Chapitre 3

Apprentissage

Hugo Salou MPI*

Dernière mise à jour le 31 octobre 2022

0 Motivation

L'intelligence artificielle est vu comme un « objet magique » mais ce n'est pas le cas : c'est ce que nous allons étudier dans ce chapitre. Il existe plusieurs méthodes permettant l'apprentissage : descente de gradient, ?, ...

La base de donnée la plus utilisée est MNIST : elle contient 60 000 images de 28×28 pixels représentant un chiffre, et le chiffre correspondant. L'idée de l'apprentissage est de « deviner » le chiffre dessiné en connaissant l'image.

1 Vocabulaire

Définition: On appelle *signature de données* un n-uplet de paires nom, ensemble ; on le typographie

$$(\text{nom}_1 : S_1, \text{nom}_2 : S_2, \dots, \text{nom}_n : S_n).$$

Définition: Étant donné une signature de données $\mathbb{S} = (\text{nom}_1 : S_1, \dots, \text{nom}_n : S_n)$, on appelle *donnée* un vecteur

$$\bar{v} = (v_1, v_2, \dots, v_n) \in S_1 \times S_2 \times \cdots \times S_n.$$

Définition: Étant donné une signature de données $\mathbb{S}$, on appelle *jeu de données* un ensemble fini de vecteurs de signature $\mathbb{S}$.

Définition: Étant donnée une signature de données $\mathbb{S}$ et un ensemble de classes $\mathscr{C}$, on appelle *jeu de données classifié* la donnée
- d'un jeu de données S,
- d'une fonction $f : S \to \mathscr{C}$ de classification.

2 Apprentissage supervisé

L'objectif de cette section est de construire des fonctions de classification, à partir d'un jeu de données classifié.

Définition: Étant donné une signature de données $\mathbb{S}$, et un ensemble de classes $\mathscr{C}$, on appelle *fonction de classification* une fonction des données de de signature $\mathbb{S}$ dans $\mathscr{C}$.

REMARQUE:
On discutera de la « qualité » d'une fonction de classification en fonction de ses résultats sur les données d'un jeu de données et sur des exemples de tests.

2.1 k plus proches voisins

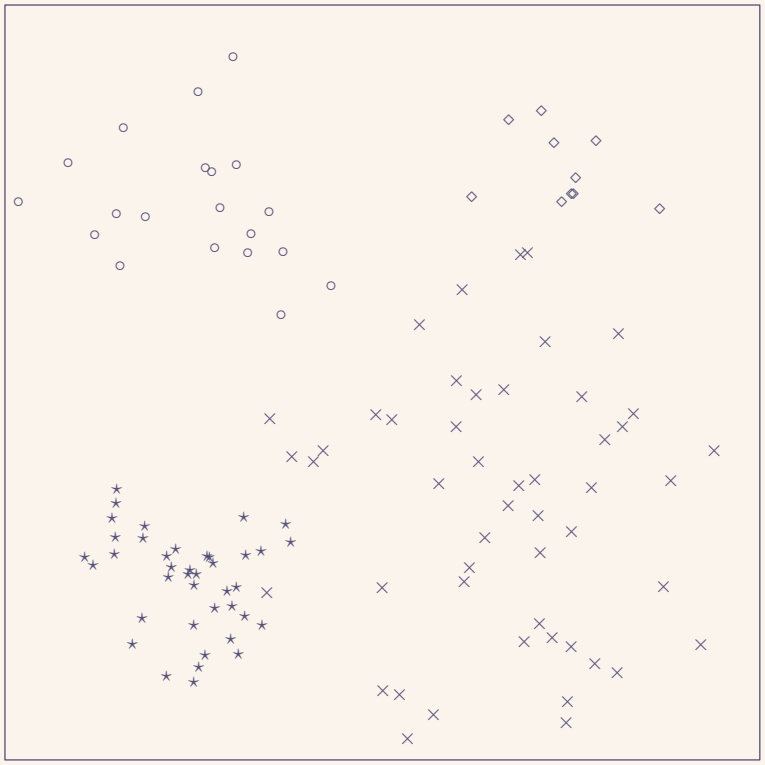

FIGURE 1 – Représentation de l'algorithme des k plus proches voisins

Algorithme 1 k-NN (k *nearest neighbors*)

Entrée Un jeu de données classifié (S, c), un vecteur d'entrée $\bar{v}$

1: On trie S par distance à croissante de v en $d_1, d_2, \ldots, d_k, d_{k+1}, \ldots$
2: Soit D un dictionnaire de $\mathscr{C}$ vers $\mathbb{N}$ initialisé à 0 [1]
3: **pour** $j \in [\![1, k]\!]$ **faire**
4: $\quad D\big[c(d_j)\big] \leftarrow D\big[c(d_j)\big] + 1$
5: **retourner** $\operatorname{argmax}_{d \in \mathscr{C}} D[d]$

REMARQUE:
On doit avoir $k \leqslant n$, et l'espace doit être muni d'une distance. Les résultats de l'algorithme dépendent fortement du jeu de données, du paramètre k et de la distance choisie.

Matrice de confusion

Définition: On appelle *matrice de confusion* d'un algorithme de prédiction $\mathscr{A}$ sur un jeu de données classifié (T, c), la matrice

$$\Big(\operatorname{Card}\{t \in T \mid \mathscr{A}(t) = i \text{ et } c(t) = j\}\Big)_{(i,j) \in \mathscr{C}^2}.$$

Dans le cas particulier dans le cas d'une classification $(\boldsymbol{V}, \boldsymbol{F})$, on nomme

1. où toutes les valeurs sont initialisées à 0, pas un dictionnaire vide

$\mathcal{A}$ \ vrai	$\boldsymbol{F}$	$\boldsymbol{V}$
$\boldsymbol{F}$	vrai négatif	faux négatif
$\boldsymbol{V}$	faux positif	vrai positif

TABLE 1 – Matrice de confusion dans le cas d'une classification en $\boldsymbol{V}$ et $\boldsymbol{F}$

Comment améliorer la performance de l'algorithme des k plus proches voisins ? En dimension 1, on peut utiliser une dichotomie. Mais, dans des dimensions plus grandes, l'ordre lexicographique, et l'ordre produit ne fonctionnent pas. Mais, on peut appliquer une "dichotomie" en changeant de dimension. Par exemple, en deux dimension, on a

À faire : Représenter le schéma

FIGURE 2 – Représentation de la "dichotomie" en dimension 2

Pour représenter cette structure de données, on utilise un arbre binaire comme montré ci-dessous. Cet arbre est appelé un arbre k-dimensionnels.

À faire : Faire l'arbre

FIGURE 3 – Arbre 2-dimensionnel représentant la "dichotomie" précédente

2.2 Arbres k-dimensionnels

REMARQUE (Notations):
Étant donné un jeu de données S, on note pour $v \in S$,

$$S^{\leqslant_i v} = \{u \in S \mid u_i \leqslant v_i\} \qquad \text{et} \qquad S^{>_i v} = \{u \in S \mid u_i > v_i\}.$$

Algorithme 2 "F" : Fabrication d'un arbre k-dimensionnel

Entrée $\mathscr{V}$ un jeu de données et $i \in [\![0, n-1]\!]$, où n est la dimension des données
1: **si** $\mathscr{V} = \varnothing$ **alors**
2: | **retourner** Vide
3: **sinon**
4: | On cherche $v \in \mathscr{V}$ tel que v_i est la médiane de $\{u_i \mid u \in \mathscr{V}\}$
5: | **retourner** Nœud$\Big((v,i), \mathrm{F}\big((\mathscr{V} \setminus \{v\})^{\leqslant_i v}, i+1 \bmod n\big), \mathrm{F}\big((\mathscr{V} \setminus \{v\})^{>_i v}, i+1 \bmod n\big)\Big)$

Algorithme 3 “R” : Recherche du point le plus proche

Entrée Un arbre k-dimensionnel et un vecteur v

si T est vide **alors**
  retourner $\varnothing$
sinon
  Nœud$\big((u, i), G, D\big) \leftarrow T$
  si $u_i \leqslant v_i$ **alors**
    $W \leftarrow \mathrm{R}(D, v)$
    si $W = \text{None}$ **alors**
      $W' \leftarrow \mathrm{R}(G, v)$
      si $W' = \varnothing$ **alors**
        retourner Some(u)
      sinon
        Some$(z) \leftarrow W'$
        retourner le plus proche de v entre u et z
    sinon
      Some$(w) \leftarrow W$
      si $v_i - u_i \leqslant d(w, v)$ **alors** ▷ $d(w, v)$ *représente la distance entre* w *et* v
        $W' \leftarrow \mathrm{R}(G, v)$
        si $W' = \text{None}$ **alors**
          retourner Some(plus proche de v entre u et w)
        sinon
          Some$(z) \leftarrow w$
          retourner Some(plus proche de v entre $u,\ v,$ et w
      sinon
        retourner W