Chapitre 3

Apprentissage

Hugo Salou MPI*

Dernière mise à jour le 31 mars 2023

Table des matières

0 Motivation

L'intelligence artificielle est vue 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, k-plus proches voisins, ...

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 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

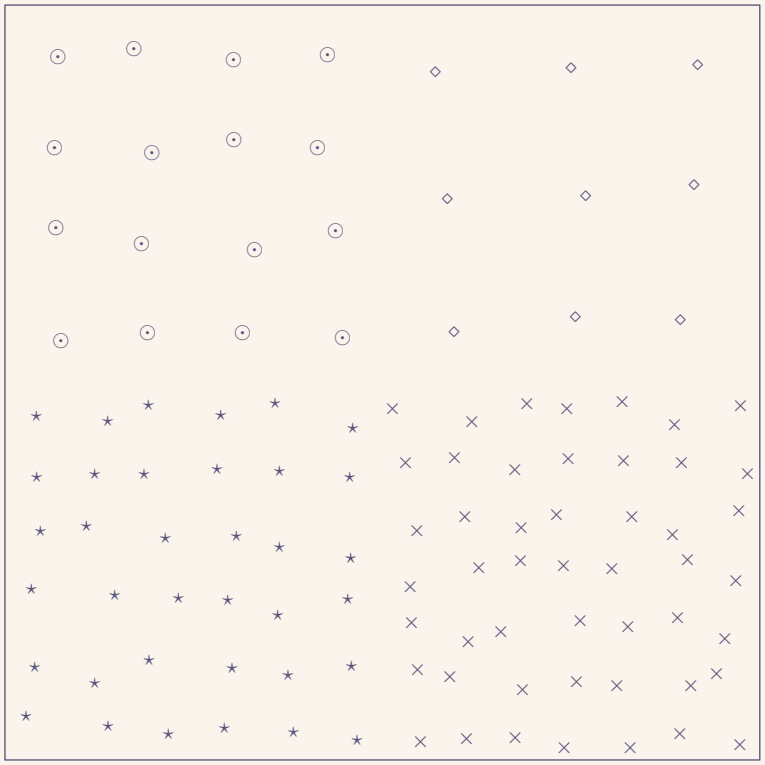

FIGURE 1 – Représentation de l'algorithme des k plus proches voisins

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 croissante de v en d_1, d_2, …, d_k, d_{k+1}, …
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** $\arg\max_{d \in \mathscr{C}} D[d]$

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

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

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

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

$\mathcal{A}$ \ 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 obtient la dichotomie ci-dessous.

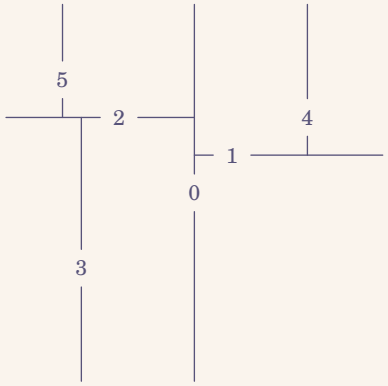

FIGURE 2 – Représentation de la "dichotomie" en dimension 2

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

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

si $\mathscr{V} = \varnothing$ **alors**
  retourner Vide
sinon
  On cherche $v \in \mathscr{V}$ tel que v_i est la médiane de $\{u_i \mid u \in \mathscr{V}\}$
  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 None
sinon
  Nœud$\big((u,i), G, D\big) \leftarrow T$
  si $u_i \leqslant v_i$ **alors**
    $W \leftarrow \mathrm{R}(D, v)$
    si $W =$ 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 $W' =$ 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, z, et w)
      sinon
        retourner W
  sinon
    ⟨comme le cas précédent⟩

2.3 Algorithme ID3

L'algorithme des k plus proches voisins (sans arbres k-dimensionnels) n'a pas de *phase d'apprentissage* : en effet, les données ne sont pas réorganisées. Mais, par exemple, pour l'utilisation des arbres k-dimensionnels, les données sont réorganisées dans un arbre.

Ce qu'on aimerai avoir, c'est des *bordures* entre les différentes classes. Par exemple, dans l'exemple précédent, on aimerai avoir les différentes zones ci-dessous.

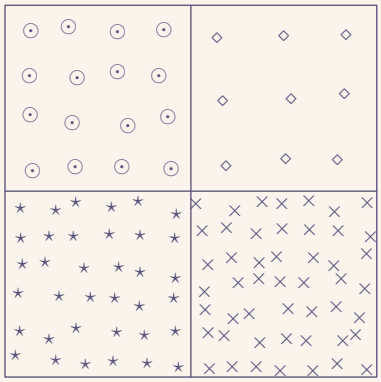

FIGURE 4 – Représentation de *bordures* entre les différentes classes

De ces zones, on peut construire un algorithme qui classifie les données, que l'on représente sous forme d'arbre. Ce type d'arbre est un *arbre de décision*. Dans l'exemple précédent, on peut donc créer l'arbre ci-dessous. [2]

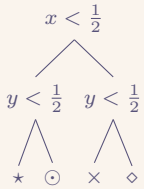

FIGURE 5 – Arbre de décision pour la classification

Dans le reste de cette section, on s'intéresse uniquement à des données de $\mathbb{B}^n$ (une liste de n booléens) pour un certain $n \in \mathbb{N}$.

On considère l'exemple dont les données ci-dessous.

Transport	Moteur	Rails	Sous-terre	$\geqslant 320$ km/h	Train ?
A380	✓	✗	✗	✓	✗
TGV	✓	✓	✗	✓	✓
Métro	✓	✓	✓	✗	✓
Wagonnet	✗	✓	✓	✗	✗
Draisine	✗	✓	✗	✗	✗
Tram	✓	✓	✗	✗	✓

TABLE 2 – Exemple de données

Entropie

Définition : Étant donné un variable aléatoire finie X à valeurs dans E. On note $p_X : E \to [0, 1]$ sa loi de probabilité :

$$\forall x \in E, \quad p_X(x) = P(X = x).$$

2. Dans cet arbre, si un nœud représente une condition, son fils gauche représente si cette condition est vraie, et le fils droit sinon.

On définit l'*entropie* $\mathrm{H}(X)$ de cette variable aléatoire comme

$$\mathrm{H}(X) = -\sum_{x\in E} p_X(x)\ \ln\big(p_X(x)\big).$$

On prolonge $p_X(x)\ln(p_X(x))$ par continuité à la valeur 0 lorsque $p_X(x) = 0$.

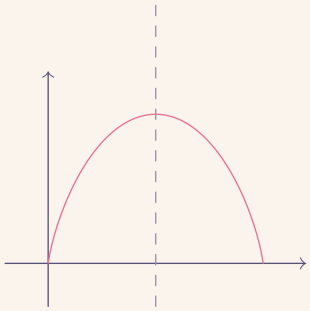

FIGURE 6 – Représentation graphique de $\mathrm{H}(X)$ en fonction de p

Lemme : Si $(p_i)_{i\in[\![1,n]\!]} \in\ ]0,1]^n$ sont tels que $\sum_{i=1}^n p_i = 1$. Soit $(q_i)_{i\in[\![1,n]\!]}$ tels que $\forall i,\ q_i = \frac{1}{n}$. On a alors

$$-\sum_{i=1}^{n} p_i \ln p_i \leqslant -\sum_{i=1}^{n} p_i \ln(q_i).$$

■

Propriété : Soit $n = |E|$. L'entropie d'une variable aléatoire à valeurs dans E est maximale lorsque

$$\forall e \in E,\ P(X = e) = \frac{1}{n}.$$

■

Définition : Étant donné un jeu de données classifiés (S, c) où $c : \mathbb{S} \to E$, on appelle *entropie* de ce jeu de données l'entropie de la variable aléatoire $c(Y)$ où $Y \sim \mathcal{U}(S)$. On a donc

$$\mathrm{H}\big((S,c)\big) = -\sum_{e\in E} \frac{\mathrm{Card}\ c^{-1}(\{e\})}{\mathrm{Card}\, S} \ln\left(\frac{\mathrm{Card}\ c^{-1}(\{e\})}{\mathrm{Card}\, S}\right).$$

Définition : Étant donné un jeu de données une partition $\{S_1, S_2, \ldots, S_p\}$ d'un jeu de données classifié (S, c), l'*entropie* de cette partition est la moyenne (pondérée par les

cardinaux et renormalisée) :

$$\mathrm{H}\big((\{S_1,\dots,S_p\},c)\big)=\sum_{i=1}^{p}\frac{\operatorname{Card}S_i}{\operatorname{Card}S}\,\mathrm{H}\big((S_i,c)\big).$$

L'entropie de $\{w,a,t,d,r,m\}$ est $\mathrm{H}=-\frac{3}{6}\ln\left(\frac{3}{6}\right)-\frac{3}{6}\ln\left(\frac{3}{6}\right)=\ln 2\simeq 0{,}69$. Mais, avec le découpage de l'arbre de décision ci-dessous, on obtient l'entropie

$$\begin{aligned}\mathrm{H}&=\frac{2}{6}\mathrm{H}(\{w;d\})+\frac{4}{6}\mathrm{H}(\{a,t,r,m\})\\&=\frac{2}{6}\times 0+\frac{4}{6}\times\left(-\frac{1}{4}\ln\left(\frac{1}{4}\right)-\frac{3}{4}\ln\left(\frac{3}{4}\right)\right)\\&\simeq 0{,}37.\end{aligned}$$

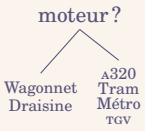

Figure 7 – Arbre de décision possible se basant sur le moteur

Avec un autre arbre (comme celui ci-dessous), on obtient une entropie différente :

$$\begin{aligned}\mathrm{H}&=\frac{1}{6}\mathrm{H}(\{a\})+\frac{5}{6}\mathrm{H}(\{w,d,t,r,m\})\\&=\frac{5}{6}\left(-\frac{2}{5}\ln\left(\frac{2}{5}\right)-\frac{3}{5}\ln\left(\frac{3}{5}\right)\right)\\&\simeq 0{,}56.\end{aligned}$$

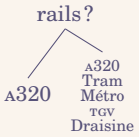

Figure 8 – Arbre de décision possible se basant sur les rails

Pour le sous-terrain, on a $\mathrm{H}=\ln 2$.

Figure 9 – Arbre de décision possible se basant sur sous-terrain

À faire : Vérifier

À faire : Autre cas

Ainsi, on choisit de commencer avec la condition "moteur" car l'entropie est la plus faible avec cette condition. Ainsi, l'arbre de décision ressemble à celui ci-dessous.

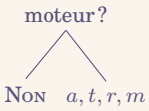

Figure 10 – Arbre de décision partiel

On réitère avec les autres conditions. L'entropie en se basant sur la vitesse est $\frac{1}{2}\ln 2 \simeq 0{,}34$. En effet, l'arbre de décision possible ressemble à celui ci-dessous.

Figure 11 – Arbre de décision possible se basant sur le moteur puis la vitesse

Mais, en se basant sur sous-terrain, on obtient une entropie de $\frac{3}{4}\left(-\frac{2}{3}\ln\left(\frac{2}{3}\right) - \frac{1}{3}\ln\left(\frac{1}{3}\right)\right) \simeq 0{,}48$.

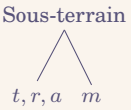

Figure 12 – Arbre de décision possible se basant sur le moteur puis sous-terrain

Et, en se basant sur les rails, on obtient une entropie de 0.

Figure 13 – Arbre de décision possible se basant sur le moteur puis les rails

On en déduit que l'arbre final de décision est celui ci-dessous.

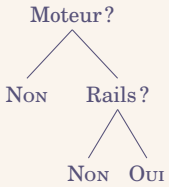

Figure 14 – Arbre de décision final pour la classification de trains

Les données que l'on a utilisées sont pour l'apprentissage. On teste notre arbre de décision sur les données ci-dessous.

Nom	Moteur	Rail	Sous-terre	Vitesse	Résultat de l'algorithme
Bus	✓	✗	✗	✗	✗
TER	✓	✓	✗	✗	✓
Cheval	✗	✗	✗	✗	✗
Ascenseur spatial	✓	✓	✗	✗	✓

TABLE 3 – Test de l'arbre de décision créé

ATTENTION : il ne faut pas faire du *sur-apprentissage*, comme montré sur la figure ci-dessus, où la modélisation correspond trop aux données utilisées pour la classification.

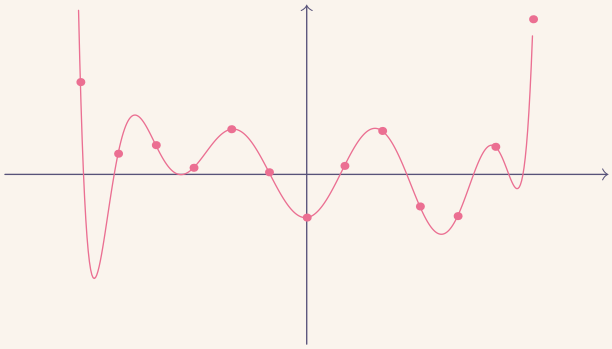

FIGURE 15 – Représentation graphique du problème de *sur-apprentissage*

Aussi, il faut faire attention aux critères : par exemple, lors de la classification de photos de chats et de chiens, les photos de chiens sont en général prises en extérieur et l'algorithme ID3 aurait donc pu choisir de baser sa décision sur l'emplacement de la photo, même si elle n'importe pas dans la différenciation chat/chiens.

Autre exemple : on considère la table de données ci-dessous. Utilisons l'algorithme ID3 sur ces données, et trouvons l'arbre de décision.

A	B	C	D	Classification
✓	✗	✗	✓	●
✓	✓	✗	✓	●
✓	✗	✓	✓	●
✓	✓	✓	✓	●
✗	✗	✓	✓	●
✗	✓	✓	✗	●
✗	✗	✗	✗	●
✗	✓	✗	✓	●

TABLE 4 – Table de données d'exemple

Pour les conditions A, B et C, on a $\mathrm{H} \simeq 0{,}63$ et, pour D, on a $\mathrm{H} = 0{,}65$. Comme on prend le $1^{\underline{\text{er}}}$ dans l'ordre lexicographique, on choisit la condition A. De même, on construit l'arbre ci-dessous.

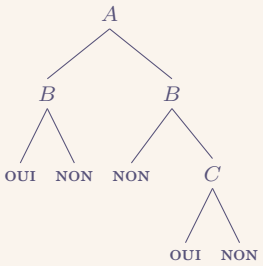

FIGURE 16 – Arbre de décision pour la table de données précédente

Le nom de ID3 vient de *iterative dichotomizer*.

3 Apprentissage non supervisé

On rappelle que l'*apprentissage non supervisé* utilise des données non classifiées. L'objectif est de les classifier. Plus rigoureusement, étant donné un jeu de données $\mathcal{D}$, on veut partitionner $\mathcal{D}$ en $\{S_1, S_2, \dots, S_n\}$ où les S_i sont regroupés par « similitude. »

> REMARQUE :
> Dans le reste de cette section, $\mathcal{D} \subseteq \mathbb{R}^p$, avec $p \in \mathbb{N}^\star$, et la notion de similitude est donnée par la distance euclidienne.

3.1 Algorithme HAC, classification hiérarchique ascendant

Définition : Étant donné deux sous-ensembles A et B de $\mathcal{D}$ disjoints et non-vides, on définit différentes *mesures de dissimilarité* :

- $D(A,B) = \min\{d(a,b) \mid a \in A, b \in B\}$;
- $D(A,B) = \max\{d(a,b) \mid a \in A, b \in B\}$;
- $D(A,B) = \frac{1}{|A|\,|B|} \times \sum_{(a,b)\in A\times B} d(a,b)$,

où $d(a,b)$ représente la distance entre a et b.

Algorithme 4 Algorithme HAC

```
P ← {{x} | x ∈ D}
tant que |P| ⩾ 2 et ?[3] faire
    Soit A, B ∈ P minimisant D(A,B) avec A ≠ B
    P ← (P \ {A,B}) ∪ {A ∪ B}
retourner P
```

3.2 k-moyenne

Au préalable, on fixe le nombre de classes k que l'on souhaite obtenir après exécution de l'algorithme.

3. où, pour ?, on peut choisir (1) un critère sur le nombre de classes (2) un critère sur la valeur de la plus petite dissimilarité

Propriété : On considère la fonction

$$f : \mathbb{R}^p \longrightarrow \mathbb{R}$$
$$H \longmapsto \sum_{v \in \mathcal{D}} \|H - v\|_2^2.$$

Cette fonction atteint son minimum en le barycentre G.

■

Dans la suite de cette sous-section, on s'intéresse à des partitionnement particuliers : étant donné une famille $(m_i)_{i \in [\![1,k]\!]}$ de vecteurs de $\mathbb{R}^p$, et un entier $j \in [\![1,k]\!]$, on définit

$$\mathcal{C}_j^m = \Big\{ v \in \mathcal{D} \;\Big|\; \operatorname*{arg\,min}_{i \in [\![1,k]\!]} \|v - m_i\|_2 = j \Big\}.$$

Remarque :
En cas d'égalité ($\|v - m_{i_1}\|_2 = \|v - m_{i_2}\|_2$), le "arg min" retourne le plus petit indice : $\min(i_1, i_2)$.

Étant donné une famille $(m_i)_{i \in [\![1,k]\!]}$ de vecteurs de $\mathbb{R}^p$, on a donc un partitionnement $\{\mathcal{C}_j^m \mid j \in [\![1,k]\!]\}$.

Définition : Étant donné une famille $(m_i)_{i \in [\![1,k]\!]}$ de vecteurs de $\mathbb{R}^p$, on définit

$$L(m) = \sum_{k=1}^{n} \Big(\sum_{v \in \mathcal{C}_i^m} \|v - m_i\|_2^2 \Big).$$

Pour minimiser localement la fonction L, on aimerait que les $(m_i)_{i \in [\![1,k]\!]}$ soient les barycentres des $(\mathcal{C}_i^m)_{i \in [\![1,k]\!]}$. Mais, la définition des $(\mathcal{C}_i^m)$ dépend de la position des (m_i). Ainsi, en itérant ce procédé, en modifiant les (m_i) pour être les barycentres des $(\mathcal{C}_i^m)$ respectifs, puis en modifiant les $(\mathcal{C}_i^m)$, la famille $(m_i)_{i \in [\![1,k]\!]}$ converge vers les barycentres des classes.

Algorithme 5 k-moyenne

Entrée $\mathcal{D}$ et k
1 : $(m_i)_{i \in [\![1,k]\!]} \leftarrow k$ vecteurs de $\mathcal{D}$
2 : stable $\leftarrow \boldsymbol{F}$
3 : **tant que** $\neg$stable **faire**
4 : $\quad C \leftarrow (\mathcal{C}_i^m)_{i \in [\![1,k]\!]}$
5 : $\quad m' \leftarrow \big(\text{barycentre}(\mathcal{C}_i)\big)_{i \in [\![1,k]\!]}$
6 : $\quad$ stable $\leftarrow m \overset{?}{=} m'$
7 : $\quad m \leftarrow m'$
8 : **retourner** C