Devoir : aspects pratiques

ENSTA

2019/2020

Adresse mail: mohammed.sedki@universite-paris-saclay.fr Page web: masedki.github.io

---

Arbre de décision unique

Présentation du jeu de données * Orange Juice oj.csv*

Les données décrivent 1070 achats où le client a acheté soit *Citrus Hill,* soit le jus d'orange *Minute Maid*. Un certain nombre de caractéristiques du client et du produit sont enregistrées.

Le jeu de données contient 1070 observations décrites par les 18 variables suivantes (reprises en anglais) :

- `Purchase` :A factor with levels CH and MM indicating whether the customer purchased Citrus Hill or Minute Maid Orange Juice
- `WeekofPurchase`: Week of purchase
- `StoreID`: Store ID
- `PriceCH`: Price charged for CH
- `PriceMM`: Price charged for MM
- `DiscCH`: Discount offered for CH
- `DiscMM`: Discount offered for MM
- `SpecialCH`: Indicator of special on CH
- `SpecialMM`: Indicator of special on MM
- `LoyalCH`: Customer brand loyalty for CH
- `SalePriceMM`: Sale price for MM
- `SalePriceCH`: Sale price for CH
- `PriceDiff`: Sale price of MM less sale price of CH
- `Store7`: A factor with levels No and Yes indicating whether the sale is at Store 7
- `PctDiscMM`: Percentage discount for MM
- `PctDiscCH`: Percentage discount for CH

- `ListPriceDiff`: List price of MM less list price of CH
- `STORE`: Which of 5 possible stores the sale occured at

1. Créer un jeu de données d'apprentissage composé de 800 observations échantillonnées au hasard. Les observations restantes serviront de jeu de données test.
2. Ajuster deux arbres de décision (avec et sans élagage) pour expliquer la variable `Purchase` par les autres variables du jeu de données. Comparer les erreurs de test des deux arbres de décision ainsi que les matrices de confusion des deux classifications du jeu de données test.
3. Tracer l'arbre réalisant la meilleure erreur de test et décrire les informations décrites sur une feuille au choix. Afficher l'importance relative des variables.

Forêt aléatoires

Présentation du jeu de données *email.csv*

Dans cette partie nous allons nous intéresser au jeu de données `email.csv`. Les différentes colonnes du jeu de données sont décrites ici.

4. Créer une partition aléatoire apprentissage-test (75%-25%). Ajuster un arbre de décision élagué pour prédire la variable `Class` à partir des autres variables. Évaluer l'erreur de test de l'arbre obtenu.
5. Ajuster un modèle pour expliquer la variable `Class` en fonction des autres variables à l'aide d'un bagging de 100 arbres de décision. Évaluer l'erreur de test du modèle obtenu.
6. Ajuster une règle de classification à l'aide d'une forêt aléatoire avec 100 arbres de décision. Utiliser la librairie appropriée pour choisir les hyper-paramètres du modèle. Vérifier la reproductibilité des résultats de la procédure de choix d'hyper-paramètres. Évaluer l'erreur de test du modèle final.
7. Commenter la hiérarchie des différents modèles en terme de précision ?

Boostrap

La méthode bootstrap a été introduite pour réduire la variance des arbres de décision. Toutefois, la méthode bootstrap est largement applicable dans d'autres contextes également, par exemple pour estimer la variance d'un paramètre (cf. compromis biais-variance).

8. Générer $y_1, \ldots, y_n$ ($n = 100$) réalisations d'une loi normale de moyenne 4 et variance 1. On souhaite estimer par maximum de vraisemblance, le paramètre θ du modèle jouet $Y = \theta + \varepsilon$ où $\varepsilon \sim \mathcal{N}(0,1)$. On note cet estimateur $\widehat{\theta}$. Tracer l'histogramme de 1000 réalisations indépendantes de $\widehat{\theta}$.
9. En pratique nous n'avons accès qu'au $n = 100$ réalisations $y_1, \ldots, y_n$. Toutefois, avec la méthode bootstrap, nous pouvons obtenir un échantillon bootstrap $y_1^*, \ldots, y_n^*$ par tirage avec remise à partir du jeu de données accessible $y_1, \ldots, y_n$. On notera $\widehat{\theta}^*$ l'estimateur de θ basé sur l'échantillon bootstrap $y_1^*, \ldots, y_n^*$. Générer 1000 réalisations boostrap $\widehat{\theta}_1^*, \ldots, \widehat{\theta}_{1000}^*$ et tracer l'histogramme de ces réalisations. Comparer cet histogramme au précédent.

Autour de l'algorithme Adaboost

Fonctions de perte exponentielle et binaire

Dans une approche de boosting, on assigne les observations aux classes $+1$ et -1. La règle de classification $\widehat{y}(x)$ provient d'un seuillage d'une fonction $c(x)$. Supposons que nous avons ajusté une fonction $\widehat{c}(x)$, la règle de classification est donnée par $\widehat{y}(x) = \mathbb{I}\{\widehat{c}(x) \geq 0\} - \mathbb{I}\{\widehat{c}(x) < 0\}$.

10. Compléter le tableau suivant. Dans quel sens la perte exponentielle est-elle plus informative que la perte binaire et comment cette information peut être utilisée lors de l'apprentissage ?

$\widehat{c}(x_\star)$	$\widehat{y}_\star$	**Perte exponentielle** $\exp(-y_\star \widehat{c}(x_\star))$	**Perte binaire** $\mathbb{I}(y_\star \neq \widehat{y}_\star)$	$y_\star$
0.3				-1
-0.2				-1
1.5				1
-4.3				1

11. Reprendre la partition précédente du jeu données *email.csv*, ajuster une règle de classification à l'aide de l'algorithme **adaboost**. La fonction `ada` du package du même nom permet d'implémenter une procédure **adaboost**. Le but ici n'est pas de choisir les valeurs optimales des hyper-paramètres. Expliquer les valeurs qui permettent de reproduire la version originale de l'algorithme **AdaBoost.M1** proposée par (Freund and Schapire 1997). Évaluer l'erreur de test du modèle après 50 itérations d'entraînement.

12. À partir d'un jeu de données synthétique de votre choix, illustrer[1] le phénomène de surajustement du au nombre d'itérations et du pas de la descente du gradient de l'algorithme de gradient boosting.

[1] Il suffit de de choix extrêmes du nombre d'itération. Cette comparaison peut être faite au choix à l'aide de l'un des deux packages `xgboost` ou `gbm`.