Méthodes d'Ensemble : Bagging, Boosting et Forêts Aléatoires

1 Cadre général

Soit un jeu de données

$$\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{n}, \quad x_i \in \mathbb{R}^p,\ y_i \in \mathcal{Y}.$$

On cherche à approximer la fonction de régression ou de décision

$$f^*(x) = \mathbb{E}[Y \mid X = x]$$

par un estimateur $\hat{f}(x)$.

Les méthodes d'ensemble combinent plusieurs prédicteurs faibles afin d'obtenir un prédicteur plus performant :

$$\hat{f}(x) = \sum_{m=1}^{M} \alpha_m h_m(x).$$

2 Principe général des méthodes d'ensemble

Les méthodes d'ensemble travaillent sur le principe suivant : au lieu d'apprendre un seul modèle complexe, on apprend plusieurs modèles simples appelés *apprenants faibles*, puis on les combine.

Chaque modèle faible capture une partie de la structure des données :

- une région locale de l'espace des variables,
- un motif simple,
- ou une erreur laissée par les modèles précédents.

L'objectif est d'approcher la fonction cible $f^*(x)$ par une combinaison de fonctions simples :

$$f^*(x) \approx \sum_{m=1}^{M} \alpha_m h_m(x).$$

3 Bagging (Bootstrap Aggregating)

3.1 Définition

On génère B échantillons bootstrap :

$$\mathcal{D}^{(b)} \sim \text{Bootstrap}(\mathcal{D}), \quad b = 1, \ldots, B.$$

Chaque modèle est entraîné indépendamment :

$$\hat{f}_b(x) = \hat{f}(x; \mathcal{D}^{(b)}).$$

L'agrégation est donnée par :

$$\hat{f}_{\text{bag}}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{f}_b(x).$$

3.2 Réduction de variance

Supposons :

$$\mathbb{E}[\hat{f}_b(x)] = \mu, \quad \text{Var}(\hat{f}_b(x)) = \sigma^2,$$

et une corrélation paire constante :

$$\text{Corr}(\hat{f}_b(x), \hat{f}_{b'}(x)) = \rho.$$

Alors :

$$\begin{aligned} \text{Var}(\hat{f}_{\text{bag}}(x)) &= \text{Var}\left(\frac{1}{B}\sum_{b=1}^{B} \hat{f}_b(x)\right) \\ &= \frac{1}{B^2}\left(B\sigma^2 + B(B-1)\rho\sigma^2\right) \\ &= \sigma^2\left(\rho + \frac{1-\rho}{B}\right). \end{aligned}$$

Ainsi, lorsque $B \to \infty$:

$$\text{Var}(\hat{f}_{\text{bag}}(x)) \to \rho\sigma^2.$$

3.3 Interprétation du Bagging

Le bagging agit principalement sur la variance du modèle. Les arbres de décision étant instables, de petites variations des données produisent des modèles très différents.

Le bootstrap génère des jeux de données légèrement différents :

$$\mathcal{D}^{(b)} \neq \mathcal{D}^{(b')}.$$

Chaque modèle apprend une approximation différente de $f^*(x)$. La moyenne de ces prédictions permet d'annuler les fluctuations aléatoires, ce qui réduit la variance sans augmenter significativement le biais.

4 Random Forest : principe et interprétation

4.1 Motivation

Les arbres de décision sont des modèles très flexibles, mais fortement instables : de petites variations dans les données peuvent entraîner des arbres très différents.

Le Random Forest vise à réduire cette instabilité en combinant un grand nombre d'arbres faiblement corrélés.

4.2 Principe mathématique

Chaque arbre $\hat{f}_b(x)$ est appris sur :

- un échantillon bootstrap $\mathcal{D}^{(b)}$,
- une sélection aléatoire de variables à chaque nœud.

Le prédicteur final est donné par :

$$\hat{f}_{\mathrm{RF}}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{f}_b(x).$$

4.3 Sur quoi travaille le Random Forest

Chaque arbre partitionne l'espace des variables $\mathbb{R}^p$ en régions rectangulaires disjointes. À l'intérieur de chaque région, la prédiction est constante.

La sélection aléatoire des variables force les arbres à explorer différentes structures des données, ce qui diminue la corrélation entre modèles.

4.4 Interprétation biais–variance

Le Random Forest agit principalement sur la variance :

- la moyenne réduit les fluctuations aléatoires,
- la décorrélation des arbres améliore la généralisation.

Le biais peut être légèrement supérieur à celui d'un arbre unique, mais la réduction de variance domine généralement, ce qui conduit à de meilleures performances globales.

5 Boosting : modèle additif

Le boosting construit un modèle additif (après m itération):

$$F_M(x) = \sum_{m=1}^{M} \nu \alpha_m h_m(x),$$

où $\nu \in (0, 1]$ est le taux d'apprentissage (shrinkage), $h(x)$: prédicteur faible (faible apprenant), $h_m(x)$: prédicteur faible appris à l'itération m et $\alpha_m : poids du modèle faible \mathrm{h}_m(x)$.

6 Interprétation du Boosting

Contrairement au bagging, le boosting est un processus séquentiel. Chaque nouveau modèle est entraîné pour corriger les erreurs du modèle précédent.

Mathématiquement, le boosting réalise une optimisation itérative dans l'espace des fonctions :

$$F_m = F_{m-1} + \nu h_m.$$

Chaque itération se déplace dans la direction qui réduit le plus rapidement la fonction de perte globale.

7 AdaBoost

7.1 Cadre

AdaBoost repose sur l'idée que plusieurs classifieurs très simples, légèrement meilleurs que le hasard, peuvent être combinés pour obtenir un classifieur puissant.

L'algorithme met l'accent sur les observations difficiles à chaque itération. On considère :

$$y_i \in \{-1, +1\}, \quad h_m(x) \in \{-1, +1\}.$$

Le modèle est :

$$F_M(x) = \sum_{m=1}^{M} \alpha_m h_m(x), \quad \hat{y} = \text{sign}(F_M(x)).$$

où chaque prédicteur faible h_m est entraîné en utilisant une distribution de poids $w_i^{(m)}$.

Les poids augmentent pour les observations mal classées :

$$w_i^{(m+1)} = w_i^{(m)} \exp(-\alpha_m y_i h_m(x_i)).$$

7.2 Fonction de perte exponentielle

AdaBoost minimise :

$$\mathcal{L}(F) = \sum_{i=1}^{n} \exp(-y_i F(x_i)).$$

7.3 Optimisation stagewise

À l'itération m :

$$w_i^{(m)} = \exp(-y_i F_{m-1}(x_i)).$$

Erreur pondérée :

$$\varepsilon_m = \sum_{i=1}^{n} w_i^{(m)} \mathbb{I}(y_i \neq h_m(x_i)).$$

Le problème revient à minimiser :

$$(1 - \varepsilon_m)e^{-\alpha} + \varepsilon_m e^{\alpha}.$$

Dérivation :

$$\frac{d}{d\alpha} = -(1 - \varepsilon_m)e^{-\alpha} + \varepsilon_m e^{\alpha} = 0.$$

On obtient :

$$\boxed{\alpha_m = \frac{1}{2} \ln\left(\frac{1 - \varepsilon_m}{\varepsilon_m}\right)}$$

Mise à jour des poids :

$$w_i^{(m+1)} = w_i^{(m)} \exp(-\alpha_m y_i h_m(x_i)).$$

7.4 Sur quoi travaille AdaBoost

AdaBoost travaille directement sur les erreurs de classification. À chaque itération :

- les observations bien classées deviennent moins importantes,
- les observations mal classées deviennent prioritaires.

Ainsi, les nouveaux modèles se concentrent sur les régions de l'espace où le classifieur est faible.

7.5 Interprétation biais–variance

AdaBoost réduit principalement le biais :

- chaque itération améliore la frontière de décision,
- le modèle devient progressivement plus expressif.

Cependant, si les données contiennent du bruit, l'accent excessif mis sur certaines observations peut augmenter la variance.

8 Gradient Boosting

8.1 Motivation

Le Gradient Boosting généralise AdaBoost à une large famille de fonctions de perte. L'objectif n'est plus uniquement la classification binaire, mais l'optimisation directe d'une fonction de perte arbitraire (régression, classification, survie, etc.).

Le Gradient Boosting construit progressivement une fonction $F(x)$ qui minimise le risque empirique :

$$\mathcal{R}(F) = \sum_{i=1}^{n} L(y_i, F(x_i)).$$

8.2 Descente de gradient dans l'espace des fonctions

Contrairement aux méthodes classiques qui effectuent une descente de gradient dans l'espace des paramètres, le Gradient Boosting effectue une descente dans l'espace des fonctions.

À l'itération m, on cherche une fonction $h_m(x)$ telle que :

$$F_m(x) = F_{m-1}(x) + \nu h_m(x),$$

où ν est le taux d'apprentissage.

La direction optimale est donnée par le gradient fonctionnel négatif :

$$r_i^{(m)} = - \left. \frac{\partial L(y_i, F(x_i))}{\partial F(x_i)} \right|_{F=F_{m-1}}.$$

Ces quantités $r_i^{(m)}$ sont appelées *pseudo-résidus*.

8.3 Rôle des arbres de décision

Les pseudo-résidus $\{r_i^{(m)}\}$ ne dépendent que des observations. On entraîne alors un arbre de régression $h_m(x)$ pour approximer la relation :

$$x_i \longmapsto r_i^{(m)}.$$

L'arbre partitionne l'espace des variables en régions, dans lesquelles les erreurs résiduelles sont similaires.

8.4 Mise à jour du modèle

Après l'apprentissage de h_m, on cherche un coefficient optimal :

$$\gamma_m = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) + \gamma h_m(x_i)).$$

La mise à jour finale est :

$$F_m(x) = F_{m-1}(x) + \nu \gamma_m h_m(x).$$

8.5 Interprétation biais–variance

Le Gradient Boosting agit principalement sur le biais :

- chaque itération corrige une partie de l'erreur systématique,
- les modèles faibles sont volontairement simples (arbres peu profonds).

Le paramètre ν et le nombre d'itérations M permettent de contrôler la variance et d'éviter le sur-apprentissage.

8.6 Lien avec AdaBoost

AdaBoost est un cas particulier du Gradient Boosting, correspondant à la perte exponentielle :

$$L(y, F(x)) = \exp(-yF(x)).$$

Dans ce cas, les pseudo-résidus sont proportionnels aux poids utilisés par AdaBoost.

9 XGBoost

9.1 Objectif régularisé

XGBoost a été conçu pour améliorer le Gradient Boosting en introduisant une régularisation explicite et une optimisation plus efficace.

Il vise à obtenir un compromis optimal entre précision et complexité du modèle. XGBoost minimise une fonction objectif régularisée :

$$\mathcal{L} = \sum_{i=1}^{n} \ell(y_i, \hat{y}_i) + \sum_{t=1}^{T} \Omega(f_t),$$

$$\Omega(f) = \gamma K + \frac{\lambda}{2} \sum_{j=1}^{K} w_j^2.$$

où $\Omega(f)$ pénalise la complexité de chaque arbre.

L'utilisation du développement de Taylor d'ordre 2 permet une optimisation plus précise à chaque itération.

9.2 Approximation de Taylor

À l'itération t :

$$\ell(y_i, \hat{y}_i + f_t(x_i)) \approx \ell(y_i, \hat{y}_i) + g_i f_t(x_i) + \frac{1}{2} h_i f_t(x_i)^2.$$

où :

$$g_i = \frac{\partial \ell}{\partial \hat{y}_i}, \quad h_i = \frac{\partial^2 \ell}{\partial \hat{y}_i^2}.$$

9.3 Optimisation par feuille

Soit une feuille j :

$$G_j = \sum_{i \in I_j} g_i, \quad H_j = \sum_{i \in I_j} h_i.$$

Poids optimal :

$$\boxed{w_j^* = -\frac{G_j}{H_j + \lambda}}$$

Valeur optimale :

$$\boxed{\mathcal{L}^{(t)} = -\frac{1}{2} \sum_{j=1}^{K} \frac{G_j^2}{H_j + \lambda} + \gamma K}$$

9.4 Gain d'un split

$$\boxed{\text{Gain} = \frac{1}{2} \left(\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{G^2}{H + \lambda} \right) - \gamma}$$

9.5 Sur quoi travaille XGBoost

À chaque itération, XGBoost travaille sur :

- le gradient g_i (erreur locale),
- le Hessien h_i (incertitude locale).

Les arbres sont construits de manière à regrouper les observations ayant des gradients similaires, ce qui permet une correction ciblée des erreurs.

9.6 Interprétation biais–variance

XGBoost agit sur les deux composantes :

- réduction du biais par l'apprentissage itératif,
- contrôle de la variance par la régularisation.

Les paramètres λ, γ, le sous-échantillonnage et le taux d'apprentissage permettent un contrôle fin du sur-apprentissage.

10 Conclusion biais–variance

- Bagging / Random Forest : réduction de la variance.
- Boosting : réduction du biais par optimisation additive.
- XGBoost : boosting + régularisation + information du second ordre.