Introduction aux méthodes ensemblistes

Table of contents

Restriction du champ: méthodes ensemblistes à base d'arbres.

Lecture de base: chapitres 9-12: https://bradleyboehmke.github.io/HOML/

1 Survol des méthodes ensemblistes

Principe: cette partie propose une présentation intuitive des méthodes ensemblistes, à destination notamment des *managers* sans bagage en *machine learning*. Elle ne contient aucune formalisation mathématique.

1.1 Principe des méthodes ensemblistes

1.1.1 Pourquoi utiliser des méthodes ensemblistes?

Avantages:

- Méthodes adaptées à un grand nombre de cas d'usage de la statistique publique:
  - Elles sont notamment applicables à tous les problèmes pour lesquels on utilise une régression linéaire ou une régression logistisque);
  - Elles s'appliquent à des données tabulaires (enregistrements en lignes, variables en colonnes), situation très fréquente dans la statistique publique.
- Performances quasi systématiquement supérieures aux méthodes économétriques traditionnelles;
- Scalabilité: ces méthodes peuvent être appliquées à des données volumineuses;
- Coût d'entrée modéré (comparé à des approches plus avancées comme le *deep learning*).

Inconvénients:

- Temps d'entraînement potentiellement long, notamment pour l'optimisation des hyperparamètres.
- Ces méthodes peuvent nécessiter une puissance de calcul importante et/ou une mémoire vive de grande taille.
- Interprétabilité moindre que les méthodes économétriques traditionnelles (et encore, ça se discute)
- Risque de surapprentissage, en particulier pour le *boosting*
- La prise en main de ces méthodes requiert un temps d'apprentissage (une bonne maîtrise de Python ou R est un prérequis).

1.1.2 L'union fait la force

Plutôt que de chercher à construire d'emblée un unique modèle très complexe, les approches ensemblistes vise à obtenir un modèle très performant en combinant un grand nombre de modèles simples.

Il existe trois grandes approches ensemblistes:

- le *bagging*;
- le *stacking*;
- le *boosting*.

Le présent document se concentre sur deux approches: le *bagging* et le *boosting*.

1.1.3 Critères de performance et sélection d'un modèle

La performance d'un modèle augmente généralement avec sa complexité, jusqu'à atteindre un maximum, puis diminue. L'objectif est d'obtenir un modèle qui minimise à la fois le sous-apprentissage (biais) et le sur-apprentissage (variance). C'est ce qu'on appelle le compromis biais/variance. Cette section présente très brièvement les critères utilisés pour évaluer et comparer les performances des modèles.

1.2 Comment fonctionnent les méthodes ensemblistes?

Trois temps:

- les arbres de décision et de régression (CART);
- les forêts aléatoires;
- le boosting.

1.2.1 Le point de départ: les arbres de décision et de régression

Présenter *decision tree* et *regression tree*. Reprendre des éléments du chapitre 9 de https://bradleyboehmke.github.io/HOML/

Principes d'un arbre:

- partition de l'espace des *features*;
- fonction constante par morceaux;
- un arbre mobilise notamment les interactions entre variables.

Illustration, et représentation graphique (sous forme d'arbre et de graphique).

1.3 Le *bagging* et les *random forests*

1.3.1 Le *bagging*

Présenter le *bagging* en reprenant des éléments du chapitre 10 de https://bradleyboehmke.github.io/HOML.

- Présentation avec la figure en SVG;
- Illustration avec un cas d'usage de classification en deux dimensions.

1.3.2 Les *random forests*

Expliquer que les *random forests* sont une amélioration du *bagging*, en reprenant des éléments du chapitre 11 de https://bradleyboehmke.github.io/HOML/

- Présentation avec la figure en SVG;
- Difficile d'illustrer avec un exemple (car on ne peut pas vraiment représenter le *feature sampling*);
- Bien insister sur les avantages des RF: 1/ faible nombre d'hyperparamètres; 2/ faible sensibilité aux hyperparamètres; 3/ limite intrinsèque à l'overfitting.

1.4 Le *boosting*

Reprendre des éléments du chapitre 12 de https://bradleyboehmke.github.io/HOML/ et des éléments de la formation boosting.

Le *boosting* combine l'**approche ensembliste** avec une **modélisation additive par étapes** (*forward stagewise additive modeling*).

- Présentation du principe: entraîner séquentiellement des modèles simples et peu performants (*weak learners*) pour obtenir un modèle complexe très performant (*strong learner*);
- Avantage du boosting: performances particulièrement élevées.
- Inconvénients: 1/ nombre élevé d'hyperparamètres; 2/ sensibilité des performances aux hyperparamètres; 3/ risque élevé d'overfitting.
- Préciser qu'il est possible d'utiliser du subsampling par lignes et colonnes pour un algoithme de boosting. Ce point est abordé plus en détail dans la partie sur les hyperparamètres.

1.4.1 Les différences entre *random forests* et *boosting*

Les forêts aléatoires et le *gradient boosting* paraissent très similaires au premier abord: il s'agit de deux approches ensemblistes, qui construisent des modèles très prédictifs performants en combinant un grand nombre d'arbres de décision. Mais en réalité, ces deux approches présentent plusieurs différences fondamentales:

- Les deux approches reposent sur des fondements théoriques différents: la loi des grands nombres pour les forêts aléatoires, la théorie de l'apprentissage statistique pour le *boosting*.
- Les arbres n'ont pas le même statut dans les deux approches. Dans une forêt aléatoire, les arbres sont entraînés indépendamment les uns des autres et constituent chacun un modèle à part entière, qui peut être utilisé, représenté et interprété isolément. Dans un modèle de *boosting*, les arbres sont entraînés séquentiellement, ce qui implique que chaque arbre n'est ni utilisable, ni interprétable indépendamment de l'ensemble des arbres qui l'ont précédé dans l'entraînement.
- *overfitting*: borne théorique à l'*overfitting* dans les RF, contre pas de borne dans le *boosting*. Deux conséquences: 1/ lutter contre l'overfitting est essentiel dans l'usage du *boosting*; 2/ le *boosting* est plus sensible au bruit et aux erreurs sur y que la RF.
- Les points d'attention dans l'entraînement ne sont pas les mêmes: arbitrage puissance-corrélation dans la RF, arbitrage puissance prédictive-overfitting dans le *boosting*.

- Conditions d'utilisation: la RF peut être utilisée sur le *train* grâce aux prédictions *out-of-bag*, pas le *boosting*. Exemple: repondération d'une enquête.
- Complexité d'usage: peu d'hyperparamètres dans les RF, contre un grand nombre dans le *boosting*.

1.4.2 Quel algorithme utiliser?

Comment choisir entre forêt aléatoire et boosting:

- Temps dont on dispose: RF si peu de temps;
- Puissance de calcul dont on dispose: RF si peu de puissance;
- Compréhension des algorithmes: RF si on est débutant;
- Nombre de *features*: RF si nombreuses;
- Nature du problème: y a-t-il des spécificités locales (au sens mathématique) que même un arbre assez profond aura du mal à prendre en compte? Si oui, le *boosting* est indiqué.
- Y a-t-il beaucoup de bruit, de valeurs aberrantes ou d'erreurs dans les données: si oui, la RF est préférable.

2 Présentation formalisée des méthodes ensemblistes

2.1 Rappels sur l'apprentissage supervisé

Veut-on faire quelque rappels sur l'apprentissage supervisé?

2.2 Les arbres de décision et de classification

2.2.1 La brique élémentaire: l'arbre de décision

2.2.2 L'algorithme CART, un partitionnement binaire récursif

Description des CART.

2.3 Le bagging et les forêts aléatoires

2.3.1 Le *bagging*

2.3.2 Les forêts aléatoires

2.4 Le *gradient boosting*

Cette section détaille la mécanique du *gradient boosting* en reprenant les notations de l'article décrivant XGBoost (2016). Cette partie décrit rapidement: le cadre théorique *weak learner* et *strong learner* (R. Shapire [1]), le principe de la modélisation additive par étapes (J. H. Friedman [2]), l'approximation faite par XGBoost, la méthode de calcul des poids optimaux, la méthode de recherche des *splits* optimaux, les fonctions de perte couramment utilisées.

Cette partie mentionnera rapidement les poinds suivants:

- il existe des implémentations du *boosting* qui ne sont pas du *gradient boosting* (exemple: l'*adaptative boosting* de l'algorithme AdaBoost).

- Il existe de multiples implémentations du *gradient boosting* (GBM, lightGBM, XGBoost, Catboost…), globalement similaires mais qui diffèrent sur des points de détail. La présentation qui suit doit donc être complétée par la lecture de la documentation des différents algorithmes.
- Cette approche permet de construire des modèles de *boosting*, mais aussi des forêts aléatoires entraînées par descente de gradient.

3 Comment (bien) utiliser les approches ensemblistes

3.1 Préparer les données

3.1.1 La *target*

- Penser aux transformations préalables (log, ratio…).
- Quid des variables catégorielles ordonnées?

3.1.2 Les *features*

- Que faire des variables continues?
  - les transformations monotones sont inutiles;
  - les transformations non monotones peuvent être utiles;
  - attention aux paramètres de la *quantization* par histogramme;
- La gestion des variables catégorielles:
  - Il est possible de passer les variables catégorielles ordonnées en integer.
  - Pour les variables catégorielles non ordonnées:
    - Réduire le nombre de modalités?
    - utiliser le one hot encoding ou le support expérimental des variables catégorielles (split selon la méthode de W. D. Fisher [3])

3.2 Comment entraîner un algorithme

3.2.1 Entraîner une *random forest*

3.2.1.1 Rôle et interprétation des principaux hyperparamètres

Faire systématiquement le renvoi vers la partie formalisée, pour que les lecteurs sachent où intervient chaque hyperparamètre.

3.2.1.2 Guide d'entraînement d'une forêt aléatoire

Reprendre les recommandations de P. Probst, M. N. Wright, and A.-L. Boulesteix [4].

3.2.2 Entraîner un algorithme de *boosting*

3.2.2.1 Rôle et interprétation des principaux hyperparamètres

Faire systématiquement le renvoi vers la partie formalisée, pour que les lecteurs sachent où intervient chaque hyperparamètre.

3.2.2.2 Guide d'entraînement d'un algorithme de *boosting*

Reprendre les recommandations de C. Bentéjac, A. Csörgő, and G. Martínez-Muñoz [5]. Que dit HOML là-dessus?

3.3 Usage avancé

3.3.1 Choisir une fonction de perte non standard

Exemple: Huber?

3.3.2 L'utilisation des pondérations

Cette partie présente l'usage des pondérations des observations (`sample_weight`) et de la pondération de la classe positive (`scale_pos_weight`).

Bien expliquer où ces pondérations interviennent dans la partie formalisée.

3.4 Interprétabilité des méthodes

- Mesure d'importance: intérêt et limites.
- Quels frameworks veut-on présenter?
  - Interprétabilité globale;
  - Interprétabilité locale.

4 Cas d'usage

- Données (pouvant être rendues) publiques
- Notebooks déployables sur le datalab
- Code en Python

4.1 Régression

4.1.1 Cas général

4.1.2 Régression en présence d'outliers

=> Changement de fonction de perte

4.2 Classification

4.2.1 Cas général

4.2.2 Classification déséquilibrée

=> Utiliser la pondération de la classe minoritaire

Bibliography

[1] R. Shapire, "The strength of weak learning," *Machine Learning*, vol. 5, no. 2, 1990.

[2] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of statistics*, pp. 1189–1232, 2001.

[3] W. D. Fisher, "On grouping for maximum homogeneity," *Journal of the American statistical Association*, vol. 53, no. 284, pp. 789–798, 1958.

[4] P. Probst, M. N. Wright, and A.-L. Boulesteix, "Hyperparameters and tuning strategies for random forest," *Wiley Interdisciplinary Reviews: data mining and knowledge discovery*, vol. 9, no. 3, p. e1301, 2019.

[5] C. Bentéjac, A. Csörgő, and G. Martínez-Muñoz, "A comparative analysis of gradient boosting algorithms," *Artificial Intelligence Review*, vol. 54, pp. 1937–1967, 2021.