Arbres

Les arbres de régression et de classification (Classification And Regression Trres, CART) font partie des méthodes d'apprentissage supervisée. On dispose d'observations qui appartiennent à $\mathbb{R}^p$, soit $X_1, \dots, X_p$. Un arbre de classification cherche à assigner une observation à l'un des K groupes sur la base de $X_1, \dots, X_p$. Un arbre de régression cherche à prédire la valeur d'une valeur numérique continue Y à partir de $X_1, \dots, X_p$.

La construction d'arbres, on partitionne l'ensemble R^p en hyper-rectangles en effectuant des divisions binaires successives en fonction de la valeur d'une des p variables (algorithme de dédoublements binaires successifs).

1. On effectue des divisions binaires successives de l'ensemble R^p en fonction de la valeur d'une variable. Le choix de la variable et le choix de la limite pour la division binaire sont déterminés par un certain critère.
2. On cesse d'effectuer des divisions lorsqu'un critère d'arrêt est rencontré (nombre minimal d'observations par feuille atteint, nombre de feuilles désirées atteint, etc.)
3. Chaque sous-division finale de l'ensemble $\mathbb{R}^p$est appelée "feuille" ou "noeud terminal". On prédit une catégorie à chaque observation selon la feuille à laquelle elle appartient.

Toutes les observations dans une même "feuille" reçoivent la même prédiction, soit la classe la plus fréquente parmi les données du jeu d'entrainement qui aboutissent dans cette feuille.

> Remarque
>
> L'algorithme de partitions binaires successives est un algortihme glouton qui ne nous garantit pas l'arbre optimal…
> Supposons que nous désirions partionner $\mathbb{R}^p$ en J hyper-rectangles (donc un arbre à J feuilles) et soit c_j un critère de "coût" pour les observations qui sont mal classées dans la région j. On aimerait pouvoir trouver les hyper-rectangles $R_1, \dots, R_J$ tels que $\sum_{j=1}^{J} c_j$ soit minimal, mais il n'est pas computationnellement possible de le faire !

Algortihme

Pour chaque noeud de l'arbre, on doit décider si on le découpe en deux et si oui, en utilisant quelle valeur de quelle variable :

1. Pour chacune des p variables et pour chaque point de coupure possible pour cette variable, calculer l'amélioration du critère de qualité et choisir la variable et le point de coupure résultant en la meilleur amélioration du critère.
2. Créer une division binaire de l'arbre à l'aide de la variable et du point de coupure choisis en 1.
3. Répéter 1 et 2 jusqu'à ce que l'arbre ait le nombre de feuilles désiré ou que le nombre minimal d'observations soit atteint dans chaque feuille.

Critères

Soit $\hat{p}_{jk}$, la proporiton d'observations de la je région de l'ensemble $\mathbb{R}^p$ qui appartiennent à la ke classe.

Trois critères sont habituellement utilisés pour déterminer les divisions optimales à chaque étape de la construction de l'arbre :

1. Taux d'erreurs de classification
$$E_j = 1 - \max_k \hat{p}_{jk}.$$
Pas assez sensible pour construire l'arbre.
2. Indice de Gini
$$G_j = \sum_{k=1}^{K} \hat{p}_{jk}(1 - \hat{p}_{ij}).$$
Plus c'est faible, mieux c'est.
3. Entropie croisée
$$D_j = -\sum_{k=1}^{K} \hat{p}_{jk} \log(\hat{p}_{mk}).$$
Plus c'est faible, mieux c'est.

Le gain d'information est ce que l'on cherche à maximiser lorque l'on choisit comment scinder en deux un noeud donné. Par example, avec l'indice de Gini, le gain d'information est donné par
$$\text{indice de Gini avant la division} - \sum_{j=1}^{2} \frac{n_j}{n_1 + n_2} G_j,$$
où $j = 1, 2$ dénote les deux morceaux créés par la division, n_j est le nombre d'observations qui tombent dans le morceau j et G_j est l'indice de Gini pour le morceau j.

Choix du nombre de divisions/feuilles

L'algorithme va faire croître l'arbre tant qu'on n'aura pas atteint un nombre de feuilles fixé ou tant que le nombre de d'observations dans chaque feuille ne sera pas trop petit. Mais, pas assez de feuilles veut dire manque d'ajustement car classification trop simpliste et inadéquate. Et trop de feuilles n'est pas mieux, car on fait du sur-ajustement et l'arbre n'aura pas une bonne performance sur un jeu de validation.

Élage des arbres de classification (pruning) : la stratégie commune en pratique est de laisser croître l'arbre et ensuite de l'élaguer de façon à atteindre un compromis entre ajustement aux données d'entrainement et taille de l'arbre (élagage coût-complexité). Si $|T|$ est la taille (nombre de feuilles) de l'arbre T, on veut

$$\min \sum_{j=1}^{|T|} c_j + \alpha |T|,$$

où $\alpha \geq 0$ est un hyper-paramètre qui contrôle le compromis et que l'on chosit par validation croisée.

Avantages

- Fonctionnent aussi bien avec des variables continues que catégorielles.
- Aucune hypothèse a priori n'est faite sur la distribution des données.
- Robuste aux données extrêmes.
- Faciles à interpréter.
- Tiennent implicitement compte des interactions possibles entre les variables.
- Sélectionnent implicitement les variables importantes.
- Permettent d'obtenir un modèle non linéaire.

Points faibles

- Nécessitent un grand nombre de données.
- Peuvent être instables dans les résultats qu'elle produit.
- Ne fonctionnent pas très bien lorsque certaines des q classes sont rares ou que certaines des modalités de X_j catégorielles sont rares.

Example