Projet - Arbres de décision optimaux

Zacharie Ales

2022

L'objectif de ce projet est de construire des arbres de décision optimaux pour plusieurs jeux de données en utilisant la modélisation F.

1 Description du code Julia fourni

Le code contient :
— 3 jeux de données (Iris, Wine et Seeds) ;
— La définition de la formulation F présentée en cours pour des séparations univariées ou multivariées ;
— Les deux méthodes (naïve et exacte) de regroupement de données vues en cours.

Ci-dessous une description des principaux fichiers et répertoires :
— **Fichier** `main.jl`
Contient une méthode `main()` qui applique F univarié et multivarié sur plusieurs jeux de données pour plusieurs profondeurs d'arbre ($D \in \{2, 3, 4\}$).

Pour exécuter la méthode, taper simplement `main()` dans une console Julia après avoir inclus le fichier (`include("main.jl")`).

— **Fichier** `building_tree.jl`
Contient la définition des méthodes de résolution de F (1 méthode sans regroupement et 1 avec regroupement).

— **Fichier** `main_merge.jl`
Similaire à `main.jl` mais en appliquant l'algorithme de regroupement naïf vu en cours pour $\gamma \in \{0, 0.2, 0.4, 0.6, 0.8, 1\}$.

Pour exécuter la méthode, taper simplement `main_merge()` dans une console Julia après avoir inclus le fichier (`include("main_merge.jl")`).

— **Fichier** `merge.jl`
Contient les fonctions
— `simpleMerge()` : effectuant l'algorithme naïf de regroupement de données ;
— `exactMerge()` : effectuant le regroupement exact vu en cours
(il n'est pas utilisé dans `main_merge()` car en pratique il ne parvient à regrouper aucune donnée des trois jeux de données testés).

— **Répertoire** `data`
Contient les jeux de données.

— **Répertoire** `struct`
Contient les différentes structures utilisées dans le code (`Tree`, `Cluster` et `Distance`).

2 Travail demandé

Appliquer les méthodes `main()` et `main_merge()` sur les 3 jeux de données ainsi que 2 autres jeux de données de votre choix. Attention :
— les caractéristiques des jeux de données que vous ajoutez doivent être dans $[0, 1]$ pour que la modélisation soit valide ;

— le vecteur Y contenant la classe de chaque donnée doit contenir des entiers entre 1 et le nombre de classes.

Vous devez ensuite traiter une des questions d'ouverture suivante :

1. Proposer et tester d'autre(s) méthode(s) de regroupement ;
2. Démontrer que certains regroupement permettent d'obtenir des arbres de décision optimaux ou obtenir une borne sur la détérioration du nombre de bonnes classifications ;
3. Implémenter l'heuristique figurant dans [Dunn 2018] et comparer ses performances aux méthodes fournies ;
4. Identifier et utiliser des inégalités valides pour améliorer les performances de résolution. Vous pouvez également améliorer les performances en retirant des contraintes de la formulation que vous ajouterez si besoin au cours de la résolution (nécessite l'utilisation de callbacks).
5. Tout autre idée permettant d'améliorer les temps de calculs ou la qualité des prédictions.

3 Rendu

Rendre pour le 31 mars :

— vos fichiers (données, code, ...) ;

— votre rapport contenant

  — vos résultats commentés : temps de calcul et performance du classifieur en fonction du jeu de données, de la profondeur de l'arbre, du type de séparations et du type de regroupement considéré, le cas échéant.

  — la description de votre traitement de la question d'ouverture.

Le rapport peut être succint. L'important est de bien y décrire le travail que vous avez effectué. Il n'est par exemple par nécessaire de présenter dans votre rapport le contexte du projet (modélisation F, type de séparations, principe du regroupement, ...).