Méthodes ensemblistes

Lorsqu'on travaille avec de grands jeux de données, il est souvent plus efficace de combiner les prédictions de plusieurs modèles simples, appelés **classificateurs faibles**, que d'essayer de construire un unique modèle très complexe. Cette idée est à la base des **méthodes d'ensemble**, qui visent à produire une prédiction unique à partir des prédictions de plusieurs modèles. En regroupant ainsi plusieurs modèles, on peut généralement obtenir une meilleure performance globale et souvent une meilleure robustesse par rapport au bruit et au sur-ajustement.

Plusieurs stratégies peuvent être employées pour combiner les résultats de plusieurs modèles. Une première approche consiste à faire la **moyenne des probabilités prédites** par chaque modèle. Une autre méthode courante est le **vote de majorité**, où la classe la plus souvent prédite est retenue. On peut également adopter une stratégie inspirée de la sélection de variables utilisée dans le modèle de régression. Au lieu de choisir les "meilleures" variables, on sélectionne les modèles les plus pertinents pour l'ensemble. Enfin, des techniques plus avancées comme le ***bagging***, le ***boosting*** et les **forêts aléatoires** ont été développées pour automatiser et optimiser cette combinaison de modèles.

1 Bagging

La méthode du ***bagging*** (*bootstrap aggregating*) consiste à construire plusieurs versions d'un même modèle à partir d'échantillons différents du jeu de données d'origine. Plus précisement, on commence par générer B échantillons *bootstrap* (tirages aléatoires avec remise) à partir des données d'apprentissage. Un arbre de classification est ensuite construit pour chacun de ces B échantillons. On obtient donc B arbres de classification. On prédit la classe d'une nouvelle observation pour chacun des B arbres et on retient la classe prédite le plus souvent, ce qui correspond à un vote de majorité. Il est également possible de calculer la moyenne des probabilités de classe fournies par chaque arbre pour obtenir un score plus nuancé.

Le principal avantage du *bagging* est qu'il permet de réduire la variance et d'augmenter la stabilité des prédictions. Cependant, cette méthode rend plus difficile l'interprétation du rôle joué par chaque variable dans la classification. Pour évaluer l'importance d'une variable, on peut additionner les réductions de l'indice de Gini observées à chaque division basée sur cette variable, puis faire la moyenne de ces réductions sur tous les arbres. On exprime enfin ces

moyennes sous forme de pourcentage relatif à la réduction maximale observée parmi toutes les variables.

2 Forêts aléatoires

Les **forêts aléatoires** reprennent le principe du *bagging*, mais introduisent une source supplémentaire d'aléa qui permet de décorréler les arbres construits. En effet, les arbres construits par l'algorithme de *bagging* tendent à se ressembler car les classes les plus fréquentes dominent. Pour résoudre ce problème, la méthode des forêts aléatoires impose qu'à chaque division dans un arbre, on sélectionne aléatoirement un sous-ensemble de m variables parmi les p disponibles et que seules ces m variables soient considérées pour choisir la meilleure division. Un choix usuel est $m \approx \sqrt{p}$. Cette diversification améliore la performance globale de l'algorithme, ainsi que sa robustesse.

3 Boosting

À la différence du *bagging*, le ***boosting*** adopte une approche séquentielle. L'idée est de construire une série de modèles simples en accordant progressivement plus d'importance aux observations mal classées par les modèles précédents. Chaque nouveau modèle est entraîné pour corriger les erreurs du précédent, ce qui permet d'atteindre une très bonne précision globale. Pour fonctionner efficacement, le *boosting* doit utiliser des modèles faibles, laissant au processus le soin de combiner leurs prédictions de manière optimale.

Il existe plusieurs variantes du *boosting*. L'une des plus connues est *AdaBoost* (*boosting adaptatif*), qui ajuste les poids des observations à chaque itération. Une autre approche très utilisée aujourd'hui est le *gradient boosting*, qui repose sur la minimisation d'une fonction de perte à l'aide d'une procédure itérative inspirée de la descente du gradient.

4 Optimisation des hyper-paramètres

Comme toute méthode de classification, les modèles d'ensembles nécessitent une optimisation de leur hyper-paramètres pour atteindre un bon compromis entre performance et robustesse. On observe d'ailleurs empiriquement que plus une méthode est puissante, plus l'optimisation des hyper-paramètres est complexe et sensible.

Voici quelques exemples de paramètres à ajuster selon les méthodes :

- *Bagging* : nombre d'échantillons *bootstrap* B ;
- Forêts aléatoires : nombre d'échantillons *bootstrap* B et nombre de variables m à considérer à chaque division ;

— *Boosting* : taille des arbres, fraction du jeu de données à échantillonner à chaque étape, nombre d'itérations de l'algorithme, …

Pour optimiser ces hyper-paramètres et éviter le sur-ajustement, une validation croisée pour chacun des paramètres est généralement nécessaire.