Introduction au Machine Learning

Boosting

Théo Lopès-Quintas

BPCE Payment Services,
Université Paris Dauphine

2022-2025

AdaBoost

Modélisation

- T le nombre d'itérations[1] que l'on réalisera
- $w_t^{(i)}$ la note comprise entre 0 et 1 à l'itération $t \leqslant T$ pour l'observation $i \leqslant n$
- h_θ un weak learner[2] d'AdaBoost paramétré par le vecteur d'information θ

Nombre d'époques

$$f_\theta(x) = \sum_{t=1}^{T} \alpha_t \, h_{\theta_t}(x) \tag{AdaBoost}$$

Note du weak learner de l'époque t

1. On parle également d'époques.
2. Dans le cas d'AdaBoost on parle de souche : arbre de profondeur 1 ou 2.

AdaBoost

Comment noter la souche ?

$$\theta_t = \underset{\theta \in \Theta}{\arg\min} \frac{\sum_{i=1}^{n} w_t^{(i)} \mathbb{1}_{\{y_i \neq h_\theta(x^{(i)})\}}}{\sum_{i=1}^{n} \underbrace{\boxed{w_t^{(i)}}}_{\text{Note de l'observation } i}} \qquad \text{(Apprentissage des souches)}$$

$$\varepsilon_t = \frac{\sum_{i=1}^{n} w_t^{(i)} \mathbb{1}_{\{y_i \neq h_\theta(x^{(i)})\}}}{\sum_{i=1}^{n} w_t^{(i)}} \qquad \text{(Calcul de l'erreur } \varepsilon_t\text{)}$$

$$\alpha_t = \frac{1}{2} \ln\left(\frac{1 - \varepsilon_t}{\varepsilon_t}\right) \qquad \text{(Note de la souche)}$$

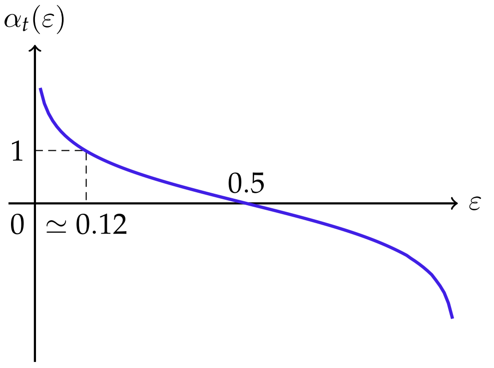


Figure – Graphe de la fonction : $x \mapsto \frac{1}{2} \ln \left(\frac{1-x}{x} \right)$

Exercice 1 (Etude de *alpha*$_t$)

Soit $t \leqslant T$ une époque, on s'appuiera sur la figure (1).

1. *Montrer que $\varepsilon_t \in [0, 1]$*
2. *Commenter la forme de fonction quand ε est au voisinage de 0.5. Même question pour 0 et pour 1.*

AdaBoost

Comment apprendre ?

- Initialisation : Nombre d'époques T et initialiser les notes des observations
- Pour chaque époque :
  1. Trouver le meilleur paramétrage pour une souche dans un problème prenant en compte la difficulté de classification de chaque observation
  2. Calculer la note du weak learner appris
  3. Mettre à jour les notes des observations à l'aide de la formule :

$$\forall i \leqslant n,\ w_{t+1}^{(i)} = w_t^{(i)} e^{-\alpha_t (2h_{\theta_t}(x^{(i)}) - 1)(2y_i - 1)}$$

$$f_0 = \arg\min_{\gamma \in \mathbb{R}} \sum_{i=1}^{n} \mathcal{L}(y_i, \gamma) \qquad \text{(Initialisation du boosting)}$$

Prédire un dataset par une constante n'est pas très performant. Donc on cherche à l'améliorer itérativement. Ainsi, on cherche à améliorer f_{m-1} à l'étape m de sorte que :

$$\begin{aligned} \forall i \leqslant n,\ f_m\left(x^{(i)}\right) &= f_{m-1}\left(x^{(i)}\right) + h_m\left(x^{(i)}\right) = y_i \\ &\Longleftrightarrow \\ \forall i \leqslant n,\ h_m\left(x^{(i)}\right) &= y_i - f_{m-1}\left(x^{(i)}\right) \end{aligned}$$

Exercice 2 (Descente de gradient et résidus)

Soit la fonction de perte $\mathcal{L}(y,f(x)) = (y-f(x))^2$ *et la fonction de coût* $\mathcal{C}(y,f(x)) = \sum_{i=1}^{n} \mathcal{L}\left(y_i, f\left(x^{(i)}\right)\right)$.

Montrer que :

$$-\frac{\partial \mathcal{C}}{\partial f_{m-1}\left(x^{(i)}\right)}\left(y_i, f_{m-1}\left(x^{(i)}\right)\right) = \frac{2}{n} h_m\left(x^{(i)}\right)$$

Le résultat de cet exercice se généralise, il y a un lien entre l'opposé du gradient de la fonction de coût et les résidus. Ainsi, si l'on compile les différentes équations que l'on a écrites jusqu'à présent on a :

$$\begin{aligned} f_m(x) &= f_{m-1}(x) - \gamma \sum_{i=1}^{n} \frac{\partial \mathcal{C}}{\partial f_{m-1}\left(x^{(i)}\right)}\left(y_i, f_{m-1}\left(x^{(i)}\right)\right) \\ &= f_{m-1}(x) + \gamma' h_m(x) \end{aligned}$$

Gradient Boosting

Comment apprendre ?

On peut optimiser la valeur de γ pour qu'elle prenne la valeur qui minimise la fonction de perte :

$$\gamma_m = \underset{\gamma \in \mathbb{R}}{\arg\min} \sum_{i=1}^{n} \mathcal{L}\left(y_i, f_{m-1}\left(x^{(i)}\right) + \gamma h_m\left(x^{(i)}\right)\right)$$

Ainsi, on exploite à nouveau la théorie de la descente de gradient pour définir un learning rate $\eta \in]0, 1]$. Finalement, on peut résumer le Gradient Boosting à :

$$\begin{aligned} f_0(x) &= \underset{\gamma \in \mathbb{R}}{\arg\min} \sum_{i=1}^{n} \mathcal{L}(y_i, \gamma) && \text{(Initialisation)} \\ \forall m \leqslant 1,\ f_m(x) &= f_{m-1}(x) + \eta \gamma_m h_m(x) && \text{(Itération)} \\ F(x) &= \sum_{m=0}^{M} \gamma_m h_m(x) \text{ avec } \gamma_0 = 1 && \text{(Strong learner)} \end{aligned}$$

Gradient Boosting

En résumé

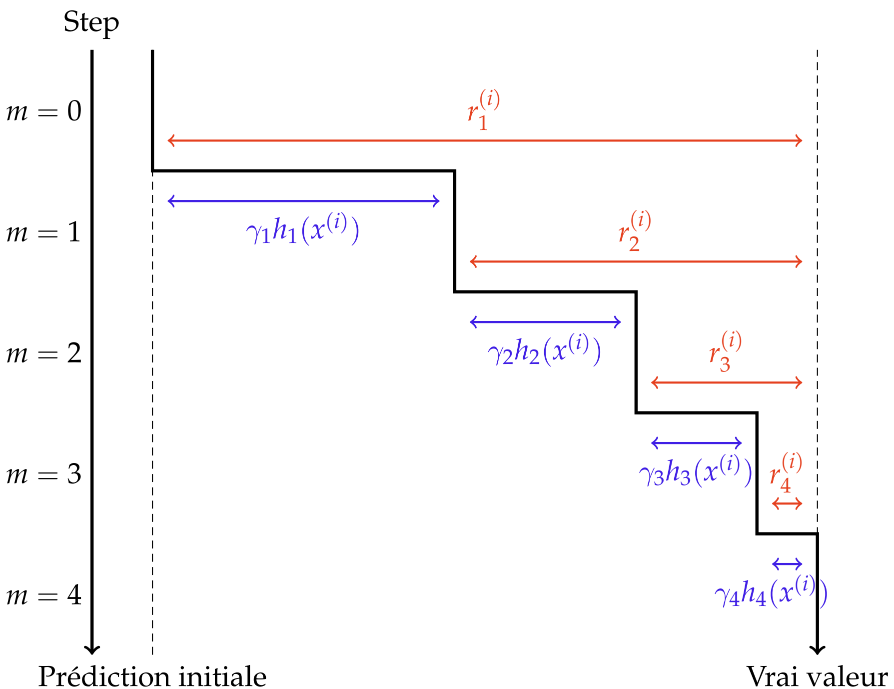


Figure – Principe du Gradient Boosting pour une observation

Si l'on reprend les explications de l'algorithme de Gradient Boosting classique, à chaque étape nous choisissons un arbre h_m qui répond au problème :

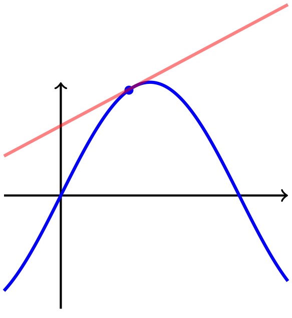
$$h_m^* = \underset{h_m \text{ possible}}{\arg\min} \sum_{i=1}^{n} \mathcal{L} \left(y_i, f_{m-1} \left(x^{(i)} \right) + h_m \left(x^{(i)} \right) \right)$$

Théorème 1 (Taylor)

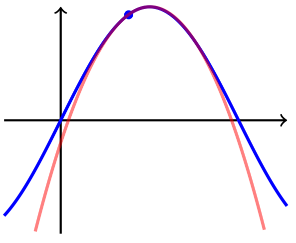
Soit $I \subset \mathbb{R}$ et $a \in I$. Soit $f : I \mapsto \mathbb{R}$ une fonction n-fois dérivable en a. Alors :

$$f(x) = \sum_{k=0}^{n} \frac{f^{(k)}(a)}{k!}(x-a)^k + R_n(x)$$

Avec le reste $R_n(x)$ négligeable devant $(x-a)^n$ au voisinage de a.



(a) Développement à l'ordre 1



(b) Développement à l'ordre 2

Figure – Développement de Taylor pour $f(x) = 2\sin(x)$ au point $x = 1.3$

Exercice 3 (Nouvelle fonction de coût)

Nous reprenons l'ensemble des notations définies jusqu'à présent.

1. *Soit* $f : \mathcal{I} \mapsto \mathbb{R}$ *une fonction* n *fois dérivable,* $a \in I$ *et* $h \in \mathbb{R}$ *tel que* $a + h \in I$*. Justifier :*

$$f(a+h) = \sum_{k=0}^{n} \frac{f^{(k)}(a)}{k!} h^k + R_n(h)$$

   Avec $R_n(x)$ *une fonction négligeable devant* h^n *au voisinage de* 0*.*

2. *A l'aide de l'expression précédente, proposer une approximation à l'ordre 2 de l'expression :*

$$\phi\left(f_{m-1}\left(x^{(i)}\right) + h_m\left(x^{(i)}\right)\right) = \sum_{i=1}^{n} \mathcal{L}\left(y_i, f_{m-1}\left(x^{(i)}\right) + h_m\left(x^{(i)}\right)\right)$$

   Où $\mathcal{L}$ *est une fonction dérivable deux fois sur* $\mathbb{R}$*.*

3. *Nous obtenons une approximation du problème du choix du meilleur weak learner* h_m*. Identifier les termes constants et commenter sur la vitesse de calcul par rapport à la méthode classique.*

XGBoost : une amélioration

Solution et avantages

Pour la question 3, en reprenant l'expression précédente et en écrivant en rouge les termes constants pour chacune des itérations, on a :

$$h_m^* = \underset{h_m \text{ possible}}{\arg\min} \sum_{i=1}^{n} \mathcal{L}\left(y_i, \hat{y}_i^{(m-1)}\right) + \nabla \mathcal{L}\left(y_i, \hat{y}_i^{(m-1)}\right) h_m\left(x^{(i)}\right) + \frac{1}{2} \nabla^2 \mathcal{L}\left(y_i, \hat{y}_i^{(m-1)}\right) h_m\left(x^{(i)}\right)$$

XGBoost : une amélioration

Paramétrages

Il existe d'autres algorithmes majeurs de Gradient Boosting qui ont chacun leurs spécificités et atout.
Pour chacun des algorithmes, les principaux hyperparamètres sont :

- **Paramétrer les arbres**
  - criterion : pour définir la métrique à utiliser pour faire une coupure
  - max_depth : limiter la profondeur maximale d'un arbre
  - min_samples_leaf : nombre minimal d'observations dans une feuille
  - max_features : nombre d'informations à considérer pour chaque coupure

XGBoost : une amélioration

Paramétrages

Il existe d'autres algorithmes majeurs de Gradient Boosting qui ont chacun leurs spécificités et atout.
Pour chacun des algorithmes, les principaux hyperparamètres sont :

- **Paramétrer les arbres**
  - criterion : pour définir la métrique à utiliser pour faire une coupure
  - max_depth : limiter la profondeur maximale d'un arbre
  - min_samples_leaf : nombre minimal d'observations dans une feuille
  - max_features : nombre d'informations à considérer pour chaque coupure
- **Paramétrer le boosting**
  - n_estimators : nombre d'arbres à construire dans la forêt
  - learning_rate : pas de descente pour réduire le poids des arbres successifs
  - subsample : fraction des données à utiliser pour apprendre chaque weak learner. Si inférieur à 1, alors on obtient une descente de Gradient Stochastique
  - init : premier modèle qui sera amélioré. Si non renseigné, un modèle très simple sera utilisé

XGBoost : une amélioration

Paramètrages

Il existe d'autres algorithmes majeurs de Gradient Boosting qui ont chacun leurs spécificités et atout. Pour chacun des algorithmes, les principaux hyperparamètres sont :

- **Paramétrer les arbres**
  - criterion : pour définir la métrique à utiliser pour faire une coupure
  - max_depth : limiter la profondeur maximale d'un arbre
  - min_samples_leaf : nombre minimal d'observations dans une feuille
  - max_features : nombre d'informations à considérer pour chaque coupure
- **Paramétrer le boosting**
  - n_estimators : nombre d'arbres à construire dans la forêt
  - learning_rate : pas de descente pour réduire le poids des arbres successifs
  - subsample : fraction des données à utiliser pour apprendre chaque weak learner. Si inférieur à 1, alors on obtient une descente de Gradient Stochastique
  - init : premier modèle qui sera amélioré. Si non renseigné, un modèle très simple sera utilisé
- **Pour arrêter plus tôt le boosting**
  - validation_fraction : proportion des données d'entraînement à conserver pour tester l'early-stopping
  - n_iter_no_change : nombre minimal d'itérations sans améliorations avant d'arrêter l'apprentissage
  - tol : valeur minimale de modification de la loss qui déclenche l'arrêt prématuré