Rapport TP 10 - Apprentissage à partir de trois jeux de données réelles

Assila Yassine, Gabet Joseph, Sébastien Dam

A21

Introduction

Ce rapport a pour but de présenter nos démarches pour construire des fonctions de prédiction les plus performantes possibles pour les trois jeux de données étudiés. Pour chacun d'entre eux, nous avons réalisé une analyse exploratoire pour avoir une première connaissance des données. Puis, nous avons utilisé les méthodes d'apprentissage vues en cours, avec dans certains cas l'utilisation de variables sélectionnées par différentes méthodes afin d'améliorer les résultats.

Phonemes

Analyse exploratoire

Le jeu de données `Phonèmes` peut être traité comme un problème de classification avec cinq classes. Les fréquences de chaque phonème sont assez équitablement réparties. Nous représentons le log-periodogram sur la figure 1. On observe que les phonèmes `dcl` et `sh` sont assez différents mais pour les autres phonèmes, leur log-periodogram est assez similaire.

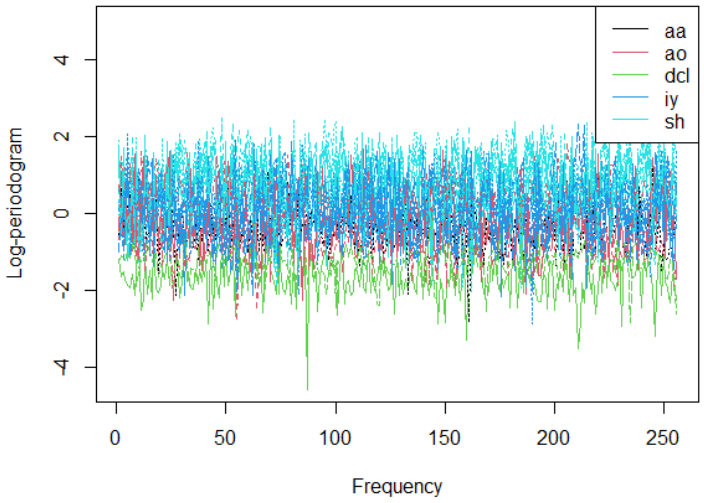

Figure 1: Log-periodogram

Sélection de variables

Au vu du grand nombre de variables, nous décidons d'appliquer des méthodes de sélection de variables. Les variables étant les longueurs des log-periodograms, il ne serait pas pertinent d'utiliser des méthodes telles que les forward/backward subset selection ou de récupérer les variables les plus importantes mesurées par une forêt aléatoire. En effet, cela n'aurait pas de sens de sélectionner seulement certaines fréquences puis de réaliser une classification sur ces dernières. Cependant, nous pouvons chercher à réduire le nombre de variables tout en conservant le maximum d'information, en transformant notamment les variables. Nous décidons alors d'appliquer une Analyse en Composantes Principales (ACP). Nous obtenons la Figure 2 montrant le pourcentage de variances expliquées selon les composantes principales :

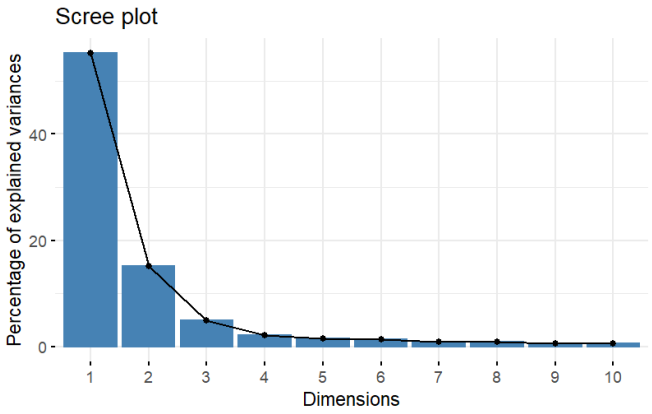

Figure 2: Pourcentage de variances expliquées en fonction des composantes principales

Ainsi, les 50 premières composantes principales expliquent environ 91.04% de la variance. Nous décidons donc de garder les 50 premières composantes principales. Nous aurions également pu appliquer une validation croisée pour obtenir le nombre de composantes principales optimal. Cependant, étant donné qu'augmenter de 50 composantes de plus nous permet de gagner environ 3.5% et qu'en retirant un trop grand nombre de composantes, les performances sont réduites de manière trop drastique, nous décidons de garder ces 50 premières composantes principales.

Entraînement des modèles

Nous appliquons les algorithmes de classification suivants : Naive Bayes (NB), K plus proches voisins (KNN), Analyse Discriminante Linéaire (LDA), Analyse Discriminante Quadratique (QDA), Analyse Discriminante Factorielle (FDA), Analyse Discriminante Régularisée (RDA), Régression Logistique (RL), Ridge, Lasso et Elastic Net, Bagging, Forêt Aléatoire (RF), Modèle de Mélange Gaussien (GMM), Mixture Discriminant Analysis (MDA) et Séparateur à Vaste Marge (SVM) avec trois noyaux différents (linéaire, polynomial et à fonctions de base radiale).

Pour chaque modèle, nous réalisons une validation croisée à 10 plis. Pour certains classifieurs nécessitant de spécifier un hyperparamètre, nous utilisons une validation croisée imbriquée pour obtenir l'hyperparamètre optimal. Le code suivant décrit la manière dont les validations croisées ont été réalisées :

```
library(caret)
CV <- rep(0,10)
#Creating folds
fold <- unname(createFolds(data$y, k=10))
for(i in (1:10)){
  #Training data
  train_data <- data[-fold[[i]], ]
  #Creating test data
```

```
  test_data <- data[fold[[i]], ]

  X.train <- train_data[,1:50]
  y.train <- train_data[, 51]
  X.test <- as.matrix(test_data[,1:50])
  y.test <- test_data[, 51]

  model <- classifieur(X.train, y.train)
  pred <- predict(model, X.test)

  CV[i] <- 1-sum(diag(table(y.test,pred)))/nrow(X.test)

  # errors est une matrice (10, n) avec n le nombre de classifieurs
  errors[i, c("Classifieur")] <- CV[i]
}
```

Nous résumons dans la Table 1 le(s) hyperparamètre(s) pour chaque modèle. Les hyperparamètres pour Ridge, Lasso et Elastic Net n'ont pas nécessité d'être déterminés avec une validation croisée imbriquée mais directement avec la fonction `cv.glmnet`. Quant au RDA, ils ont été obtenus avec la méthode `train` de la librairie `caret`. Ainsi, nous écrivons la médiane des hyperparamètres obtenus lors des validations croisées à 10 plis. Quant au SVM, son hyperparamètre a été obtenu en faisant une validation croisée parmi les valeurs de C suivantes : 0.001,0.01,0.1,1,10,100,1000,10^4.

Table 1: Valeur des hyperparamètres optimaux pour chaque modèle

Modèle	KNN	Ridge	Lasso	Elastic Net	RF	SVM-rbf	RDA
Hyperparamètre	$K = 22$	$\bar{\lambda} = 0.033$	$\bar{\lambda} = 0.00086$	$\bar{\lambda} = 0.00660$, $\bar{\alpha} = 1$	$\lambda = 20$	$C = 1$	$\bar{\lambda} = 0.83$, $\bar{\alpha} = 0.29$

Pour les GMM, nous avons utilisé la librairie `Mclust` et cette dernière a sélectionné le modèle "VVE" (ellipsoïdal, à volume et forme variants et orientation égale) à 6 composants comme étant optimal en utilisant le critère BIC. Par conséquent, nous avons utilisé ce même modèle pour entraîner notre MDA. Pour cette dernière, nous avons différencié le cas où les modèles doivent avoir un nombre de mélanges de composants et des structures de covariance (eg. des modèles sur les matrices de covariance) différents pour chaque classe (MDA) du cas où un seul composant est considéré avec la même structure de covariance pour toutes les classes (MDA_EDDA) à la manière du LDA, grâce à l'argument `modelType`. En utilisant la librairie `cvMclustDA`, une validation croisée à 10 plis a pu être réalisée directement, c'est pourquoi il n'y a pas d'intervalles de confiance sur la Figure 3.

```
# CV MDA with VVE
phonemes_gmm_VVE <- MclustDA(phonemes_pc[,1:50], phonemes_pc$y, modelNames="VVE")
cv <- cvMclustDA(phonemes_gmm_VVE)
errors[, c("MDA")] <- 1 - sum(diag(table(cv$classification, factor(phonemes$y))))/nrow(phonemes_pc)

# CV MDA with VVE and EDDA (LDA)
phonemes_gmm_VVE_EDDA <- MclustDA(phonemes_pc[,1:50], phonemes_pc$y, modelType = "EDDA",
                                  modelNames="VVE")
cv <- cvMclustDA(phonemes_gmm_VVE_EDDA)
errors[, c("MDA_EDDA")] <- 1 - sum(diag(table(cv$classification, factor(phonemes$y))))/
  nrow(phonemes_pc)
```

Résultats

Après avoir testé tous les modèles, nous obtenons la Figure 3 résumant le pourcentage d'erreur selon chaque modèle. Nous constatons que les modèles sont globalement performants avec des écart-types relativement faibles; le pire modèle ayant une erreur moyenne de moins de 15%.

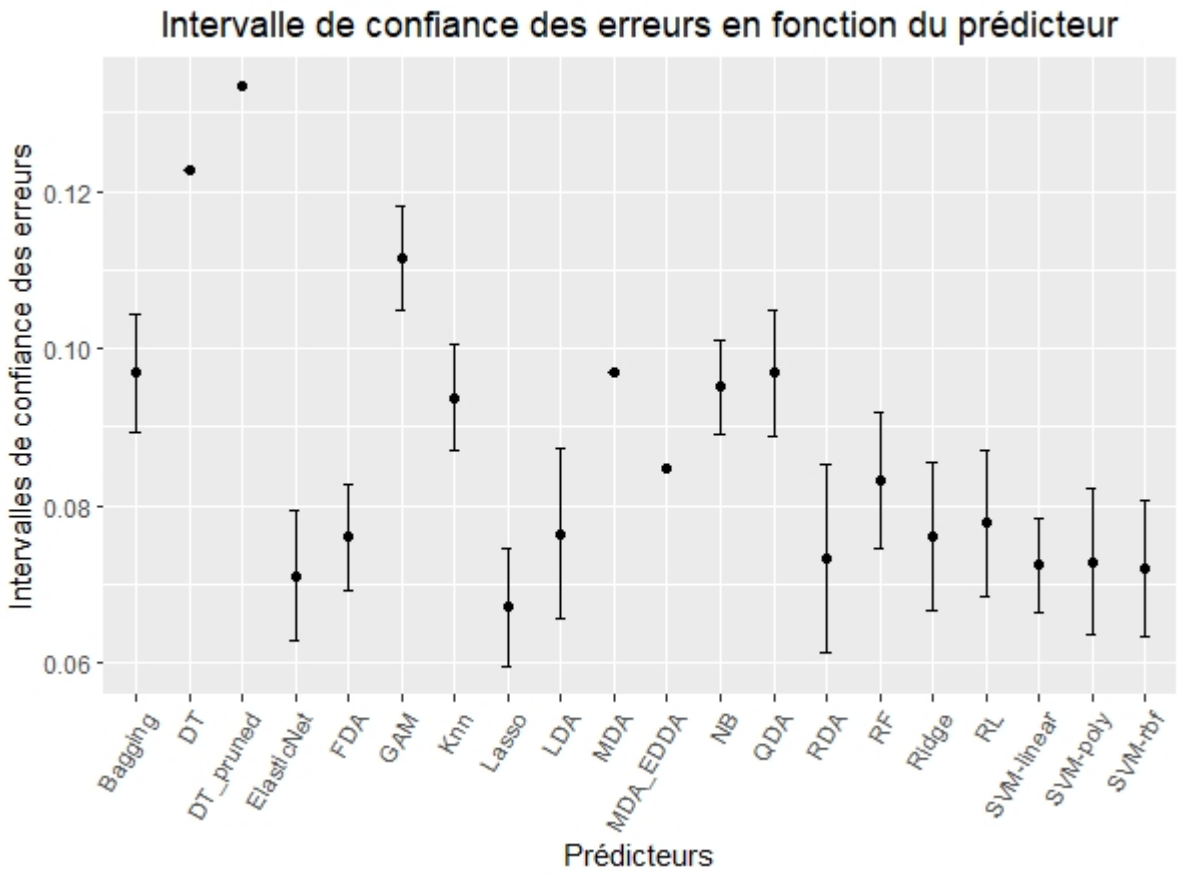

Figure 3: Résultats des modèles pour le dataset `Phonèmes`

Nous constatons que les modèles les moins performants à l'apprentissage sont l'arbre de décision (DT) et l'arbre avec élagage (DT_pruned). Cependant, en utilisant les méthodes ensemblistes avec le Bagging et les forêts aléatoires, nous parvenons à améliorer les résultats. Quant aux GAM, l'apprentissage n'a été réalisé qu'avec des splines sur les trois premières composantes principales, ce qui explique en partie des résultats moins bons.

Les méthodes basées sur l'analyse discriminante donnent globalement de bons résultats. En supposant que les matrices de covariance sont égales pour chaque classe, on obtient de meilleurs résultats. C'est pourquoi LDA, FDA et RDA ont de meilleures performances que QDA. En particulier, RDA est le plus performant, ce qui est cohérent au vu des hyperparamètres obtenus par validation croisée. En effet, on a obtenu des valeurs de λ proches de 1 tandis que celles de γ étaient assez faibles. Par conséquent, les matrices de covariance régularisées sont similaires aux estimations des matrices de covariance égales, telles qu'utilisées par la LDA.

Le SVM avec différents noyaux donnent de très bons résultats. Les méthodes avec régularisation sont encore meilleures, notamment avec le lasso. Toutefois, comme évoqué dans la sélection de variables, il faut être vigilant avec ces méthodes car elles ne semblent pas être efficaces avec notre jeu de données.

Ainsi, en conclusion, nous choisissons le **insert meilleur modèle**.

Letter recognition

Analyse exploratoire

Letter recognition est aussi un problème de classification puisqu'il s'agit de prédire une lettre parmi les lettres de l'alphabet. Les fréquences des lettres sont assez bien réparties.

Bike rental

Analyse exploratoire

Bike rental est cette fois un problème de régression dans lequel il faut prédire le nombre de locations de vélos sur l'année 2011 selon les conditions environnementales et saisonnières. Il y a à la fois des variables qualitatives et quantitatives. Tout d'abord, nous observons que la variable `yr` vaut toujours 0 peut importe l'observation, donc toutes les données ont été recueillies sur l'année 2011 et nous décidons alors d'enlever cette variable. La variable `instant` correspond simplement aux index relevés. Il se peut que les données d'autres années ne soient pas indexées de la même manière, c'est pourquoi nous décidons de la retirer également.

Commentaires sur dteday, et sur la normalité des résidus.