Chapitre VIII
Arbres de décision et méthodes ensemblistes

Cycle pluridisciplinaire d'études supérieures
Université Paris sciences et lettres

On présente dans ce chapitre les arbres de décision qui sont un type de prédicteurs, et on évoquera les algorithmes d'apprentissage connus sans toutefois s'y attarder. Il se trouve que les prédicteurs donnés par ces algorithmes sont de qualité relativement moyenne. Cependant, l'utilisation par-dessus le marché de méthodes ensemblistes, présentés dans un second temps, permet d'obtenir des résultats bien meilleurs.

Les méthodes ensemblistes utilisent un algorithme d'apprentissage donné pour construire un grand nombre de prédicteurs différents, par exemple en faisant varier la valeur des hyperparamètres, l'échantillon d'apprentissage, ou encore en introduisant des choix aléatoires, et agrègent ces prédicteurs pour obtenir un prédicteur final, meilleur. On présentera dans ce chapitre deux méthodes ensemblistes, le Bagging et AdaBoost.

I. arbres de décision

On introduit de façon formelle les arbres de décision. Les algorithmes d'apprentissage donnant des arbres de décision ne seront en revanche qu'évoqués.

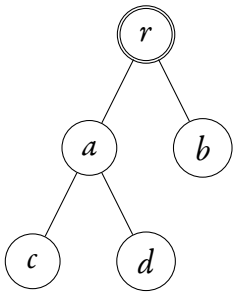
Définition. — *Arbre.* — Un *arbre enraciné* est la donnée de :
- un ensemble fini de *nœuds* V ;
- une *racine* (*root*) $r \in \mathrm{V}$;

— une application *parent* $\pi : \mathrm{V} \setminus \{r\} \to \mathrm{V}$ telle que pour tout $v \in \mathrm{V} \setminus \{r\}$, il existe $n \geqslant 1$ tel que $\pi^n(v) = r$ (n est alors appelé profondeur de v).

De plus,

— la *profondeur de l'arbre* (*depth*) est la profondeur maximale des nœuds ;
— les éléments de $\mathrm{V} \setminus \{\pi(\mathrm{V})\}$ sont appelées *feuilles* (*leafs*).

Exemple. — On se donne un ensemble de nœuds $\mathrm{V} = \{r, a, b, c, d\}$ où r est la racine, et avec une application parent π définie par : $\pi(c) = a$, $\pi(d) = a$, $\pi(a) = r$, $\pi(b) = r$. On peut représenter l'arbre par un graphe où on met une emphase sur la racine.



L'ensemble des feuilles est alors $\mathrm{V} \setminus \pi(\mathrm{V}) = \{b, c, d\}$, et la profondeur de l'arbre est égale à 2.

Définition. — *Arbre de décision.* — On se place dans un cadre d'apprentissage avec respectivement $\mathcal{X}$ et $\mathcal{Y}$ les ensembles d'entrées et de sorties. Soit :

— (V, r, π) un arbre enraciné ;
— $(y^{(v)})_{v \in \mathrm{V} \setminus \pi(\mathrm{V})} \in \mathcal{Y}^{\mathrm{V} \setminus \pi(\mathrm{V})}$ (autrement dit une sortie associée à chaque feuille de l'arbre) ;
— $(\mathcal{X}^{(v)})_{v \in \mathrm{V}}$ une famille de sous-ensembles de $\mathcal{X}$ telle que $\mathcal{X}^{(r)} = \mathcal{X}$ et telle que pour tout $v \in \pi(\mathrm{V})$, $(\mathcal{X}^{(v')})_{v' \in \pi^{-1}(v)}$ forme une partition de $\mathcal{X}^{(v)}$.

L'*arbre de décision* associé est le prédicteur $f : \mathcal{X} \to \mathcal{Y}$ défini par :

$$\forall x \in \mathcal{X}, \quad f(x) = \sum_{v \in \mathrm{V} \setminus \pi(\mathrm{V})} \mathbb{1}_{\mathcal{X}^{(v)}}(x) \cdot y^{(v)},$$

où $\mathbb{1}_{\mathcal{X}^{(v)}}(x) = 1$ si $x \in \mathcal{X}^{(v)}$ et $\mathbb{1}_{\mathcal{X}^{(v)}}(x) = 0$ sinon.

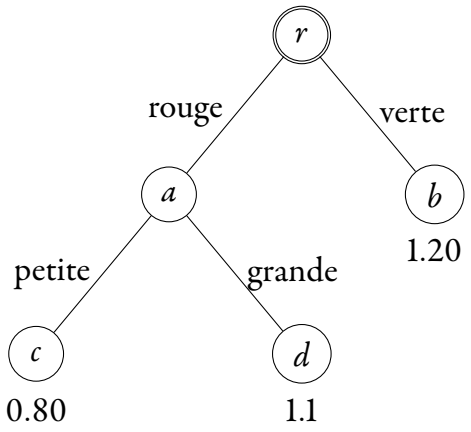
Remarque. — On peut facilement démontrer que $(\mathcal{X}^{(v)})_{v \in \mathrm{V} \setminus \pi(\mathrm{V})}$ forme une partition de $\mathcal{X}$. Cela entraîne que l'arbre de décision défini ci-dessus vérifie pour tout $v \in \mathrm{V} \setminus \pi(\mathrm{V})$:

$$f(x) = y^{(v)} \quad \Longleftrightarrow \quad x \in \mathcal{X}^{(v)}.$$

Exemple. — On considère un problème où il s'agit de déterminer le prix d'une pomme en fonction de sa couleur et de sa taille :

$$\mathcal{X} = \{\text{rouge, verte}\} \times \{\text{petite, grande}\} \times \{\text{bio, non bio}\} \quad \text{et} \quad \mathcal{Y} = \mathbb{R}_+.$$

L'arbre de décision donné de façon graphique :



peut se formaliser de la façon suivante. Les ensembles $(\mathcal{X}^{(v)})_{v\in V}$ sont donnés par :

$$\begin{aligned}
\mathcal{X}^{(r)} &= \mathcal{X} \\
\mathcal{X}^{(a)} &= \{x \in \mathcal{X} \mid x_1 = \text{rouge}\} \\
\mathcal{X}^{(b)} &= \{x \in \mathcal{X} \mid x_1 = \text{verte}\} \\
\mathcal{X}^{(c)} &= \{x \in \mathcal{X} \mid x_1 = \text{rouge et } x_2 = \text{petite}\} \\
\mathcal{X}^{(d)} &= \{x \in \mathcal{X} \mid x_1 = \text{rouge et } x_2 = \text{grande}\},
\end{aligned}$$

et les sorties $(y^{(v)})_{v\in V\setminus\pi(V)}$ par :

$$y^{(b)} = 1.20 \qquad y^{(c)} = 0.80 \qquad y^{(d)} = 1.1.$$

Remarque. — *Algorithmes d'apprentissage donnant des arbres de décision.* — Parmi les algorithmes d'apprentissage donnant des arbres de décision, les plus connus sont ID3, C4.5 et CART. Le principe général de ces algorithmes est commun. L'arbre est créé en partant de la racine et à chaque nœud v, la partition correspondant aux nœuds *successeurs* est choisie de la forme

$$(\{x \in \mathcal{X}^{(v)} \mid x_i < z\}, \{x \in \mathcal{X}^{(v)} \mid x_i \geqslant z\}),$$

où i désigne une des variables explicatives (qui est donc nécessairement quantitative pour les algorithmes concernés). La partition retenue est celle minimisant un critère d'*impureté*, de sorte que les sorties correspondant à chacun des deux ensembles de la partition soient les plus *homogènes* possibles. L'algorithme s'arrête à l'aide de différents critères, l'un d'eux étant une limite à la profondeur de l'arbre, qui est un des hyperparamètres principaux. Une fois la création des nœuds terminée, une deuxième phase éventuelle d'*élagage* peut avoir lieu, où des nœuds sont supprimés selon certaines règles, et qui peut être vue comme une forme de régularisation. Enfin, lorsque la structure de l'arbre est définitivement fixée, l'étiquette $y^{(v)}$ correspondant à chaque feuille $v \in \mathrm{V} \backslash \pi(\mathrm{V})$ est définie comme étant celle majoritaire parmi les étiquettes des exemples appartenant à l'ensemble $\mathscr{X}^{(v)}$ correspondant.

2. BAGGING

On présente dans ce paragraphe le *Bagging* (*Bootstrap aggregating*), qui est une méthode ensembliste où à partir d'un échantillon d'apprentissage initial, on tire de façon aléatoire des sous-échantillons afin de les fournir à un algorithme d'apprentissage donné, et obtenir ainsi différents prédicteurs (un grand nombre, le plus souvent). Ces prédicteurs sont ensuite agrégés pour donner le prédicteur final.

Soit $\mathscr{X}$ un ensemble d'entrées quelconque et on se place dans un cadre de régression ($\mathscr{Y} = \mathbb{R}$) ou de classification ($\mathscr{Y} = [\mathrm{N}]$, $\mathrm{N} \geqslant 1$). Soit $\mathrm{A} \in \mathscr{A}(\mathscr{X}, \mathscr{Y})$ un algorithme d'apprentissage, $n \geqslant 1$ un entier et $\mathrm{S} = (x_i, y_i)_{i \in [n]} \in \mathscr{S}(\mathscr{X}, \mathscr{Y})$ l'échantillon d'apprentissage initial. Soit $m \geqslant 1$ le nombre de prédicteurs à agréger et $n' \geqslant 1$ la taille des échantillons d'apprentissage secondaires.

Pour chaque $k \in [m]$,

- on se donne un échantillon $\mathrm{S}^{(k)} = (x_i^{(k)}, y_i^{(k)})_{i \in [n']}$ dont les exemples sont tirés indépendamment et uniformément parmi ceux de S (tirage avec remise, donc) ;
- puis on entraîne un prédicteur $\widehat{f}^{(k)}$ avec l'algorithme A et $\mathrm{S}^{(k)}$ pour échantillon d'apprentissage, autrement dit $\widehat{f}^{(k)} = \mathrm{A}(\mathrm{S}^{(k)})$.

En régression, le prédicteur final est donné par la moyenne des prédicteurs :

$$\forall x \in \mathscr{X}, \quad \widehat{f}(x) = \frac{1}{m} \sum_{k=1}^{m} \widehat{f}^{(k)}(x).$$

En classification, le prédicteur final est donné par l'étiquette majoritaire (ou *une* étiquette majoritaire) :

$$\forall x \in \mathcal{X}, \quad \hat{f}(x) = \arg\max_{y \in \mathcal{Y}} \operatorname{Card} \left\{ k \in [m] \mid \hat{f}^{(k)}(x) = y \right\}.$$

REMARQUE. — Il existe plusieurs variantes du Bagging. Le *Pasting* est similaire, mais les échantillon d'apprentissage y sont tirés sans remise. L'algorithme *Random subspaces*, défini dans le cas où l'ensemble d'entrées $\mathcal{X}$ s'écrit comme un produit cartésien, considère toujours le même échantillon d'apprentissage, mais pour chaque $k \in [m]$, seul un sous-ensemble aléatoire de variables explicatives est considéré. Lorsque le Bagging est combiné au Random subspaces et que l'algorithme initial donne des arbres de décision, on parle de *Forêts aléatoires*.

3. ADABOOST EN CLASSIFICATION BINAIRE

Le *Boosting* est une famille de méthodes ensemblistes qui consiste à construire les prédicteurs à agréger de façon séquentielle, chaque prédicteur cherchant à corriger les erreurs des précédents.

On se place dans un contexte de classification binaire : soit $\mathcal{X}$ un ensemble d'entrées quelconque et $\mathcal{Y} = \{-1, 1\}$. Soit $n \geqslant 1$ un entier et $S = (x_i, y_i)_{i \in [n]} \in \mathcal{S}(\mathcal{X}, \mathcal{Y})$ un échantillon d'apprentissage. On appelle n-simplexe l'ensemble :

$$\Delta_n = \left\{ \pi \in \mathbb{R}_+^n \;\middle|\; \sum_{i=1}^{n} \pi_i = 1 \right\}.$$

On identifie Δ_n avec l'ensemble des distributions de probabilité sur $[n]$. Pour $\pi \in \Delta_n$, on dit qu'on tire l'exemple (x_0, y_0) dans l'échantillon S selon la distribution π si :

$$\forall i \in [n], \quad \mathbb{P}\left[(x_0, y_0) = (x_i, y_i)\right] = \pi_i.$$

Soit $A \in \mathcal{A}(\mathcal{X}, \mathcal{Y})$ un algorithme d'apprentissage, $m \geqslant 1$ le nombre de prédicteurs à agréger, et $n' \geqslant 1$ la taille des échantillons d'apprentissage secondaires.

AdaBoost est alors défini comme suit. On pose $\pi^{(1)} = (1/n, \dots, 1/n) \in \Delta_n$ (ce qui correspond à la distribution uniforme sur $[n]$). Puis pour $k \in [m]$,

— on se donne $S^{(k)} = (x_i^{(k)}, y_i^{(k)})_{i \in [n']}$ un échantillon dont les exemples sont tirés indépendamment dans l'échantillon S selon la distribution $\pi^{(k)}$;

— on entraîne $\widehat{f}^{(k)}$ avec l'algorithme A et $S^{(k)}$ pour échantillon d'apprentissage, autrement dit $\widehat{f}^{(k)} = A(S^{(k)})$;
— on calcule :

$$\varepsilon^{(k)} = \sum_{i=1}^{n} \pi_i^{(k)} \mathbb{1}_{\left\{y_i \neq \widehat{f}^{(k)}(x_i)\right\}},$$
$$w^{(k)} = \frac{1}{2} \log\left(\frac{1}{\varepsilon^{(k)}} - 1\right) ;$$

— on pose :

$$\pi^{(k+1)} = \left(\frac{\pi_i^{(k)} \exp\left(-w^{(k)} y_i \widehat{f}^{(k)}(x_i)\right)}{\sum_{j=1}^{n} \pi_j^{(k)} \exp\left(-w^{(k)} y_j \widehat{f}^{(k)}(x_j)\right)} \right)_{i \in [n]},$$

ce qui assure que $\pi^{(k+1)} \in \Delta_n$.
Le prédicteur final est donné par :

$$\forall x \in \mathcal{X}, \quad \widehat{f}(x) = \operatorname{sign}\left(\sum_{k=1}^{m} w^{(k)} \widehat{f}^{(k)}(x) \right).$$

Remarque. — À chaque étape, AdaBoost constitue un échantillon d'apprentissage où les exemples se retrouvent avant une probabilité d'autant plus grande qu'ils on été mal prédits par les prédicteurs précédents, de sorte à obtenir un prédicteur qui prédit mieux sur ces exemples.

Théorème. — *S'il existe $\gamma > 0$ tel que :*

$$\forall k \in [m], \quad \varepsilon^{(k)} \leqslant \frac{1}{2} - \gamma,$$

alors l'erreur d'apprentissage du prédicteur final est majoré de la façon suivante :

$$\frac{1}{n} \sum_{i=1}^{n} \mathbb{1}_{\left\{\widehat{f}(x_i) \neq y_i\right\}} \leqslant e^{-2\gamma^2 m}.$$

Démonstration. — Voir TD. □