I Science des données

La science des données (*data science*) a pour objectif d'extraire de l'information à partir de données brutes.

Exemples :

- Données sur des fleurs : longueur et largeur des pétales et des sépales.
- Données sur des élèves : moyenne, classe...

Pour pouvoir avoir une notion de distance entre deux données, on représente chaque donnée comme un vecteur de $\mathbb{R}^p$. Les composantes de ce vecteur sont appelées les attributs.

Exemple : chaque donnée de fleur peut être représentée par un quadruplet de $\mathbb{R}^4$ correspondant à la longueur et largeur des pétales et des sépales.

Parfois il est moins évident de représenter une donnée par un vecteur :

- Variable catégorielle (non numérique : genre, couleur...) : on peut utiliser un vecteur avec un 1 et que des 0 (*one-hot vector*).

  Exemple : on peut représenter les classes MP2I/MPI/MPSI/MP par des vecteurs $\begin{pmatrix}1\\0\\0\\0\end{pmatrix}, \begin{pmatrix}0\\1\\0\\0\end{pmatrix}, \begin{pmatrix}0\\0\\1\\0\end{pmatrix}, \begin{pmatrix}0\\0\\0\\1\end{pmatrix}$.
- Image : On passe d'une matrice de pixels avec n lignes, p colonnes à un vecteur de taille np.

On représente classiquement l'ensemble des données (donc de vecteurs de $\mathbb{R}^p$) par une matrice X dont chaque ligne est une donnée et chaque colonne est un attribut.

Pour savoir si deux données sont « proches » l'une de l'autre, on utilise une distance sur les données, c'est-à-dire sur $\mathbb{R}^p$.

Exemples :

- Distance euclidienne :
$$d(x,y) = \sqrt{\sum_{i=1}^{p}(x_i - y_i)^2}$$
- Distance de Manhattan :
$$d(x,y) = \sum_{i=1}^{p}|x_i - y_i|$$

Quand les attributs n'ont pas la même échelle, un attribut peut avoir beaucoup plus d'importance qu'un autre dans les calculs de distance. Pour que les attributs aient la même importance, on peut standardiser les données, c'est-à-dire les modifier pour avoir une moyenne de 0 et un écart-type de 1, en utilisant :

Théorème

Si Z est une variable aléatoire d'écart-type $\sigma \neq 0$ alors $\dfrac{Z - \mathbb{E}(Z)}{\sigma}$ a une espérance nulle et un écart-type égal à 1.

Exercice 1.
Écrire une fonction `float standardiser(float** X, int n, int p)` qui standardise les données `X` de taille $n \times p$.

II Apprentissage supervisé

L'apprentissage supervisé consiste, à partir de données d'entraînements dont on connaît la classe, à prédire la classe d'une nouvelle donnée :

Définition : Apprentissage supervisé

Inconnu : $f : X \longrightarrow Y$, où X un ensemble de données et Y un ensemble d'étiquettes (ou classes).
Entrée : des données d'entraînement $x_1, ..., x_n \in X$ et leurs étiquettes $f(x_1), ..., f(x_n) \in Y$.
Sortie : une fonction $g : X \longrightarrow Y$ approximant f.

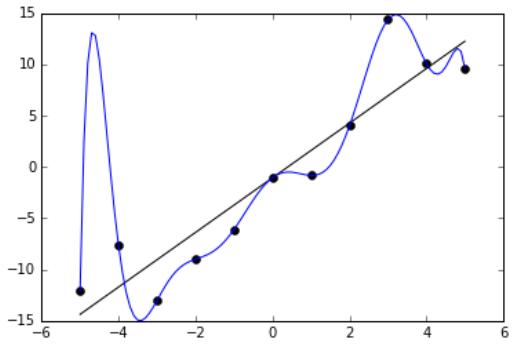
Suivant l'ensemble possible d'étiquettes, on parle de :

- Classification : Y est fini, par exemple $Y = \{1, ..., k\}$.
  Exemples : k plus proches voisins, arbre de décision, réseau de neurones...
- Régression : Y est un ensemble continu, par exemple $Y = \mathbb{R}$.
  Exemples : régression linéaire, modèle linéaire généralisé...

Exemples de problèmes de classification :

X	Y	$f(x)$
Tailles de tumeurs	Maligne, Bénigne	Gravité de x
Mails	Spam, Non-spam	Cee mail est-il un spam ?
Images	$[\![0, 9]\!]$	Chiffre représenté sur x ?
Musiques	classique, rap, rock...	Genre musical de x

On souhaite éviter le surapprentissage (*overfitting*) où l'algorithme approxime trop bien les données d'entraînement et généralise mal sur de nouvelles données.



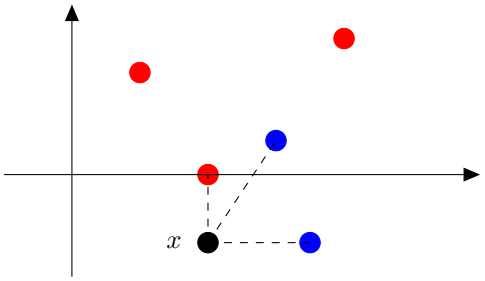
Le polynôme de Lagrange passe par tous les points d'entraînement mais généralise moins bien que la régression linéaire.

III Algorithme des k plus proches voisins (KNN)

Soit $k \in \mathbb{N}$. L'algorithme des k plus proches voisins prédit la classe d'une nouvelle donnée x de la façon suivante :

1. Trouver les k données d'entraînement les plus proches de x (en termes de distance).
2. Trouver la classe majoritaire $c \in Y$ parmi de ces k données.
3. Prédire que x est de classe c.

III.1 Trouver les k plus proches voisins



La classe de x est prédite comme étant bleue (ici, $k = 3$ et il y a deux classes : bleu et rouge)

Exercice 2.
Soit `x` un vecteur sous forme de tableau, `X` une matrice de données, `k` un entier et `d` une fonction de distance supposée définie.
Écrire une fonction `int* voisins(float* x, float** X, int k)` renvoyant un tableau des indices des k plus proches voisins de `x` dans `X`.

Autres solutions :

- Trier les données d'entraînement par ordre croissant de distance à x et prendre les k premières en $\mathrm{O}(np + n\log(n))$.
- Utiliser une file de priorité (tas min) en $\mathrm{O}(np + n\log(k))$.

III.2 Trouver la classe majoritaire parmi les k plus proches voisins

Exercice 3.
Écrire une fonction `int maj(int* T, int n, int k)` renvoyant en $\mathrm{O}(n)$ l'élément le plus fréquent d'un tableau `T` de taille `n` dont les éléments sont compris entre 0 et $k - 1$.

Exercice 4.
Écrire une fonction `int knn(float* x, float** X, int* Y, int k, int n, int p)` qui prédit la classe de `x` en utilisant l'algorithme KNN, où :

- `x` est la donnée à prédire,
- `X` est la matrice des données d'entraînement,
- `Y[i]` est la classe de la donnée `X[i]`,
- `k` est le nombre de voisins à considérer,
- `n` est le nombre de données d'entraînement,
- `p` est le nombre d'attributs.

IV Évaluation d'un algorithme d'apprentissage

IV.1 Données de test

Supposons posséder des données X avec des étiquettes Y et qu'on veuille savoir si KNN est un bon classifieur. Pour cela, on peut réserver une partie des données pour tester l'algorithme obtenu, en partitionnant X et Y en deux ensembles :

- Ensemble d'entraînement X_{train} (de classes Y_{train}) : données parmi lesquelles on va chercher les k plus proches voisins.
- Ensemble de test X_{test} (de classes Y_{test}) : données utilisées pour évaluer l'algorithme, en comparant les classes prédites par KNN avec les classes réelles.

IV.2 Mesures d'évaluation

Définition : Mesures d'évaluation

- La précision d'un algorithme d'apprentissage est la proportion de données de test bien classées par rapport au nombre total de données.
- L'erreur est égale à $1 -$ précision.
- La matrice de confusion est une matrice carrée dont les lignes et les colonnes sont les classes possibles. La case (i, j) contient le nombre de données de test de classe i qui ont été prédites comme appartenant à la classe j.

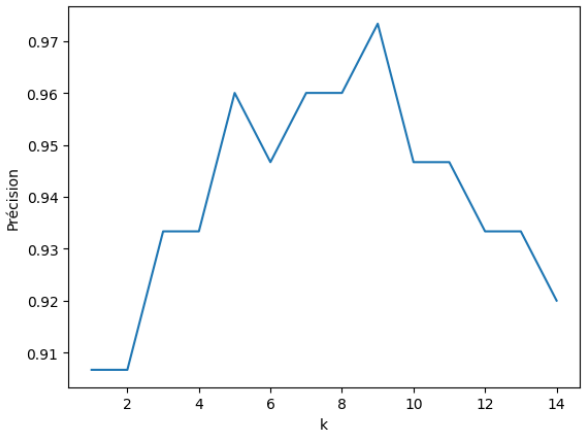
Exemple : Dans la matrice suivante, on voit que toutes les données de classe 0 ont été prédites correctement, une donnée de classe 1 a été prédite à tord comme appartenant à la classe 2.

$$\begin{pmatrix} 21 & 0 & 0 \\ 0 & 29 & 1 \\ 0 & 2 & 23 \end{pmatrix}$$

La précision est la somme des éléments diagonaux divisée par la somme de tous les éléments.

IV.3 Choix du nombre de voisins k

Pour choisir la meilleure valeur de k, on peut utiliser la méthode du coude : afficher la précision en fonction de k pour choisir la valeur de k qui maximise la précision.

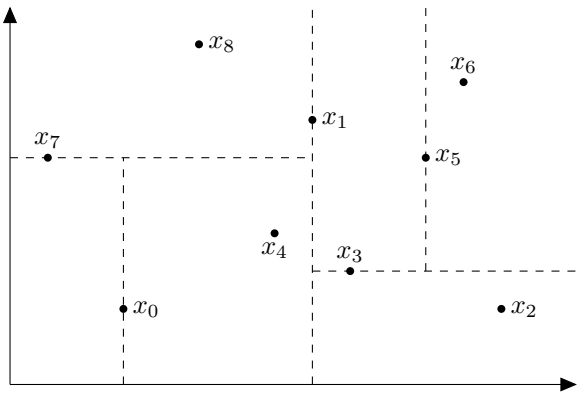


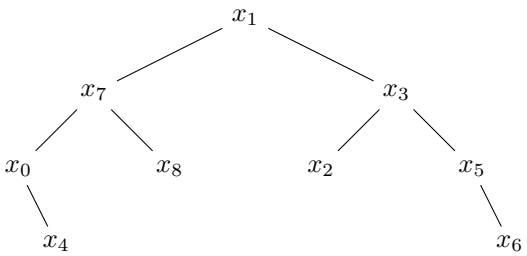
Choisir $k = 9$ permet de maximiser la précision

V Arbre $k - d$

Un arbre $k - d$ est une structure de données permettant de calculer plus rapidement les plus proches voisins.

Construction de l'arbre : à la profondeur i, on divise les points de $\mathbb{R}^p$ en deux parties égales (± 1) en prenant comme hyperplan de division l'axe i modulo p (x, y, z...).





Pour trouver le point le plus proche de $y \in \mathbb{R}^p$:

1. On trouve la feuille de l'arbre correspondant à la zone contenant y.
2. On remonte l'arbre en conservant la distance minimum trouvée et en explorant l'autre sous-arbre si nécessaire.

On peut adapter cette méthode pour trouver les k plus proches voisins parmi n points efficacement et accélérer l'algorithme des k plus proches voisins.