STT3030 - Cours #6

Arthur Charpentier

Automne 2024

Rappel

- On veut faire de la prédiction: $\hat{Y} = \hat{f}(\boldsymbol{X})$
- Étant donnée des prédicteurs $\boldsymbol{X}$ quelle est notre meilleur estimation de la réponse.

Nous avons un ensemble de données

$$D_n = \{(\boldsymbol{x}_i, y_i) \text{ pour } i \in (1, ..., n)\} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), ..., \boldsymbol{x}_n, y_n)\},$$

avec des observations dans $\mathcal{X} \times \mathcal{Y} = \mathcal{X}_i \times ... \times \mathcal{X}_p \times \mathcal{Y}$.
C'est grâce à ces donnés que nous allons entrainer notre modèle, apprendre a imiter f.
Arbres de décision prennent la forme:

$$\hat{Y} = \begin{cases} c_1 & \text{si } \mathbf{x} \in R_1 \\ c_2 & \text{si } \mathbf{x} \in R_2 \\ ... & \\ c_m & \text{si } \mathbf{x} \in R_m \end{cases} \tag{1}$$

Rappel

Pour former un arbre $\hat{f}$ à l'aide d'un échantillon D, il nous faut **quatre** choses selon, Friedman et al. (2001).

1. Une collection de partitionnement binaire.
2. Une évaluation de la qualité du partitionnement.
3. Une règle d'arrêt.
4. Une règle étiquettage.

Partionnement binaire: on divise (itérativement) nos données en **deux groupes**.
Pour chaque prédicteurs X_j on regarde chacune des manière de diviser les donnés $(\boldsymbol{x}_i, y_i),\ i \in D_{ent}$ - ou le sous-échantillon $\boldsymbol{x}_i \in$ groupe.

- Si X_j est continue, les partionnement sont de la forme $X_j \leq t$ contre $X_j > t$ et nous allons tester toute les division possible dans l'échantillon, on regarde donc les $n-1$ possible partionnement.
- Si X_j est catégorielle avec c classes, nous les ordonnons, et regardons les $c-1$ regroupements

Rappel

Soit Q_j, une mesure d'hétérogénéité de la région j (**impurity measure**)

- Si y est continue, on évalue l'erreur quadratique: $Q_j = \sum_{i \in R_j} (y_i - \frac{1}{n} \sum_{i \in R_j} y_i)^2$.
- Si y est catégorielle, on utilise d'autres mesures de "mixitude," tel que l'indice de Gini: $Q_j = \sum_{k=1}^{c} \hat{p_{jc}}(1 - \hat{p_{jc}})$ où $\hat{p_{jc}}$ est la proportion de la classe c dans la région j (i.e., $\hat{p_{jc}} = \frac{1}{n_j} \sum_{i \in R_j} \mathbf{1}_{y_i = c}$).

On choisir le partitionnement qui minimise l'hétérogénéité.
Finalement, pour chaque partionnement, on évalue l'hétérogénéité moyenne des 2 régions résultantes. Si l'on cherche à diviser R_p en deux nouvelle région R_t et R_r on regarde:

$$n_r Q_r + n_t Q_t \tag{2}$$

Rappel

Les règles d'arrêt typique sont:

- (1) Lorsque les régions sont homogènes (si y est catégoriel)

et/ou

- (2) Jusqu'à ce qu'il y ai moins de β observations.

- Si la réponse y est continue alors, $c_j = \frac{1}{n_j}\sum_{i \in R_j} y_i$ est déterminé, la moyenne des observations y_i de l'échantillon dans la région R_j.
- Si la réponse y est catégoriel, $c_j = \underset{c}{\text{argmax}} \hat{p_{jc}}$ soit la classe majoritaire dans la région R_j.

On dit que la formation d'arbre de décision est une procédure instable et prompt au surrapprentissage.

Bien que l'émmondage/élagage d'arbre résout partiellement le problème de surraprentissage, cette technique n'addresse pas l'instabilité.

Introduction

Un problème majeur des arbres de décision est relié à sa procédure de formation.

On prend toujours le meilleur partitionnement à l'instant. Donc un petit changement dans D_{ent} peut changer un partitionnement à l'instant et donc changer tout le reste de l'arbre.

Quand un petit changement dans D_{ent} mène a un grand changement dans $\hat{f}_{D_{ent}}$, alors on dit que la procédure est **instable** (plus de détail dans le cours gradué).

Instabilité des arbres de décisions

En changeant seulement quelques points dans S_{ent} on peut donc créer des arbres complètement différents.
C'est quoi le problème ?
Comment choisir le meilleur ?
Si on aime les arbres parce qu'ils sont interpretable mais que la procédure est instable peut-on se fier à l'interpretation ?
On choisit celui qui performe mieux ? (En ML on aime bien la précision de prédiction)
On peut faire mieux.
On doit prendre l'interpretabilité des arbres de décision avec un grain de sel: On peut interpreter le modèle appris $\hat{f}$ et donc comprendre comment ce dernier prédit, mais on ne fait PAS de l'inférence, on ne présuppose pas que la forme apprise $\hat{f}$ représente la réalite d'aucune manière.

Instabilité des arbres de décisions

Dans la formation de forêts aléatoires,

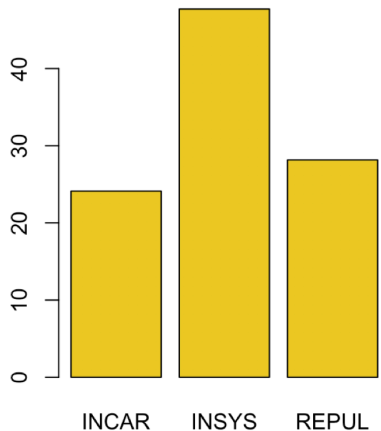
- On met à profit l'instabilité des arbres de décision pour capture différents aspects des données (variabilité/flexbilité).
- On réduit la variabilité des modèles individuelles en le regroupant (penser à la moyenne de plusieurs points).
- On perd l'interpretation mais gagne beaucoup en performance (diminution de la variance sans augmenter le biais!).

Instabilité des arbres

```
> variable=rep(NA,10000)
> for(i in 1:10000){
+ arbre = rpart(PRONO~., data=
    myocarde[sample(1:71,size=47),])
+ variable[i] = as.character(
    arbre$frame[1,"var"])
+ }
> table(variable)/100
INCAR INSYS REPUL
24.12 47.72 28.16
```

la première variable utilisée au premier noeud est

- INSYS (47.7%)
- REPUL (28.2%)
- INCAR (24.1%)



Méthodes d'ensemble

Le concept des méthodes d'ensemble (méthodes ensemblistes) est l'entrainement (en parralèle) d'une large collection (un ensemble) de fonction de prédiciton ($\hat{f}$).

Une fois que nous avons appris une collection de fonction (un ensemble), nous allons toutes les considérer lorsque nous allons prédire la réponse d'une nouvelle observation.

Brièvement, ça marche parce que ça réduit la variabilité du modèle résultant. On prend en quelque sorte la moyenne de notre collection de modèle, ce qui réduit grandement la variabilité (l'instabilité) de l'ensemble comparativement aux fonctions individuelle qui le forme.

Nous savons aussi que de réduire la variabilité/flexibilité nous protège du surraprentissage.

Il existe plusieurs méthodes pour générer notre collection de modèles $\hat{f}$ et plusieurs méthodes pour combiner les prédictions de ceux-ci, nous en parlerons rapidement.

Bootstrap

Dans cette courte séance, nous introduirons le **bootstrap aggregating**, communément appelé **bagging**, appliqué aux arbres de décisions, nous appelerons ça **bagged trees**, une forme simple de forêt aléatoire.

Soit D^* les données reçu (après avoir retiré les données test). Nous formons donc $D_B = \{D_1^b, D_2^b, ..., D_B^b\}$ un ensemble de B échantillons bootstrap. Il peut s'agir d'échantillon avec ou sans remise et taille variable.

Nous allons ensuite estimer $\hat{f}$ sur chacun des échantillons boostraps. Ici, appliqué aux arbres cela veut dire entrainer un arbre sur chaque échantillon. Comme les arbres sont instables nous aurons une forêt d'arbres très différents.

Le bagging forme une collection de B fonctions de prédiction $\hat{f}$, disons $\hat{f}_b = \{\hat{f}_1, \hat{f}_2, ...\hat{f}_B\}$, où $\hat{f}_j$ est l'arbre formé l'échantillon bootstrap D^{b_j}.

Aggregating

Maintenant parlons **aggregation**, la mise en commun de nos fonctions $\hat{f}$. Encore ici, il y existe plusieurs manière de joindre la prédiction de nos B fonctions $\hat{f}$, nous introduirons la version la plus simple de bagging pour commencer.

Soit $\mathcal{Y} = \mathbb{R}$, donc la réponse est une variable continue. Comme $\hat{f} : \mathcal{X} \to \mathcal{Y}$ $\forall i$ alors chaque fonction retourne une valeur continue et une technique proposé pour mettre en commun la prédiction de chaque fonctions est d'en prendre la moyenne.

Soit $\hat{f}_{\text{bag}}$ la fonction de prédiction produite par bagging, si $\mathcal{Y} = \mathbb{R}$ alors:

$$\hat{f}_{\text{bag}}(\boldsymbol{x}_i) = \frac{1}{B}\sum_{j=1}^{B} \hat{f}_j(\boldsymbol{x}_i) \tag{3}$$

Soit $\mathcal{Y} = \{1, .., k\}$, donc la réponse est une variable catégoriel (k classes). Comme $\hat{f}_i : \mathcal{X} \to \mathcal{Y}$ $\forall i$ alors chaque fonction retourne l'une des k catégorie. Conséquemment une technique proposé pour mettre en commun la prédiction de chaque fonctions est de percevoir chaque prédiction comme un *vote* pour une classe.

Bagging

Soit $\hat{f}_i$ la fonction de classification produite sur un échantillon, alors on peut représenter celle-ci comme un vecteur de taille K où chaque élément est 0 à l'exception de la prédiction.

En d'autres mot si $\hat{f}_i(\boldsymbol{x}_i) = j$ alors on peut réécrire $\hat{f}_i(\boldsymbol{x}_i) = [\underset{1}{0}, \underset{2}{0}, ...\underset{j}{1}, ..., \underset{k}{0}]$. Ensuite, on prend la somme de ces vecteur pour tout échantillon bootstrap, et on retourne l'élément avec le plus de vote.

$$\hat{f}_{\text{bag}}(\boldsymbol{x}_i) = \underset{k}{\text{argmax}} \sum_{j=1}^{B} \hat{f}_j(\boldsymbol{x}_i) \tag{4}$$

En résumé, nous formons une collection de B échantillons bootstrap. Par la suite nous déterminons B fonctions de prédictions, une fonction sur chaque échantillon. Finalement, la prédiction par aggrégat met à contribution chaque fonctions.

```
> library(randomForest)
> ?randomForest
```

Pourquoi ça fonctionne ?

L'espérance d'une fonction est le même que celui de l'ensemble:

$$\mathbb{E}_D[\hat{f}_{\text{bag}}(\boldsymbol{x})] = \frac{1}{B}\sum_{j=1}^{B}\mathbb{E}_D[\hat{f}_j(\boldsymbol{x})] = \frac{1}{B}B\cdot\mathbb{E}_D[\hat{f}_j(\boldsymbol{x})] = \mathbb{E}_D[\hat{f}_j(\boldsymbol{x})]$$

Pourquoi ça fonctionne ?

Alors que la variance s'écrit

$$\begin{aligned}\mathrm{Var}(\widehat{f}_{\mathrm{bag}}(\boldsymbol{x})) &= \mathrm{Var}\left(\frac{1}{B}\sum_{b=1}^{B}\widehat{f}_b(\boldsymbol{x})\right)\\ &= \frac{1}{B^2}\mathrm{Var}\left(\sum_{b=1}^{B}\widehat{f}_b(\boldsymbol{x})\right)\\ &= \frac{1}{B^2}\sum_{b_1=1}^{B}\sum_{b_2=1}^{B}\mathrm{Cov}(\widehat{f}_{b_1}(\boldsymbol{x}),\widehat{f}_{b_2}(\boldsymbol{x}))\\ &\leq \frac{1}{B^2}B^2\mathrm{Var}(\widehat{f}_b(\boldsymbol{x})) = \mathrm{Var}(\widehat{f}_b(\boldsymbol{x}))\end{aligned}$$

puisque lorsque $b_1 \neq b_2$, on a $\mathrm{Corr}[\widehat{f}_{b_1}(\boldsymbol{x}),\widehat{f}_{b_2}(\boldsymbol{x})] \leq 1$.

Pourquoi ça fonctionne ?

En fait, si $\text{Var}(\widehat{f}_b(\boldsymbol{x})) = \sigma^2(\boldsymbol{x})$ et $\text{Corr}[\widehat{f}_{b_1}(\boldsymbol{x}), \widehat{f}_{b_2}(\boldsymbol{x})] = r(\boldsymbol{x})$,

$$\text{Var}(\widehat{f}_{\text{bag}}(\boldsymbol{x})) = r(\boldsymbol{x})\sigma^2(\boldsymbol{x}) + \frac{1 - r(\boldsymbol{x})}{B}\sigma^2(\boldsymbol{x})$$

La variable sera d'autant plus faible que l'on aggrège des modèles différents.

L'instabilité des arbres en fait de bons candidats pour de l'aggrégation

Le gain est donc dans la variance du prédicteur appris.

Pourquoi ça fonctionne ?

Voici une autre perspective, si le modèle est stable, alors les fonctions $\hat{f}_i$ sont similaires par la stabilité et donc les prédiction $\hat{f}_i(\boldsymbol{x})$ seront aussi très similaire. En prendre la moyenne changera très peu le résultat.

Par contre, si le modèle est très variables, alors les fonctions $\hat{f}_i$ sont différentes et donc les prédictions $\hat{f}_i(\boldsymbol{x})$ seront aussi différentes.

En d'autre mots, le bagging profite plus aux techniques instables comme les arbres de décision ou les réseaux de neuronnes.

Le bagging met à profit les modèles instables; noous capturons différents aspects des données.

Bagged trees

Voici une procédure simple pour former un forêt aléatoire de type **bagged trees**:

1. Diviser D en D^* et D_{test}.
2. Échantillonner B échantillons bootstraps à partir de D^*.
3. Entrainer une arbre de décision $\hat{f}$ sur chaque échantillon D_j^b: $\hat{f}_j$.

On peut ensuite prédire à l'aide de cette forêt aléatoire de **bagged trees**.

Forêts aléatoires

Le bagging profite plus aux techniques instables comme les arbres de décision ou les réseaux de neuronnes.

Dans la version la plus célèbre des forêts aléatoire, on pousse la donne pour crée des arbres encore plus différents.

Pour se faire, lors de la création des arbres, nous allons échantillonner au hasard un nombre de prédicteurs, disons m parmi lesquelles nous sélectionnerons le meilleur partitionnement.

Forêts aléatoires

Supposons $m = \sqrt{p}$, à chaque étape de la formation de l'arbre, nous analysons seulement un sous-ensemble des p prédicteurs.

Conséquemment, les arbres sont plus rapides à former.

Mais surtout, ils sont encore plus différents et capture tous différents aspects des données.

Forêts aléatoires vs *bagged trees*

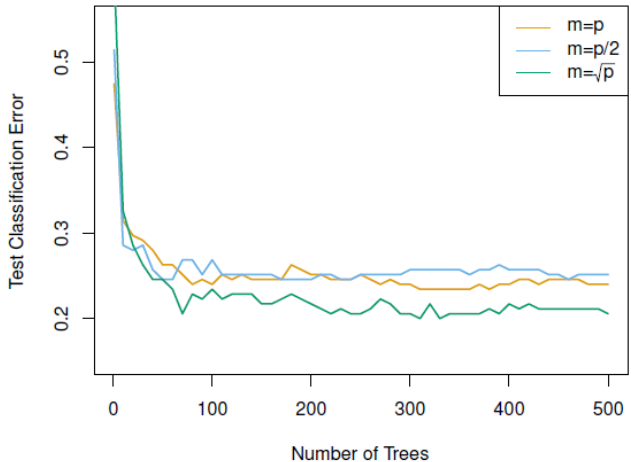


Figure 1: Figure 8.10 du manuel James et al. (2013).

Forêts aléatoires vs *bagged trees*

- On va apeller ici tout collection d'arbres de décision une forêt aléatoire.
- Il y a plusieurs sorte de forêts et de manière de former les arbres.
- Plusieurs manière de former les échantillons bootstrap (avec ou sans remise)
- Il n'y a pas de clair gagnant, mais l'idée reste la même, on veut mettre en commun une collection de modèles différents.
- Intuitivement, pourquoi pas faire ça avec les modèles linéaires ?

OOB: out-of-bag

- Lorsqu'on entre en contact avec les forêt aléatoires, on parle souvent de données OOB.
- Chaque arbres $\hat{f}_i$ est formé sur sont échantillon D_i^b et donc il y existe une collection de points qui n'ont pas contribué à l'arbre soit $D^* \setminus D_i^b$
- L'idée est donc de ne pas diviser D en deux groupes et d'utilisé les points OOB pour remplacer D_{test} (ou bien validation si nécessaire).

Voir Biau et al. (2008); Zhang and Ma (2012); Biau and Scornet (2016), pour aller plus loin, et Genuer and Poggi (2020) pour la mise en oeuvre avec R (https://rfwithr.robin.genuer.fr/).

Importance des variables

- En supposant que nous sommes convaincu que les forêts aléatoires sont une bonne alternatives aux modèles linéaires. Que faire de l'inférence et comment déterminer quel prédicteurs sont utiles ?
- Les forêts aléatoires sont re-connus comme possédant d'intéressants calculs pour évaluer la contribution des prédicteurs: on apelle ça une étude de l'importance des variables.

Importance des variables: diminution de l'erreur (GINI ou EQM)

- La méthode la plus simple est de calculer la réduction d'hétérogénéité (GINI ou EQM) encouru lorsqu'une variable est sélectionné,
- de cumulé cette réduction sur chaque arbre
- et d'en prendre la moyenne à travers la forêt.

Importance des variables: diminution de l'erreur (GINI ou EQM)

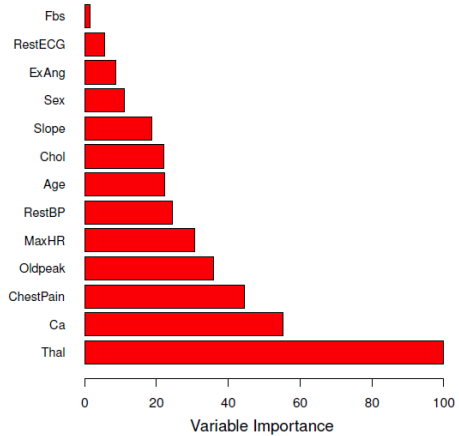


Figure 2: Figure 8.9 du manuel James et al. (2013).

Importance des variables: indice de permutation

Par contre, une technique ingénieuse qui peut être utilisé pour tout modèle et qui a fait ses preuve est l'indice de permutation:

Supposons que X_j est un prédicteur très utile dans la prédiction de Y. X_j est fortement corrélé à Y; bref X_j nous en dit beaucoup sur Y.

Si nous prennons un échantillon D_{test} et que l'on embrouille X_j il sera donc beaucoup plus difficile de prédire la réponse pour ces données.

Importance des variables: indice de permutation

Nous allons tout d'abord évalue la précision sur D_{test}.

Puis nous allons embrouiller les valeurs de X_j en permutant à travers les n_{test} observations de D_{test}.

Pour chaque jeu de donnés embrouillé D^j_{test} nous allons calculer la précision (classification ou régression).

On considère qu'une variable est plus importante qu'un autre si la précision baisse plus lorsqu'on l'embrouille.

Importance des variables: indice de permutation

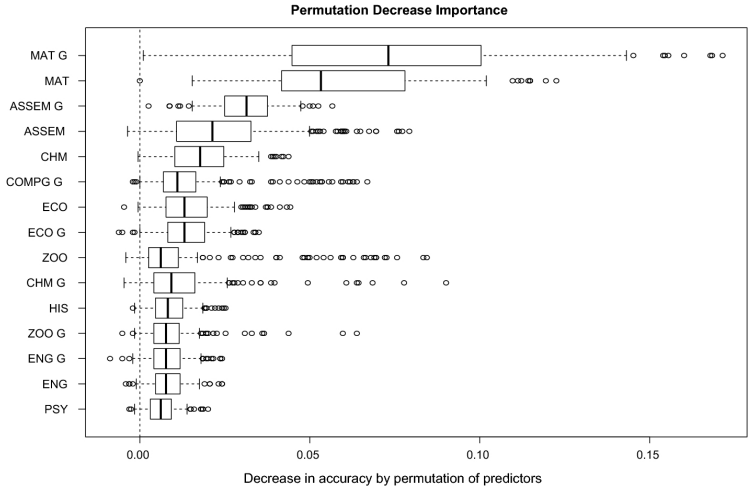


Figure 3: Figure extraite des notes de cours de Cédric Beaulac.

Importance des variables: indice de permutation

L'indice de permutation fonctionne très bien en pratique et peut être utilisé sur les modèles linéaires, réseau de neuronnes etc...

Représente la perte de pouvoir prédictif du modèle lorsqu'une variables est mélangé.

On peut refaire la permutation un grand nombre de fois pour formé des intervales de confiance.

Par contre, cette indice est biaisé si les échantillons D^b sont avec remise!

Forêts aléatoires: en pratique

- La forêt aléatoire est plus simple a utiliser en pratique que l'arbre de décision.
- Moins de paramètres à ajuster (pas d'émmondage, moins variable quant à β)
- Performe beaucoup mieux que les arbres de décision et conserve la majorité de ces forces.
- On perd un peu d'interpretabilité, mais ça ne valais pas grand chose, l'indice de permutation est plus robuste.
- Plusieurs librairies sur R et Python.
- C'est le modèle linéaire des gens en Machine learning, le modèle de base facile à utilisé!

Exercices:

- Lire le chapitre 8 (révision) et chapitre 10.1, 10.2 en préparation.
- Lab: 8.3.3
- Exercises 8.4.5

Références

Biau, G., Devroye, L., and Lugosi, G. (2008). Consistency of random forests and other averaging classifiers. *Journal of Machine Learning Research*, 9(9).

Biau, G. and Scornet, E. (2016). A random forest guided tour. *Test*, 25:197–227.

Friedman, J., Hastie, T., Tibshirani, R., et al. (2001). *The elements of statistical learning*. Springer.

Genuer, R. and Poggi, J.-M. (2020). *Random Forests witgh R*. Springer.

James, G., Witten, D., Hastie, T., Tibshirani, R., et al. (2013). *An introduction to statistical learning*, volume 112. Springer.

Zhang, C. and Ma, Y. (2012). *Ensemble machine learning*, volume 144. Springer.