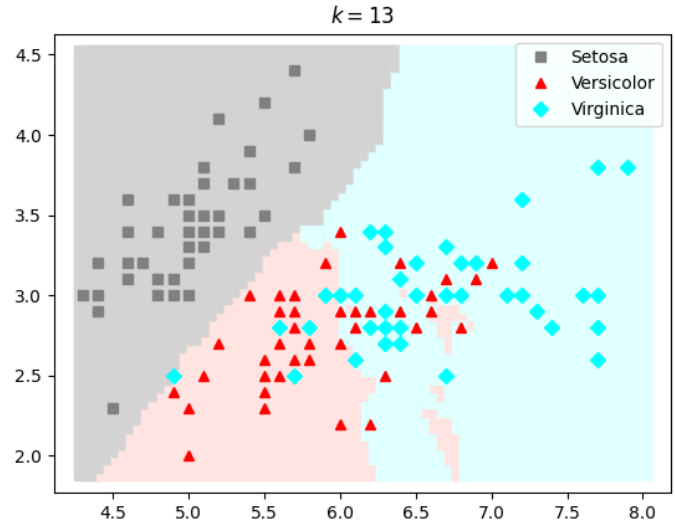
Partition de l'espace par $k - NN$ pour $k = 3$



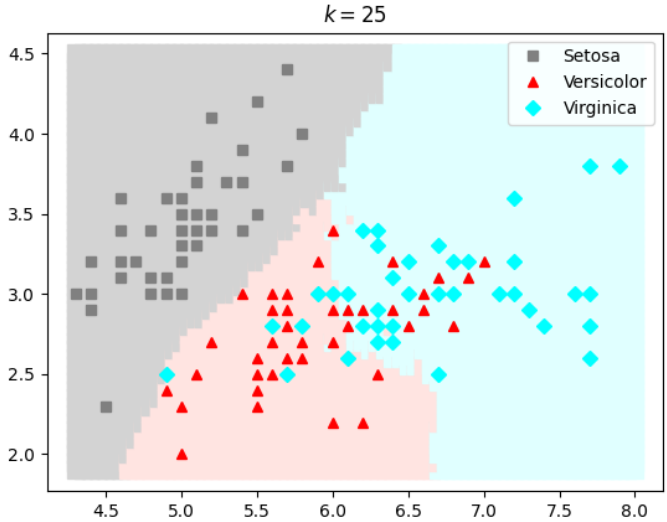
Partition de l'espace par $k - NN$ pour $k = 7$



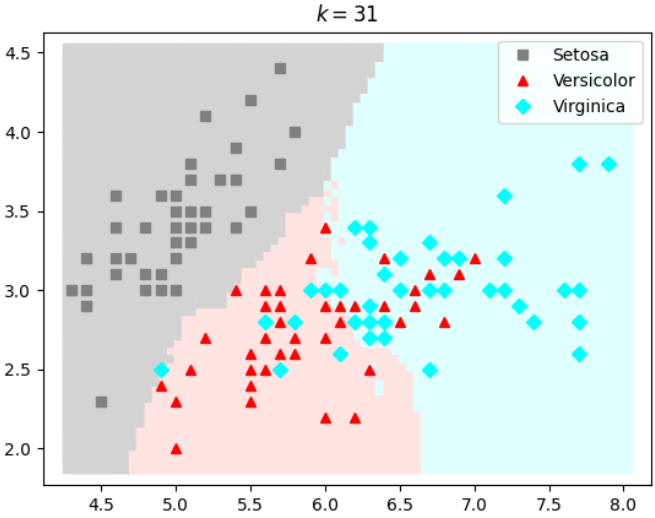
Partition de l'espace par $k - NN$ pour $k = 13$



Partition de l'espace par $k - NN$ pour $k = 25$



Partition de l'espace par $k - NN$ pour $k = 31$



Évaluation de la qualité de $k - NN$

Besoin d'une évaluation

Comme :

- On n'a pas de garanties *a priori* sur les résultats de $k - NN$
- On peut faire des choix d'instanciation

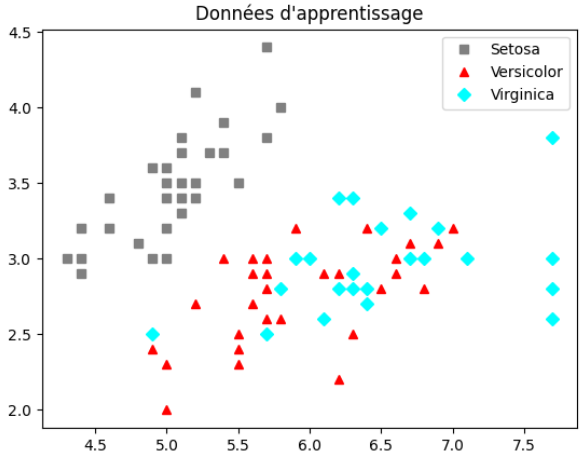
Il est important d'évaluer la qualité des résultats produits

Principe

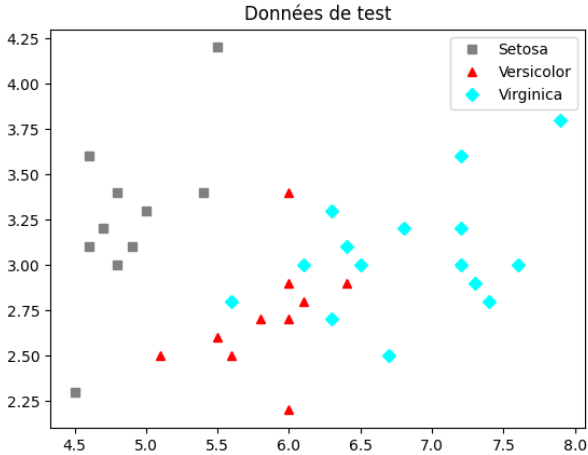
On divise $\mathcal{E}$ en deux parties :

- une partie d'apprentissage (pour exécuter l'algorithme, comme ci-dessus)
- une partie **différente** pour le test
  - on exécute l'algorithme sur des données pour lesquelles la réponse est connue, et on compare

Exemple Iris : exemple de données d'apprentissage



Exemple Iris : exemple de données de test



Matrice de confusion

Un outil pour l'évaluation avec données de test : la *matrice de confusion*

Matrice de confusion

Matrice de $\mathbb{N}^{c \times c}$, $c := \#\mathcal{C}$ le nombre de classes, définie pour :

- un certain algorithme de classification γ (par ex. un $k - NN$)
- un ensemble de test T

Par :

$$M_{i,j} := \#\{(c, i) \in T \mid \gamma(c) = j\}$$

Le nombre de réponses « j » par γ pour des données de la classe i

Matrice de confusion : interprétation

- Les bonnes réponses sont sur la diagonale
- Plus la matrice est « proche » de diagonale, meilleure est la classification
  - On ne fait que repousser le problème si on s'en tient là…
  - Mais permet néanmoins de résumer les résultats d'une campagne de test
  - Et de comparer différents choix !

Exemple Iris $k = 1, 3, 7$

$k = 1$	*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

Exemple Iris $k = 13, 25, 31$

$k = 13$	*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

Mini-conclusion $k - NN$

$k - NN$

- *Famille* d'algorithme ; beaucoup de choix/réglages à définir
  - Impact sur la qualité et/ou le coût
- Problématique non traitée : constitution des données d'apprentissage $\mathcal{E}$

Table des matières

Algorithme des k-moyennes

Algorithme k-moyennes (esquisse)

Soit :

- $\mathcal{D}$ un ensemble de données de type $\mathcal{T}$
- δ une distance sur $\mathcal{T}$ et $\|\cdot\|$ la norme associée
- $k \in \mathbb{N}$
- $\mathcal{X}$ un sous-ensemble $\{x_1, \ldots, x_k\}$ de $\mathcal{T}$ de taille k

L'algorithme des k-moyennes itère le processus suivant :

- pour chaque $x \in \mathcal{D}$, *classer* x dans le groupe $\mathcal{X}_i$ du $x_i \in \mathcal{X}$ le plus proche (pour δ)
- pour chaque $\mathcal{X}_i$, recalculer x_i comme la moyenne de ses éléments (cf. ci-dessous)
- calculer l'*inertie* de la partition (cf. ci-dessous)
- s'arrêter/continuer en fonction de la décroissance de l'inertie (cf. ci-dessous)

Remarque

Encore une fois, plutôt une famille d'heuristiques

Moyenne, inertie d'un groupe de vecteurs

Centre / (Iso)barycentre / Moyenne

Le *centre* d'un ensemble $\mathcal{X}$ de n vecteurs est défini comme le vecteur :

$$\widehat{\mathcal{X}} := \frac{1}{n} \sum_{x \in \mathcal{X}} x$$

Inertie d'un groupe

L'*inertie* ou *variance* (relativement à une distance δ / norme $\|\cdot\|$) d'un groupe $\mathcal{X}$ de vecteurs est définie par :

$$I(\mathcal{X}) := \sum_{x \in \mathcal{X}} \delta(x, \widehat{\mathcal{X}})^2 = \sum_{x \in \mathcal{X}} \|x - \widehat{\mathcal{X}}\|^2$$

où $\widehat{\mathcal{X}}$ est le centre du groupe

Inertie d'une partition ; minimisation

Inertie d'une partition

L'*inertie* d'une partition $\{\mathcal{X}_i\}_{1 \leq i \leq k}$ d'un ensemble de $\mathcal{D}$ vecteurs est la somme $\Sigma_{i=1}^{k} I(\mathcal{X}_i)$ des inerties des groupes de la partition

Minimisation de l'inertie

On utilise l'inertie d'une partition comme une mesure de la dispersion des données par rapport aux centres des groupes

- par analogie physique immédiate (moment d'inertie ; attraction gravitationnelle)

On *définit* (dans le contexte de l'algorithme des k-moyennes) une « bonne » partition comme ayant une inertie faible

Instanciation d'un k-moyennes

Quelques possibilités…

Distance

Pas vraiment de choix (pour nous) : distance euclidienne

Choix de k le nombre de groupes

- Par évaluation expérimentale des résultats (cf. ci-dessous)
- Compromis à trouver : $k = \#\mathcal{D}$ minimise trivialement l'inertie à zéro

Instanciation d'un *k*-moyennes *bis*

Initialisation de $\mathcal{P}$

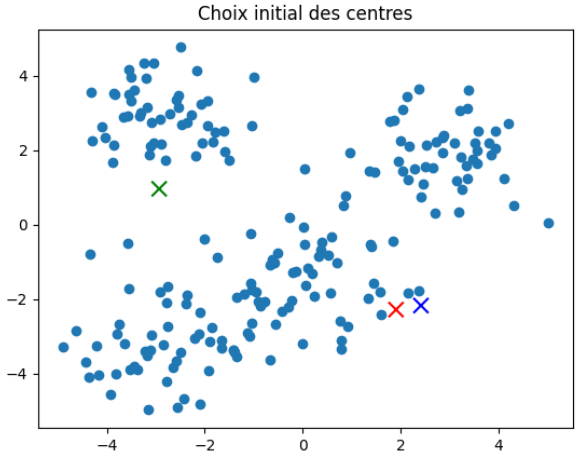
- Points aléatoires de l'espace (suivant une certaine distribution appropriée)
- Éléments aléatoires de $\mathcal{D}$ (*ditto*)

Nombre d'itérations

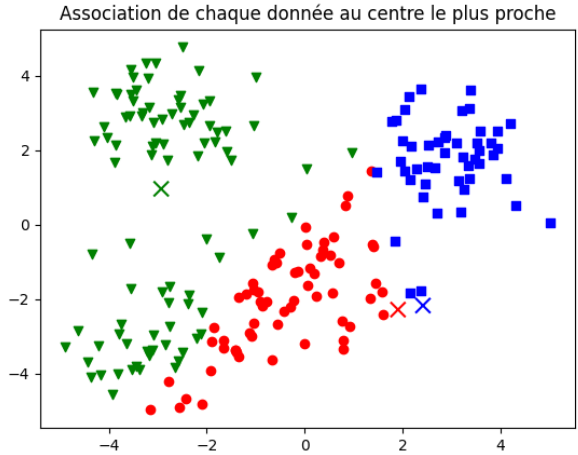
Deux questions :

- Quand s'arrêter dans *une* exécution
  - Quand les partitions / l'inertie n'évoluent plus (beaucoup)
- *Combien* d'exécutions
  - Le caractère aléatoire des initialisations peut faire tomber dans des *minimaux locaux*

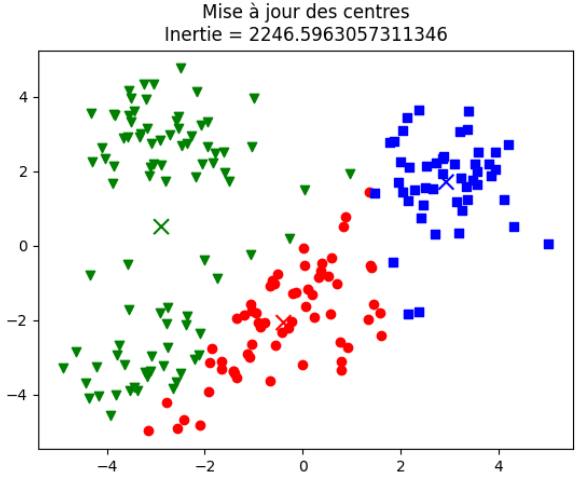
Exemple jouet : initialisation



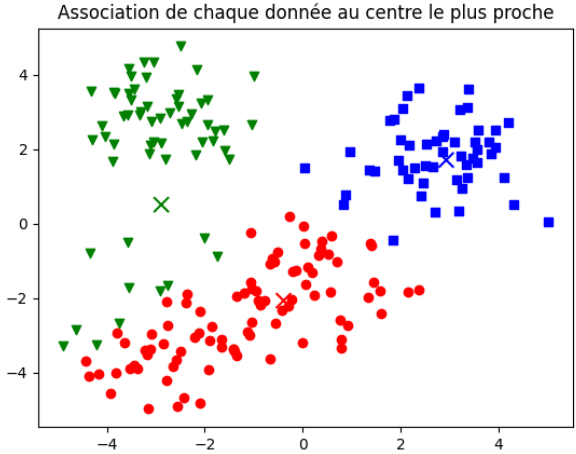
Exemple jouet



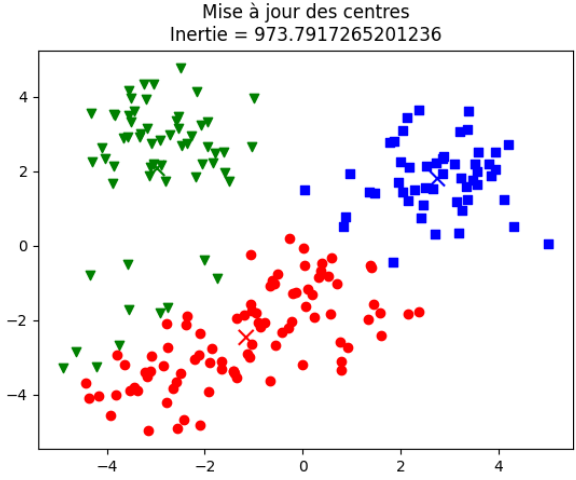
Exemple jouet



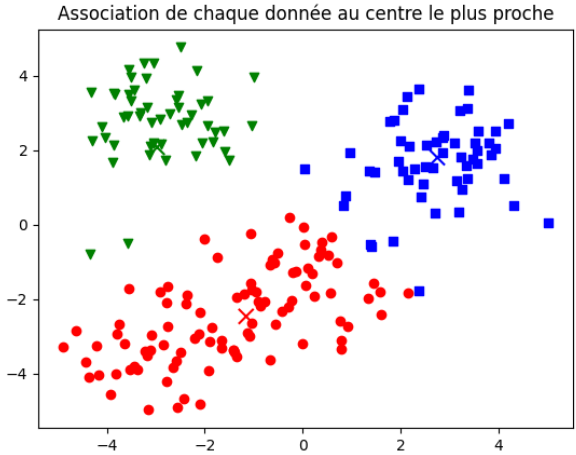
Exemple jouet



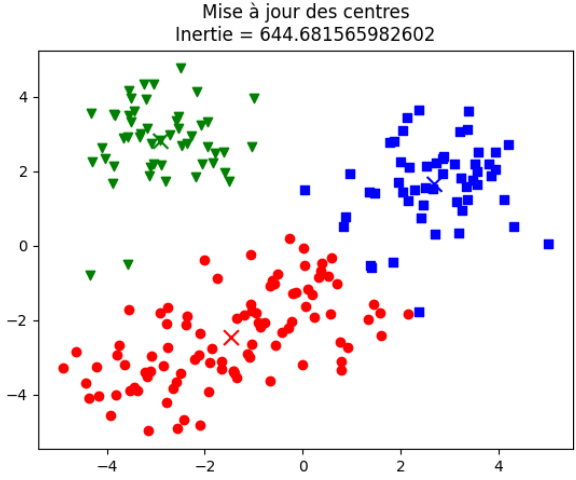
Exemple jouet



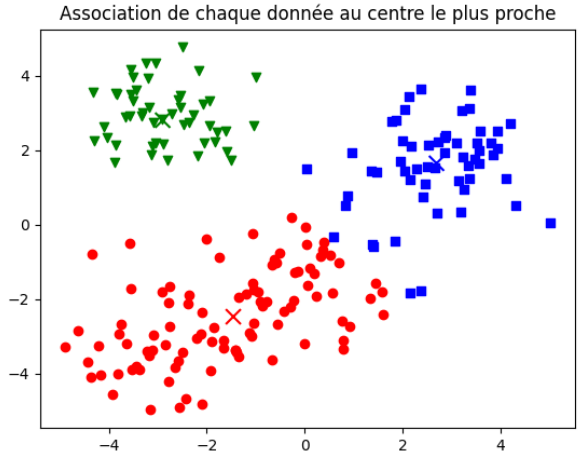
Exemple jouet



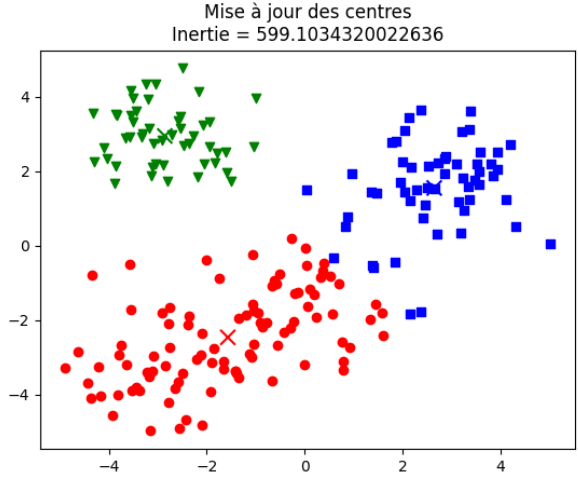
Exemple jouet



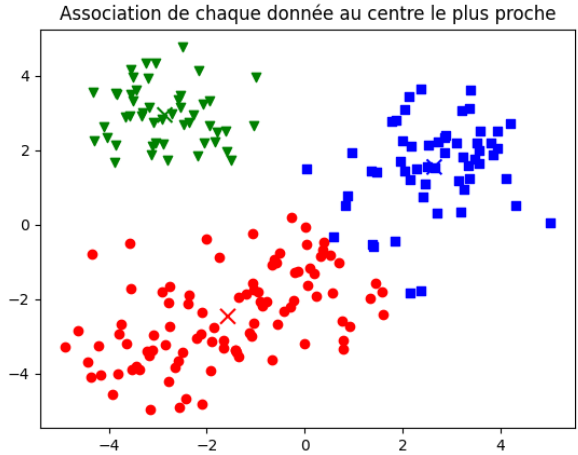
Exemple jouet



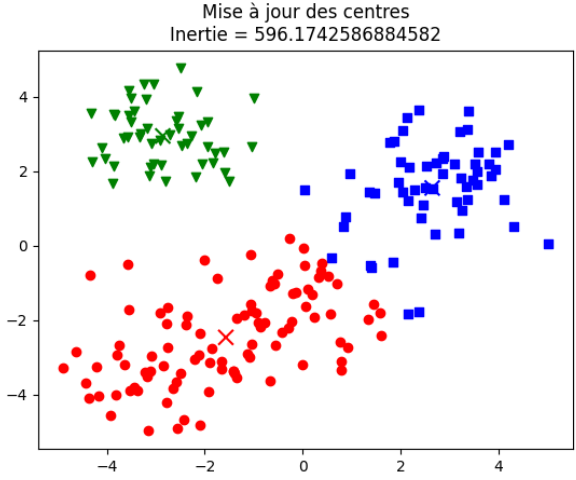
Exemple jouet



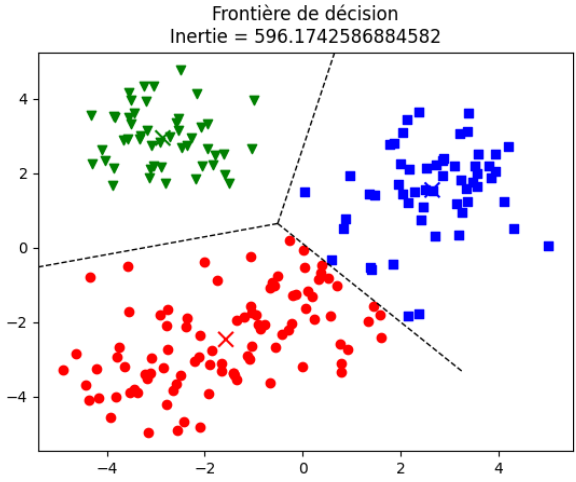
Exemple jouet



Exemple jouet



Exemple jouet : résultat



Terminaison 

Terminaison

Un algorithme des k-moyennes termine (si les calculs sont effectués avec une précision suffisante)

Esquisse d'argument :

- L'inertie (est positive et) ne peut prendre qu'un nombre fini de valeurs
- La mise à jour des centres fait baisser l'inertie d'une partition
- Si un point change de partition il fait baisser l'inertie globale

Donc l'inertie est un variant

Minimaux locaux

Mais encore une fois, cette terminaison peut se faire en un minimum local

En pratique

On n'attend pas forcément un point fixe pour s'arrêter (cf. ci-dessus)

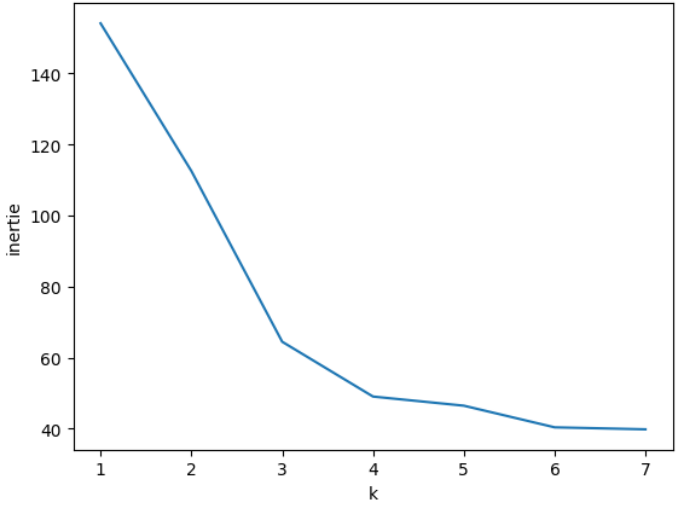
Heuristique pour le choix de k

Une approche : « méthode du coude »

- On exécute (éventuellement plusieurs fois) l'algorithme pour des valeurs de k croissantes
  - plus ou moins vite (linéairement, exponentiellement…) en fonction de la valeur finale de k conjecturée
- On choisit le plus grand k pour lequel l'inertie a décru de façon « significative » (par rapport à $k-1$)

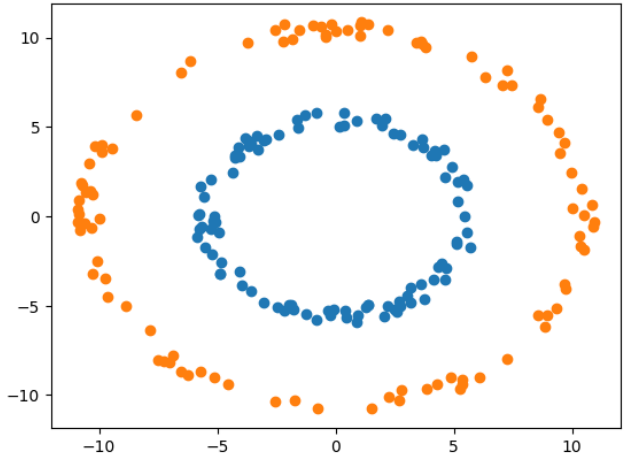
Intuition : le nombre k ainsi obtenu est nécessaire pour correctement catégoriser les données ; l'augmenter relève plus de la micro-optimisation

Exemple de coude



Limitations

L'approche k-moyenne ne fonctionne que sur des données *linéairement séparable* (c'est-à-dire pouvant être séparées par des hyperplans). Contre-exemple :



Exemple d'application : compression naïve d'images

Stockage BMP + Palette

- Chaque pixel *individuel* d'une image couleur (numérique classique) est stocké sur 24 bits
  - par ex. $256 = 2^8$ intensités pour rouge, vert, bleu
- Une image « bitmap » non compressée de dim. X et Y a une taille $24XY$ bits
- Toutes les couleurs ne sont pas forcément utilisées : on peut compresser une image utilisant une *palette* de C couleurs en taille : $24C + \lceil \log_2 C \rceil XY$
  - comment ?

Compression *avec perte*

- On peut encore plus compresser une image en réduisant le nombre de couleurs utilisées, et en substituant chaque couleur originale par la couleur « la plus proche »
  - réduit C ci-dessus
- La palette des couleurs utilisables peut être fixée *a priori*
- …ou déterminée en fonction de l'image

Palette adaptative pour une image

Principe

- On fixe une taille de palette C (par ex. 256)
- On utilise un algorithme d'apprentissage pour calculer 256 couleurs, dont les clusters associées (pour l'image) minimisent l'inertie
  - Par ex. l'algorithme des k-moyennes
- On remplace chaque couleur par la couleur la plus proche

Choix d'instanciation

Ceux de l'approche k-moyennes :

- L'initialisation de la palette peut se faire à partir de couleurs existantes, aléatoires, une palette prédéfinie… En pratique les k-moyennes convergent vite
- On peut se fixer *a priori* une erreur acceptable entre image originale & compressée (en complément de l'inertie de la partition), et augmenter la taille de palette au fur et à mesure

Illustration : image originale

Figure: Martre des pin avec sandwich (Photo : Vince Smith)

Illustration : 4 couleurs

Figure: Martre des pin avec sandwich en 4 couleurs

Illustration : 8 couleurs

Figure: Martre des pin avec sandwich en 8 couleurs

Illustration : 16 couleurs

Figure: Martre des pin avec sandwich en 16 couleurs

Illustration : 32 couleurs

Figure: Martre des pin avec sandwich en 32 couleurs

Illustration : 64 couleurs

Figure: Martre des pin avec sandwich en 64 couleurs

Illustration : 128 couleurs

Figure: Martre des pin avec sandwich en 128 couleurs

Illustration : 256 couleurs

Figure: Martre des pin avec sandwich en 256 couleurs

Illustration : image originale

Figure: Martre des pin avec sandwich (Photo : Vince Smith)