TP1 - Forêt d'arbres aléatoires et dilemme Biais/Variance

Ce TP est composé de deux parties. La première concerne les forêts d'arbres aléatoires et la seconde est dédiée à l'étude du compromis Biais/Variance.

Forêt d'arbres aléatoires

I. Chargement et visualisation des données

Chargez les données (TP1a.npy) et visualisez-les avec la commande :

```
data = np.load("TP1a.npz")
X_train, y_train, X_test, y_test = (data[key] for key in ["X_train", "y_train", "X_test", "y_test"])
plt.scatter(X_train[:, 0], X_train[:, 1], c=y_train, s=1, cmap='rainbow');
plt.show()
```

Questions

Combien y a-t-il de points dans la base d'apprentissage ? Dans la base de test ? Quelle est la dimension des données ?

II. Arbre de décision

a. Principe des arbres de décision

On réalise la classification avec un arbre de décision. Pour reproductibilité lors de la correction, la variable random_state correspondra aux trois derniers chiffres de votre numéro d'étudiant.

```
from display import visualize_classifier
tree1 = DecisionTreeClassifier(criterion='entropy', max_depth = 3, random_state= ****)
tree1.fit(X_train, y_train)
visualize_classifier(tree1, X_train, y_train)

tree.plot_tree(tree1)
plt.show()
text_representation = tree.export_text(tree1)
print(text_representation)
```

Questions

Que représente la variable max_depth ?

Qu'observe-t-on avec visualize_classifier, tree.plot_tree et tree.export_text ?

Retrouvez-vous visuellement toutes les découpes sur la figure issue de visualize_classifier?

b. Performance d'un classifieur muti-classes

Réalisez la classification sur la base de test avec :

```
y_pred = tree.predict(X_test)
C=confusion_matrix(y_test, y_pred)
print(classification_report(y_test, y_pred))
print('Accuracy= ',accuracy_score(y_test, y_pred))
```

Questions

Que représentent les valeurs renvoyées par classification_report ?
Retrouvez par le calcul la première ligne (classe 1) renvoyée par classification_report à partir de la matrice de confusion.

c. Optimisation de la profondeur de l'arbre

Faites varier max_depth et estimez pour chaque valeur le taux de reconnaissance. Conclusion sur l'évolution du taux de reconnaissance en fonction de la profondeur de l'arbre.
Affichez le partitionnement de l'espace obtenu pour l'arbre avec le meilleur taux de classification

Questions
Est-ce que ce partitionnement vous paraît visuellement satisfaisant ? Commentez.

d. Arbre de décision sur des données de grande dimension

Chargez maintenant les données TP1b.npz et utilisez la question précédente pour optimiser la profondeur de l'arbre.

Questions
Combien y a-t-il de points dans la base d'apprentissage ? Dans la base de test ? Quelle est la dimension des données ?
Quel est le taux de classification obtenu avec le meilleur classifieur ?

III. Forêt d'arbres aléatoires

a. Test d'une forêt d'arbres aléatoires

Utilisez RandomForestClassifier pour construire une forêt d'arbres aléatoires. Pour reproductibilité lors de la correction, la variable random_state correspondra aux trois derniers chiffres de votre numéro d'étudiant.

```
from sklearn.ensemble import RandomForestClassifier
RF = RandomForestClassifier(criterion='entropy', n_estimators=10, random_state=***)
RF.fit(X_train, y_train)
```

Testez le classifieur avec les paramètres par défaut et 10 arbres. Conclusion.

b. Influence des paramètres

- Observez l'évolution du taux de bonne classification en fonction du nombre d'arbres. Conclusion sur le meilleur compromis temps/Taux.
- En conservant le nombre d'arbres optimal, observez l'évolution du taux de bonne classification en fonction de la profondeur de l'arbre. Conclusion.
- En conservant le nombre d'arbres et la profondeur, observez l'évolution du taux de bonne classification en fonction du nombre de caractéristiques utilisées lors du bagging. Conclusion.

Est-ce que toutes ces évolutions étaient prévisibles ?

c. Choix des paramètres optimaux

Utilisez la fonction GridSearchCV() pour optimiser les paramètres de la forêt autour des valeurs trouvées précédemment.

Quel est le taux de reconnaissance obtenu avec les paramètres trouvés par GridSearchCV ?

Conclusion

Questions
Que représente le paramètre cv de GridSearchCV ?

Compromis Biais/Variance et autre

I. Chargement et visualisation des données

Chargez les données (TP1c.npz) et visualisez-les avec la commande :

```
data = np.load("TP1c.npz")
X_train, y_train, X_test, y_test = (data[key] for key in ["X_train", "y_train", "X_test", "y_test"])
plt.scatter(X_train[:, 0], X_train[:, 1], c=y_train, s=1, cmap='rainbow');
plt.show()
plt.scatter(X_test[:, 0], X_test[:, 1], c=y_test, s=1, cmap='rainbow');
plt.show()
```

Questions
Combien y a-t-il de points dans la base d'apprentissage ? Dans la base de test ? Quelle est la dimension des données ?
Combien y a-t-il de classes ? Est-ce que les classes sont bien équilibrées dans la base d'apprentissage et la base de test ?

II. Calcul du biais et de la variance du classifieur 1PPV

On souhaite calculer le biais et la variance du classifieur 1PPV. Pour cela, réalisez Nrun=30 apprentissages en tirant aléatoirement 60% de la base d'apprentissage (random_state sera initialisé avec la somme des 3 derniers numéros de votre carte d'étudiants+le numéro du run) . Vous prédirez à chaque fois la classe des exemples de la base de test. A partir des Nrun prédictions, calculez le biais et la variance du classifieur. On pourra pour cela utiliser la fonction mode() de scipy.stats.

Pour un exemple de test,

- le biais vaut 0 si la classe avec la plus grande prédiction est la classe correcte, 1 sinon.
- la variance vaut $1 - \textit{probabilité de la classe avec la plus grande prédiction.}$

Le biais et la variance sont ensuite obtenus en moyennant sur tous les exemples de test. Par exemple, considérons un exemple de test appartenant à la classe 1. Parmi les 30 runs, cet exemple est classé 28 fois 1 et deux fois autre chose. Pour cet exemple, le biais est 0 (car la classe avec le plus de prédictions est la classe correcte) et la variance est 1-28/30.

III. Etude du compromis biais/variance du classifieur k-PPV

Tracez l'évolution du biais et de la variance en fonction du k des k-ppv. Déterminez la valeur de k qui amène au meilleur compromis Biais, Variance.

Questions

Comment évoluent le biais et la variance avec *k* ? Retrouve-t-on la théorie ? Comment trouver la valeur de k qui optimise le compromis Biais, variance ?

IV. Reprendre la même étude en utilisant un arbre de décision

Pour les arbres de décision, l'étude sera réalisée en fonction de la profondeur maximale.

Questions
Comment évoluent le biais et la variance avec k ? Retrouve-t-on la théorie ?Comment trouver la valeur de k qui optimise le compromis Biais/variance ?

V. Reprendre la même étude avec les random forest

Pour les forêts d'arbres aléatoires, l'étude sera réalisée en fonction du nombre d'arbres de la forêt.

Questions
Comment évoluent le biais et la variance avec Nb ? Retrouve-t-on la théorie ? Trouver la valeur de Nb qui optimise le compromis Biais/variance ?

VI. Test sur un nouveau jeu de données

On s'intéresse maintenant aux données « TP1d.npz ». Utilisez les hyperparamètres optimaux trouvés à la question précédente et réalisez la classification des données de test avec les random forest.

Affichez le rapport de classification et commentez le résultat en fonction des classes. Conclusion ? D'où vient le problème ? Observer bien les bases d'apprentissage et de test.

Proposez deux solutions afin de remédier au problème. Conclusion sur celle qui amène aux résultats les plus performants.