Projet SY19 - Régression et classification

Tom Bourg, Guillaume Sabbagh, Jiali Xu

06/01/2021

1 Phoneme

1.1 Analyse exploratoire

Dans cette première partie, il s'agit d'une classification des phonemes. Le jeu de données contient 2250 observations avec 256 préditeurs et une seule variables de réponse. On observe que la classification est en 5 classes et tous les préditeurs sont quantitatives.

```
##  aa  ao dcl  iy  sh
## 343 555 357 560 435
```

On remarque ici que le nombre d'observation n'est pas balancé parmi les classes.

1.2 Analyse de Composantes Principales

On s'intéresse à une analyse de composantes principales afin de réduire le nombre de préditeurs sachant que 256 est assez nombreux. A l'aide de la fonction *prcomp()*, on trace la proportion de variance expliquée cumulée en fonction du nombre de composante dans la figure 1 dessous.

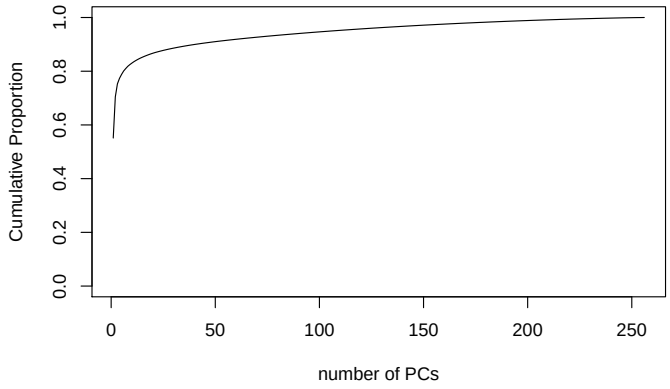

FIGURE 1 – Proportion de variance expliquée cumulée

On constate que les 107 premières composantes expliquent 95% de variance. Du fait que pour les réseaux neurones, le nombre de ports d'entrée de chaque couche est préférable d'être exponentiel de 2, on choisit ici 128 composantes pour les traitements suivants.

1.3 Apprentissage de modèles

1.3.1 LDA, QDA et Naive-Bayes

Le package **MASS** nous propose des fonctions pour LDA et QDA et le pakcage **naivebayes** pour un modèle Naive-Bayes. On observe que le taux d'erreur du modèle QDA est beaucoup plus haut que les deux autres. De ce fait, on constate que la matrice de covariance pour chaque classe reste presque identique.

1.3.2 Regression Logistique

Afin de réaliser une regression logistique pour une classification multinomiale, on utilise le modèle *multinom()* dans le package **nnet**. A partir de cette méthode, on réalise également un modèle additive généralisé en ajoutant un *natural cubic spline*.

Ensuite, on teste la regression ridge et lasso. Le package **glmnet** nous permet de trouver *lambda* optimisé en faisant une validation croisée. Le modèle lasso nous donne un meilleur résultat.

1.3.3 Arbres de regression

Pour les méthodes basées sur arbre, on a essayé une arbre binaire de décision *rpart()* du package **rpart** et son élagage en choisissant une valeur **CP** qui minimise l'erreur de validation croisée. On a également réalisé un modèle **Bagging** et **RandomForest** en appliquant différentes valeurs de *mtry*.

1.3.4 SVM

La fonction *ksvm* du package **kernlab** est utilisée pour tester avec deux noyaux différents. Pour chaque noyau, on fait d'abord une validation croissée pour trouve une valeur **C** optimisée parmi un ensemble [0.001, 0.01, 0.1, 1, 10, 100, 1000, 10e4].

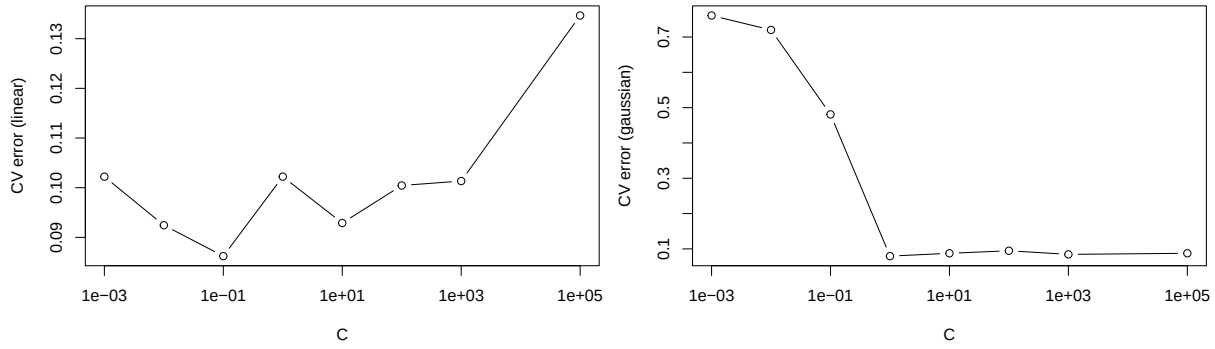

FIGURE 2 – Performance de modèles

1.3.5 Réseaux neurones

1.3.5.1 Réseau neurone artificiel On s'intéresse d'abord au réseau neurone artificiel avec deux couches. Sachant que nous sommes sur un problematique de classification, la fonction d'activation pour la couche de sortie est donc *softmax*. On paramètre *batch_size* à 128 et *epoch* à 30.

1.3.5.2 Réseau neurone profond régularisé Un réseau neurone profond est un réseau neurone artificiel avec plusieurs couches. Ici, on réalise un réseau neurone profond avec 4 *layer_dense* et 2 *layer_dropout*. On fixe le coefficient à 0.5 pour les couche de régulation afin d'éviter un surapprentissage. On applique une fonction d'activation *relu* pour tous les couches sauf celui de sortie. Les hyperparamètres à optimiser sont donc *units* des deux couches au milieu. Après des essais en utilisant une validation croisée, on n'observe pas une amélioration très évidente pour les différentes valeurs.

1.3.5.3 Réseau neurone convolutionnel Enfin, on effectue un essai sur un réseau neurone convolutionnel. Pour ce faire, il nous faut d'abord convertir les données en matrice 3D. Dans ce cas, la dimension de matrice de notre données est 2250*128 (après extraction de caractéristique). On choisit de la convertir en dimension 2250*16*8. La structure de modèle est comme ci-dessous.

```
## Model: "sequential"
## _________________________________________________________________________
## Layer (type)                 Output Shape              Param #
## =========================================================================
## conv1d (Conv1D)              (None, 14, 16)            400
## _________________________________________________________________________
```

```
## average_pooling1d (AveragePooling1D (None, 7, 16)                0
## ________________________________________________________________________________
## conv1d_1 (Conv1D)                   (None, 5, 32)                1568
## ________________________________________________________________________________
## average_pooling1d_1 (AveragePooling (None, 2, 32)                0
## ________________________________________________________________________________
## dropout (Dropout)                   (None, 2, 32)                0
## ________________________________________________________________________________
## flatten (Flatten)                   (None, 64)                   0
## ________________________________________________________________________________
## dense (Dense)                       (None, 128)                  8320
## ________________________________________________________________________________
## dropout_1 (Dropout)                 (None, 128)                  0
## ________________________________________________________________________________
## dense_1 (Dense)                     (None, 5)                    645
## ================================================================================
## Total params: 10,933
## Trainable params: 10,933
## Non-trainable params: 0
## ________________________________________________________________________________
```

1.4 Comparaison de modèle

A la fin, on fait une comparaison de tous les modèles réalisés. Voir les performance sur figure 3 et 4

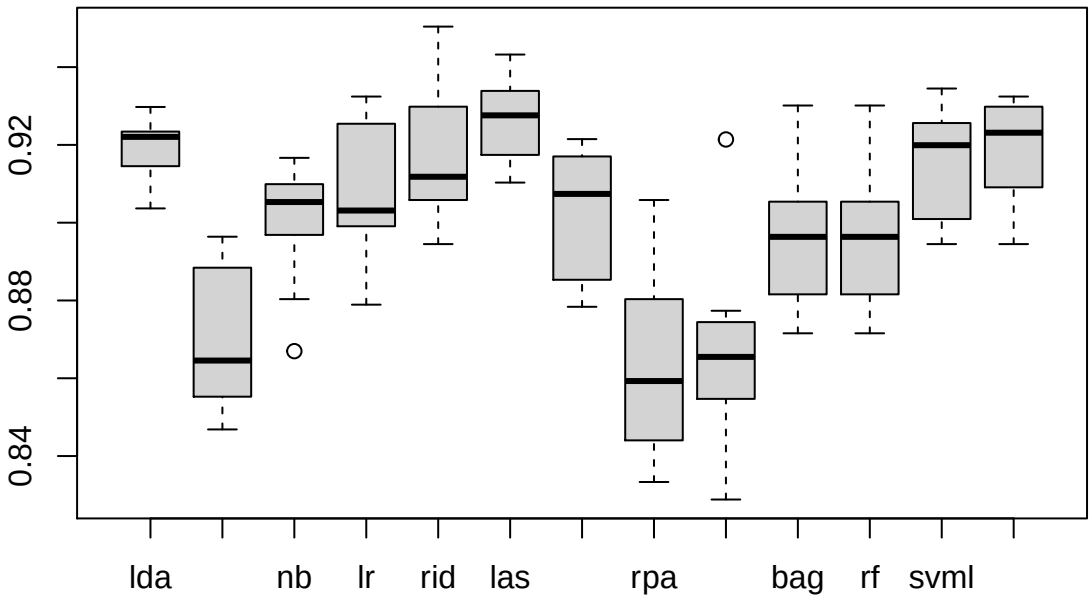

FIGURE 3 – Performance de modèles

On constate que le modèle réseau neurone profond nous donne une précision la plus haute et une variance plus petite.

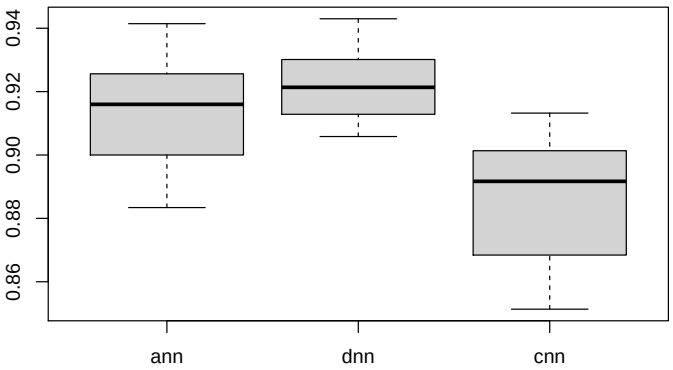

FIGURE 4 – Performance de réseau neurone