Polycopié Complet

From Decision and Regression Trees to Ensemble Methods

Florence d'Alché-Buc & Ekhine Irurozki
Télécom Paris, IP Paris

Table des matières

1 Introduction et Contexte

Ce polycopié couvre la deuxième conférence du cours SD-TSIA-210, dont l'objectif est d'explorer :

— Les arbres de décision et de régression
— Les méthodes d'ensemble (ensemble methods) appliquées à l'apprentissage supervisé
— Les méthodes de bagging, forêts aléatoires (Random Forests) et boosting (AdaBoost et Gradient Boosting)

Nous verrons comment ces techniques s'inscrivent dans le cadre de la minimisation du risque empirique, en mettant l'accent sur la construction de partitions de l'espace d'entrée et la combinaison de plusieurs modèles pour réduire la variance et améliorer la performance.

2 Cadre de l'Apprentissage Supervisé

2.1 Notations et Problématique

Soient :

— $X \in \mathcal{X} \subset \mathbb{R}^p$, un vecteur aléatoire représentant les caractéristiques.
— Y une variable aléatoire dont le domaine dépend du problème :
  — $Y = \mathbb{R}$ pour la régression.
  — $Y = \{-1, 1\}$ pour la classification binaire.
  — $Y = \{1, \ldots, C\}$ pour la classification multi-classes.
— D la distribution conjointe de (X, Y).

La tâche de l'apprentissage consiste à trouver une fonction $f : \mathcal{X} \to Y$ qui minimise le risque vrai

$$R(f) = \mathbb{E}_{(X,Y)\sim D}[\ell(Y, f(X))],$$

à partir d'un échantillon d'apprentissage $S = \{(x_i, y_i),\ i = 1, \ldots, n\}$.

2.2 Minimisation du Risque Empirique et Régularisation

On définit la minimisation du risque empirique dans un espace d'hypothèses $\mathcal{H}$ par :

$$\hat{f} \in \arg\min_{f\in\mathcal{H}} \frac{1}{n}\sum_{i=1}^{n} \ell(y_i, f(x_i)) \quad \text{sous contrainte } \Omega(f) \leq C,$$

ou de manière pénalisée :

$$\hat{f} \in \arg\min_{f\in\mathcal{H}} \left\{ \frac{1}{n}\sum_{i=1}^{n} \ell(y_i, f(x_i)) + \lambda\, \Omega(f) \right\}.$$

3 Arbres de Décision et de Régression

3.1 Présentation Générale

Les arbres sont des modèles prédictifs qui partitionnent l'espace d'entrée en régions homogènes. On exprime généralement le modèle sous la forme :

$$f(x) = \sum_{\ell=1}^{M} \mathbf{1}\{x \in R_\ell\} f_\ell,$$

où les R_ℓ forment une partition de $\mathcal{X}$ et f_ℓ est la prédiction associée à la région R_ℓ.

3.2 Arbres pour la Classification

Pour la classification à C classes, la prédiction dans chaque feuille est obtenue par un vote majoritaire :

$$f_\ell = \arg\max_{c\in\{1,\dots,C\}} \hat{p}_{\ell,c} \quad \text{avec} \quad \hat{p}_{\ell,c} = \frac{1}{N_\ell}\sum_{x_i\in R_\ell} \mathbf{1}\{y_i = c\}.$$

Le choix du split se fait en minimisant une mesure d'impureté (Gini, entropie, etc.).

3.3 Arbres pour la Régression

Pour la régression, la prédiction dans chaque région est la moyenne des valeurs :

$$f_\ell = \frac{1}{N_\ell}\sum_{x_i\in R_\ell} y_i.$$

Le critère de split est souvent basé sur la réduction de la variance :

$$\min_{j,s} \sum_{x_i\in S\cap R_r(j,s)} (y_i - f_r)^2 + \sum_{x_i\in S\cap R_l(j,s)} (y_i - f_l)^2,$$

avec

$$f_r = \frac{1}{N_r}\sum_{x_i\in S\cap R_r(j,s)} y_i, \quad f_l = \frac{1}{N_l}\sum_{x_i\in S\cap R_l(j,s)} y_i.$$

3.4 Algorithme Récursif de Construction des Arbres

1. Définir le jeu d'apprentissage S.
2. Créer un nœud racine et y associer S.
3. Pour chaque nœud courant, déterminer la meilleure séparation (split) t en recherchant la variable x_j et le seuil s qui minimisent le critère (impureté pour la classification ou variance pour la régression).
4. Assigner la règle de séparation au nœud et diviser S en deux sous-ensembles S_r et S_l.
5. Répéter récursivement pour chaque nœud enfant jusqu'à ce que le critère d'arrêt (profondeur maximale, nombre minimal d'exemples, etc.) soit satisfait.

3.5 Critères d'Impureté (Classification)

Les mesures courantes d'impureté sont :

— **Entropie** :

$$H(S) = -\sum_{k=1}^{C} p_k(S)\log p_k(S), \quad p_k(S) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}\{y_i = k\}.$$

— **Indice de Gini** :

$$G(S) = \sum_{k=1}^{C} p_k(S)(1 - p_k(S)).$$

— **Erreur de classification** (pour la feuille) :

$$H(S) = 1 - \max_k p_k(S).$$

3.6 Arrêt et Élagage (Pruning)

Pour éviter le surapprentissage (overfitting), on peut :
— Fixer une profondeur maximale.
— Imposer un nombre minimal d'exemples par feuille.
— Utiliser une procédure d'élagage post-construction pour supprimer les branches peu significatives.

4 Méthodes d'Ensemble

Les méthodes d'ensemble combinent plusieurs modèles de base pour améliorer la performance prédictive.

4.1 Ensemble et Variance

L'idée fondamentale est de réduire la variance d'un estimateur instable (par exemple, un arbre) en moyennant plusieurs prédictions issues de modèles divers.

4.2 Bagging (Bootstrap Aggregating)

— **Principe :** Générer plusieurs jeux bootstrap $B_1, \ldots, B_T$ à partir du jeu d'apprentissage S, apprendre un modèle f_t sur chacun, et agréger les prédictions par moyenne (pour la régression) ou vote majoritaire (pour la classification).
— **Effet :** La variance du modèle agrégé est divisée par T, alors que le biais reste similaire.
— **Algorithme (Bagging)** :

1. Pour $t = 1, \ldots, T$, générer un bootstrap B_t à partir de S.
2. Apprendre un modèle f_t sur B_t.
3. Prédire avec :

$$f_{\text{bag}}(x) = \frac{1}{T}\sum_{t=1}^{T} f_t(x) \quad \text{(régression)} \quad \text{ou} \quad f_{\text{bag}}(x) = \text{majority vote}\{f_t(x)\} \quad \text{(classification)}.$$

4.3 Forêts Aléatoires (Random Forests)

Les forêts aléatoires ajoutent une étape de randomisation supplémentaire dans la sélection des variables à chaque split.

— **Procédure :**

1. Pour chaque arbre, générer un bootstrap B_t à partir de S.
2. Lors de la construction de chaque nœud, choisir au hasard un sous-ensemble de k variables parmi les p variables totales.
3. Sélectionner le meilleur split parmi ces k variables.
4. Ne pas élaguer l'arbre.

— **Avantages :** Amélioration de la performance, réduction de la corrélation entre arbres, et possibilité d'estimer l'importance des variables.

4.4 Extra-Trees

Les *Extra-Trees* (Extremely Randomized Trees) diffèrent des forêts aléatoires par le fait que le choix du seuil de split est également randomisé parmi un ensemble de candidats. Cela renforce la diversité des arbres.

4.5 Boosting

Le boosting combine itérativement des classificateurs faibles pour construire un classificateur fort.

4.5.1 Idée de Base d'AdaBoost

— À chaque itération, on entraîne un classificateur faible h_t sur les données pondérées par une distribution w_t.
— On ajuste ensuite les poids en augmentant l'importance des exemples mal classifiés.
— Le classificateur final est une combinaison linéaire :

$$H_T(x) = \sum_{t=1}^{T} \alpha_t h_t(x), \quad \text{et} \quad F_T(x) = \text{sign}(H_T(x)).$$

4.5.2 Algorithme AdaBoost (Freund et Schapire, 1996)

1. Initialiser $w_1(i) = \frac{1}{n}$ pour $i = 1, \ldots, n$.
2. Pour $t = 1, \ldots, T$:
   (a) Entraîner le classificateur faible h_t sur S pondéré par w_t.
   (b) Calculer l'erreur de classification :

$$\epsilon_t = \sum_{i=1}^{n} w_t(i) \mathbf{1}\{h_t(x_i) \neq y_i\}.$$

   (c) Calculer le coefficient :

$$\alpha_t = \frac{1}{2} \ln \frac{1 - \epsilon_t}{\epsilon_t}.$$

   (d) Mettre à jour les poids :

$$w_{t+1}(i) = \frac{w_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t},$$

   où Z_t est un facteur de normalisation.
3. La prédiction finale est :

$$F_T(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x) \right).$$

4.5.3 Analyse et Propriétés d'AdaBoost

— Les poids mis à jour encouragent la correction des erreurs.
— On peut montrer que l'erreur d'entraînement de F_T est bornée par

$$R_n(F_T) \leq \exp\left(-2 \sum_{t=1}^{T} \left(\frac{1}{2} - \epsilon_t \right)^2 \right).$$

— AdaBoost peut être vu comme une descente de gradient dans l'espace des combinaisons linéaires de classificateurs faibles.

4.6 Gradient Boosting

Le gradient boosting généralise le boosting en considérant la minimisation par descente de gradient d'une fonction de perte générale.

- **Principe :** À chaque itération, on cherche à corriger le modèle courant $H_{t-1}(x)$ en ajoutant un nouvel estimateur $h_t(x)$ qui se rapproche du gradient négatif de la fonction de perte.
- **Formulation :** Trouver (h_t, α_t) qui minimise

$$\sum_{i=1}^{n} \ell\Big(y_i, H_{t-1}(x_i) + \alpha\, h_t(x_i)\Big).$$

- Le processus se poursuit de manière additive :

$$H_t(x) = H_{t-1}(x) + \alpha_t\, h_t(x).$$

- Pour le cas de la perte exponentielle, le gradient boosting et AdaBoost deviennent équivalents.

5 Aspects Pratiques et Conclusion

5.1 Avantages et Inconvénients des Méthodes d'Ensemble

Avantages :

- Amélioration de la précision par réduction de la variance.
- Robustesse par la diversification des prédicteurs.
- Possibilité d'estimer l'importance des variables (notamment dans Random Forests).

Inconvénients :

- Complexité de calcul et difficulté d'interprétation (les modèles agrégés sont souvent des "boîtes noires").
- Risque de surapprentissage si le modèle de base est trop complexe ou mal régularisé.

5.2 Utilisations Pratiques

Les méthodes d'ensemble, notamment les forêts aléatoires et le gradient boosting, sont largement utilisées pour l'apprentissage sur des données tabulaires, et ont prouvé leur efficacité dans de nombreux défis et applications industrielles (marketing, médecine, finance, etc.).

5.3 Conclusion

Ce polycopié a permis de reprendre l'ensemble des points abordés dans la conférence :

- La modélisation par arbres de décision et de régression, avec leurs critères de séparation et leur construction récursive.
- Les principes d'ensemble (bagging, forêts aléatoires) pour réduire la variance.
- Les méthodes de boosting (AdaBoost et Gradient Boosting) pour transformer un ensemble de classificateurs faibles en un classificateur fort.

Ces méthodes constituent des outils essentiels dans l'arsenal du machine learning moderne.

6 Bibliographie

- Breiman, L. (2001). *Random Forests.* Machine Learning, 45, 5-32.

— Breiman, L., Friedman, J., Olshen, R., & Stone, C. (1984). *Classification and Regression Trees*. Wadsworth.
— Freund, Y., & Schapire, R. E. (1996). *Experiments with a New Boosting Algorithm*. In *ICML* (pp. 148-156).
— Geurts, P., Ernst, D., & Wehenkel, L. (2006). *Extremely Randomized Trees*. Machine Learning, 63, 3-42.
— Hastie, T., Tibshirani, R., & Friedman, J. (2001). *The Elements of Statistical Learning*. Springer.
— Mason, L., Baxter, J., Bartlett, P., & Frean, M. (1999). *Boosting Algorithms as Gradient Descent*. In *Advances in Neural Information Processing Systems*, 12.
— Chen, T., & Guestrin, C. (2016). *XGBoost : A Scalable Tree Boosting System*. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 785-794).