U4.1 Apprentissage Automatique 1

TP Arbres de Décision

Master SID
Science des Données
Année universitaire 2019-2020
simon.bernard@univ-rouen.fr

Contents

Ce TP est à traiter en Python dans un *notebook Jupyter*, qui sera à soumettre au dépôt UniversiTICE prévu à cet effet. Pensez à mentionner vos noms et prénoms dans le *notebook*. La date limite de dépôt sera précisée sur l'espace UniversiTICE et/ou par votre encadrant.

1 Préparation de l'environnement

Dans ce TP, vous allez utiliser une librairie de *Machine Learning* très complète appelée *Scikit-Learn*. Si ce n'est déjà fait, vous pouvez installer cette librairie avec la commande:

```
$ pip3 install -U scikit-learn
ou
$ conda install scikit-learn
```

Scikit-Learn propose de nombreuses implémentations de méthodes d'apprentissage, parmi lesquels une méthode de construction automatique d'arbres de décision. Pour traiter les exercices de cet énoncé, vous aurez besoin de consulter la documentation:

- Les références de l'API : `http://scikit-learn.org/stable/modules/classes.html`
- Le guide de l'utilisateur: `http://scikit-learn.org/stable/user_guide.html`

Vous aurez également besoin de la documentation des librairies *Matplotlib* et *NumPy*:

- `https://matplotlib.org/contents.html`
- `https://docs.scipy.org/doc/`

Pour finir, vous utiliserez l'outil *Graphviz*[1] pour visualiser la structure des arbres de décisions qui auront été construits. Il existe des visualiseurs en ligne[2] qui nous évite l'instal-

[1] `http://www.graphviz.org/`
[2] `http://viz-js.com/`

lation de *packages* python supplémentaires. On peut cependant aussi utiliser des outils permettant de visualiser et d'exporter des graphes *Graphviz* directement en python:

```
$ pip3 install -U graphviz pydotplus
ou
$ conda install python-graphviz pydotplus
```

2 Construire et visualiser un arbre de décision

Ce premier exercice vise à tester l'apprentissage d'arbres de décision sur la base de données *Iris* que vous avez vu en cours:

- ▷ Importez les modules `sklearn.datasets` et `sklearn.tree`.
- ▷ Utilisez la fonction `load_iris`[3] pour charger la base *Iris*.
- ▷ Utilisez la fonction `DecisionTreeClassifier`[4] pour construire un arbre de décision sur cette base. Vous utiliserez les paramètres par défaut de l'algorithme.

Pour visualiser le résultat de cet apprentissage, vous pouvez exporter la structure de l'arbre obtenu au format `DOT` de *Graphviz*:

- ▷ Utilisez la fonction `export_graphviz` pour exporter la structure de l'arbre dans un fichier `.dot`.
- ▷ Chargez le contenu du fichier obtenu dans un visualiseur en ligne.
- ▷ Si vous avez installé les *packages* python pour *Graphviz*, vous pouvez visualiser et/ou exporter directement le graphe dans le script python[5]

Vous devriez obtenir l'arbre représenté en figure 1.

3 Comparaison de classifieurs

Cet exercice vise à comparer les performances d'un arbre de décision à celles des classifieurs *Gaussian Naive Bayes* et *K-Plus Proches Voisins*:

- ▷ Découper la base *Iris* en deux sous-ensembles: un pour l'apprentissage et l'autre pour évaluer les performances en test. *Scikit-learn* propose des fonctions pour faire cela[6].

[3] `https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_iris.html`
[4] `https://scikit-learn.org/stable/modules/tree.html`
[5] `https://pythonprogramminglanguage.com/decision-tree-visual-example/`
[6] `https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html`

- ▷ Lancez l'apprentissage d'un arbre de décision sur le sous-ensemble d'apprentissage.
- ▷ Lancez l'apprentissage d'un `GaussianNB`[7] et d'un `KNeighborsClassifier`[8] sur le même sous-ensemble d'apprentissage.
- ▷ Pour chacun de ces classifieurs, calculez le taux de bonne classification et affichez le résultat, à l'aide de la fonction `score`.

Pour obtenir un indicateur de performance plus fiable, vous allez recommencer en utilisant une validation croisée:

- ▷ Lancez une procédure de validation croisée à 5 *folds* pour calculer un taux de bonne classification moyen et un écart-type correspondant. *Scikit-learn* propose des fonctions pour faire cela[9].
- ▷ Pour chaque classifieur, affichez le résultat sous la forme:

```
Accuracy: mean (+/- std)
```

où `mean` et `std` sont à remplacer par les valeurs obtenues

Maintenant que votre protocole expérimental est établi, vous pouvez reproduire cette comparaison sur d'autres bases de données:

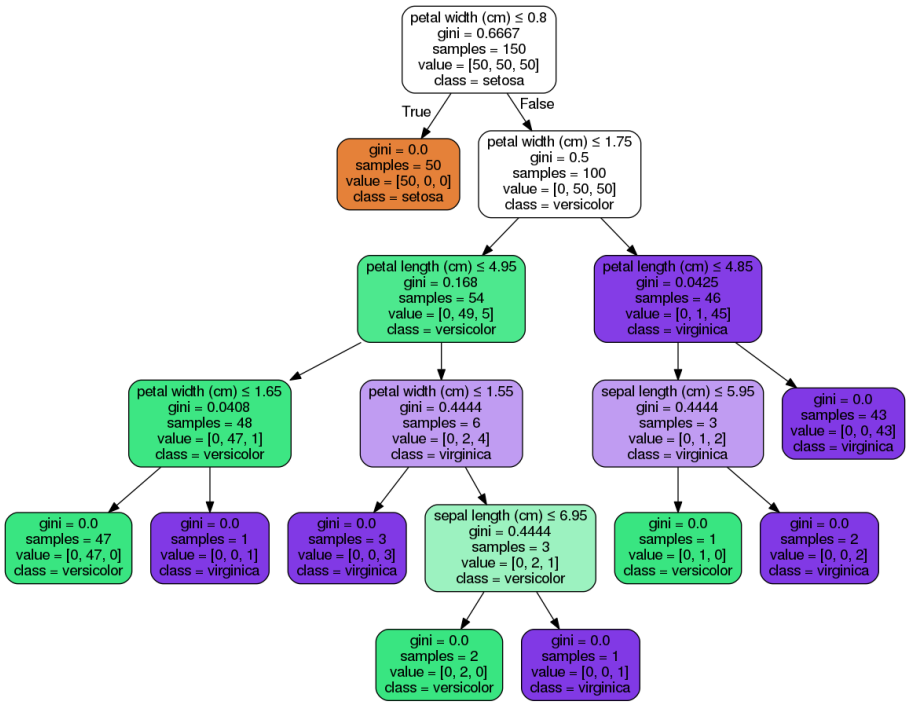

Figure 1: Arbre de décision appris sur l'intégralité de la base Iris

[7] `https://scikit-learn.org/stable/modules/naive_bayes.html`
[8] `https://scikit-learn.org/stable/modules/neighbors.html`
[9] `https://scikit-learn.org/stable/modules/cross_validation.html`

- ▷ Reproduisez cette comparaison sur plusieurs bases de données jouets proposées par *Scikit-Learn*[10].
- ▷ Analysez les résultats et donnez vos conclusions.

4 Visualiser les frontières de décision

Sur un problème de classification en deux dimensions, on peut facilement visualiser les frontières de décision d'un arbre de décision et analyser comment elles évoluent en fonction de certains paramètres d'apprentissage.

- ▷ Générer une base de données synthétique en deux dimensions avec la fonction `make_moon` de *Scikit-learn*[11]. Cette base sera composée de 2 classes, 500 instances, et un taux de bruit de 0.2.
- ▷ Lancez l'apprentissage d'un arbre de décision, d'un `GaussianNB` et d'un `KNeighborsClassifier` sur cette base, comme à l'exercice précédent.
- ▷ Créez une fonction python `plot_decision_frontiers` qui prend en paramètre un classifieur, sa base d'apprentissage et une figure dans laquelle seront dessinés les points (2D) de la base d'apprentissage et les frontières de décision du classifieur. Cette fonction procédera en plusieurs étapes:
  1. Générer une grille de points 2D: $\{(i, j)\}$ pour $i = x_{min}^{(1)} \dots x_{max}^{(1)}$ et $j = x_{min}^{(2)} \dots x_{max}^{(2)}$. Pour cela, utilisez la fonction `meshgrid` de *NumPy*.
  2. Rassembler les prédictions de l'arbre pour chacun des points de cette grille. Les fonctions `ravel`, `c_` et `reshape` de *NumPy* pourront vous être utiles.
  3. Afficher les points de la base d'apprentissage avec la fonction `scatter` de *Matplotlib*.
  4. Afficher les frontières de décisions en utilisant la fonction `contourf` de *Matplotlib*.
- ▷ Créez une figure contenant 3 sous-figures avec la fonction `subplots` de *Matplotlib*.
- ▷ Appelez la fonction `plot_decision_frontiers` pour chacun des trois classifieurs et chacune des 3 sous-figures.

5 Étude de paramètres

L'objectif est d'analyser l'influence de quelques paramètres sur les performances d'un arbre de décision. Dans un premier temps, analysez l'effet du critère de partitionnement:

- ▷ Générez une nouvelle base avec `make_moon`, avec 1000 données et un taux de bruits de 0.3.

[10] `https://scikit-learn.org/stable/datasets/index.html#toy-datasets`
[11] `https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_moons.html`

- ▷ Lancez l'apprentissage de deux arbres de décision sur cette base: l'un avec le critère de partitionnement *Gini*, l'autre avec *Entropy*.
- ▷ Visualisez les frontières de décision des deux classifieurs ainsi obtenus en utilisant la fonction définie à l'exercice précédent.
- ▷ Estimez également les performances de ces deux algorithmes d'apprentissage avec une procédure de validation croisée.
- ▷ Analysez les résultats et donnez vos conclusions.

Dans un deuxième temps, vous allez créer une expérience pour visualiser l'évolution des performances en fonction des valeurs de deux paramètres:

- ○ la profondeur maximum de l'arbre de décision
- ○ le nombre minimum de données par feuille

Pour cela:

- ▷ Faites varier les valeurs de chaque paramètre (séparément):
  - ○ Profondeur maximum: de 1 à 30
  - ○ Nombre minimum de données par feuille (en pourcentage): de 1% à 50%
- ▷ Pour chaque valeur, testez l'apprentissage d'un arbre via une procédure de validation croisée et mémoriser le taux moyen de bonne classification dans un tableau.
- ▷ Tracez la courbe de l'évolution du taux moyen de bonne classification en fonction des valeurs de paramètre testées.
- ▷ Analysez les résultats et donnez vos conclusions.