Cycle Ingénieurs : Deuxiéme Année - Parcours MI

Examen de fouille de données

A rendre sur examManager

14 décembre 2018

Durée 3h - Documents de cours et de TDs autorisés

1 Analyse d'un jeu de données

Le jeu de données utilisé dans cet examen provient du package mlbench.
library(mlbench)
data(Vehicle)
help(Vehicle)

The purpose is to classify a given silhouette as one of four types of vehicle, using a set of features extracted from the silhouette. The vehicle may be viewed from one of many different angles. The features were extracted from the silhouettes by the HIPS (Hierarchical Image Processing System) extension BINATTS, which extracts a combination of scale independent features utilizing both classical moments based measures such as scaled variance, skewness and kurtosis about the major/minor axes and heuristic measures such as hollows, circularity, rectangularity and compactness. Four "Corgie" model vehicles were used for the experiment : a double decker bus, Cheverolet van, Saab 9000 and an Opel Manta 400. This particular combination of vehicles was chosen with the expectation that the bus, van and either one of the cars would be readily distinguishable, but it would be more difficult to distinguish between the cars.

Le jeu de données comprend 846 véhicules caractérisés par 19 variables dont 18 variables explicatives, toutes continues, et une variable cible, le type de véhicule, variable qualitative à 4 modalités :

summary(Vehicle$Class)
bus opel saab van
218 212 217 199

1.1 Préparations des données

Les instructions suivantes permettent de nommer les variables entre -1 et 1 :

```
Mydata=Vehicle
# Find max and min of each column (except the target)
max = apply(Mydata[,1 :18] , 2 , max)
min = apply(Mydata[,1 :18], 2 , min)
data.scaled = scale(Mydata [,1 :8], center = min, scale = max-min) # Scale the data
data.scaled=as.data.frame(data.scaled)
data.scaled=cbind(data.scaled,Mydata$Class) # Add the target
names(data.scaled)=names(Mydata) # Add the column names
```

— Construire une base d'apprentissage *data.train* et une base test *test.data* avec un ratio 2/3, 1/3. A quoi sert chacune de ces bases ?

1.2 Réseaux de neurones

Nous allons considérer un réseau de neurones avec :
— Nc neurones sur la couche cachée
— Un biais entre la couche d'entrée et la couche cachée
— Un biais entre la couche cachée et la couche de sortie

1. Combien y-a-t-il de neurones sur la couche d'entrée (biais compris) ?
2. Combien y-a-t-il de neurones sur la couche de sortie ?
3. A quoi correspondent les neurones de la couche de sortie ?
4. Comment fait-on pour prédire la classe d'une nouvelle entrée à partir de la couche de sortie du réseau de neurones ?
5. Combien y-a-t-il de poids à ajuster dans ce réseau de neurones ?
6. A l'aide de la fonction nnet, ajuster un réseau à 30 neurones avec une fonction d'activation sigmoïde ($logit^{-1}$) entre la couche d'entrée et la couche cachée. Combien faut-il d'itérations (environ) à l'algorithme d'optimisation pour l'ajustement des poids avant de converger ? Que se passe-t-il si on passe à 5 neurones ?
7. Construire la matrice de confusion sur la base test. En déduire le taux d'erreur de prévision.
8. Est-ce que certains types de véhicules sont plus facilement reconnus que d'autres ou vice-versa ?

1.3 Arbres de décision

Nous allons construire un arbre de décision pour prédire le type de véhicule.

1. Construire un arbre complet sur la base d'apprentissage. Utiliser la fonction rpart du package du même nom. Y-a-t-il des feuilles pures ? On pourra utiliser l'instruction prp(arbre,extra=1) avec la fonction prp du package rpart.plot, pour représenter l'arbre.
2. Pour quelle valeur du coefficient de complexité (cp) faut-il élaguer l'arbre ? On utilisera la fonction plotcp.
3. Construire l'arbre élagué avec la fonction prune.Y-a-t-il des feuilles pures ?
4. Construire la matrice de confusion sur la base test. En déduire le taux d'erreur de prévision.
5. Comparer avec les réseaux de neurones.

1.4 K-Means

1. Appliquer l'algorithme k-means sur *data.scaled* après avoir enlevé la variable cible et en prenant k=10.
2. Relever les inerties intra-clusters ainsi que l'inertie inter-cluster.
3. Trouver la matrice de correspondance classes-clusters. Quelle est la différence avec la matrice de confusion ?
4. Proposer une mesure de pureté d'un cluster par rapport aux valeurs de la variable cible. Quel est le cluster ayant la pureté maximale ? Quel est celui ayant la valeur minimale ?
5. Y-a-t-il un lien entre la pureté d'un cluster et son inertie dans ce dataset ? Peut-on trouver une explication ?

1.5 Random Forest

1. Rappelez brièvement (3-4 phrases) la manière dont une "forêt aléatoire" est construire et la manière dont elle est utilisée dans un problème de classification.
2. Construire maintenant une base d'apprentissage *Mydata.train* et une base test *Mydata.test* à partir de Mydata avec un ratio 2/3, 1/3.
3. En utilisant la base d'apprentissage construire une forêt aléatoire *rf* avec un nombre d'arbres (*ntree*) égal à 50 et un nombre de variables à tester à chaque nœud (*mtry*) égal à 4.
4. Appliquez *rf* à la base de test. Donnez la matrice de confusion et déduisez-en l'erreur.

5. Refaites le travail demandé dans les deux questions précédentes et notez les erreurs obtenues. Que constatez-vous ?
6. Expliquez pourquoi les forêts aléatoires peuvent poser des problèmes de '**stabilité**' : pour le même ensemble d'apprentissage ont peut en lançant l'algorithme deux fois, obtenir deux forêts différentes et ayant des performances différentes.
7. Donnez la valeur de l'**importance** des variables et représentez la graphiquement. Quelle signification donner à ces valeurs ?