

TP no 4 - courbe ROC et k -NN

Master parcours SSD - UE Apprentissage Statistique I

1 Exercice 1 - courbes ROC

Dans cet exercice nous allons manipuler des courbes ROC avec le package `ROCR`.

1. Charger le jeu de données `tp-4_exo-1.Rdata`. Il se constitue du tableau de données (`data.frame`) `data` contenant de 200 lignes et 4 colonnes :
 - chaque ligne correspond à une observation.
 - la colonne `label` correspond à la classe de l'observation (0 ou 1).
 - les colonnes `score1`, `score2`, `score3` contiennent les scores obtenus par 3 classifieurs.
2. Calculer les matrices de confusion et les valeurs de sensibilité et spécificité obtenues par ces 3 classifieurs en fixant le seuil de décision à 0 : on affecte une instance à la catégorie "1" si son score est positif (ou nul) et à la catégorie "0" sinon.
3. Calculer les courbes ROC que l'on obtient en faisant varier ce seuil, en utilisant les fonctions `prediction` et `performance` du package `ROC`. Les représenter sur un même graphique.
 - voir la documentation et/ou l'exemple donné en cours.
 - NB : on peut tracer plusieurs courbes ROC sur une même figure en utilisant l'option `add=TRUE` de la fonction `plot.performance`.
4. Faire figurer sur ce graphique les valeurs de sensibilité / spécificité obtenues à la question 2.
5. Quelles valeurs de spécificité peut-on obtenir avec ces trois classifieurs pour une sensibilité de 75% ? A l'inverse, quelles valeurs de sensibilité peut-on obtenir pour une sensibilité de 75% ? Commenter.
6. Calculer les AUC correspondantes. Commenter.
 - idem, voir la documentation de la fonction `performance` et/ou l'exemple donné en cours.

2 Exercice 2 - classification par k plus proches voisins

Dans cet exercice, nous allons réaliser de la classification par k plus proche voisins en utilisant la fonction `knn` du package `class`. Nous travaillerons sur un jeu de chiffres manuscrits tel que celui utilisé dans un TP précédent. L'objectif de l'exercice sera (i) d'optimiser la valeur de k à utiliser par validation interne sur le jeu d'apprentissage, et (ii) de mesurer les performances de prédiction sur un jeu de test indépendant.

1. charger le jeu de données `tp-4_exo-2.Rdata`. Il se constitue :
 - d'un jeu d'apprentissage de 7291 caractères défini (i) par une matrice `X` de taille 7291 x 256 (7291 caractères représentés par 16x16 = 256 pixels), et (ii) par un vecteur `y` de longueur 7291 donnant la catégorie – i.e., le chiffre entre 0 et 9 – des différents caractères
 - d'un jeu de test de 2007 caractères, défini de la même manière par une matrice `X.test` et un vecteur `y.test`.
2. Séparer le jeu d'apprentissage en deux de manière à définir :
 - un jeu de validation, de taille 2000, qui nous servira à évaluer les performances de classification

- un (sous-)jeu d'apprentissage, contenant les 7291-2000 = 5291 instances restantes, qui nous servira de jeu d'apprentissage interne.
- 3. Evaluer les performances de classification obtenues par l'algorithme k -NN pour des valeurs de k prises entre 1 et 10, en utilisant les jeux d'apprentissage/validation créés ci-dessus. On utilisera pour cela la fonction `knn` du package `class`. Calculer le taux de bonne classification obtenu sur le jeu de validation pour les différentes valeurs de k . Quelle valeur donne les meilleurs résultats ?
— NB : la procédure prend un peu de temps.
- 4. Appliquer l'algorithme k -NN pour classifier le jeu de test à partir du jeu d'apprentissage global, en utilisant la valeur de k retenue. Quelle taux de bonne classification obtient-on ?
- 5. Construire la matrice de confusion à partir des prédictions obtenues, et calculer le taux de bonne classification obtenu pour chaque caractère. Quel caractère est le moins bien reconnu ? Avec quel(s) autre(s) caractère(s) le confond t'on ? Commenter.