Examen TP : régression pénalisée

16 septembre 2025, de 14h à 16h15

MSP *2025-2026*

Quelques directives :

Le TP peut être réalisé en binôme ou seul. Idéalement en Markdown sinon un script R avec une copie de chaque sortie en commentaire (précédée d'un #). Il est indispensable d'inclure les sorties de votre code dans votre document. L'absence de la sortie de votre code sera considérée comme une réponse incomplète.

Votre copie doit être en un seul fichier prenom_nom.pdf ou prenom1_nom1_prenom2_nom2.pdf pour un binôme à envoyer à mohammed.sedkiATgustaveroussy.fr et mohammed.sedkiATinserm.fr (par sécurité).

Penser à préparer votre copie à envoyer quelques minutes avant la fin du TP, idéalement au fur et à mesure du déroulement du TP.

Le jeu de données concrete :

Le jeu de données concrete conçu pour étudier la résistance du béton en fonction de sa composition, provient de la plateforme UCI.

Lecture et découverte du jeu de données

```
require(caret)
require(glmnet)
library(tidyverse)
load("concrete_data.rda")
concrete %>% glimpse()

set.seed(21)
```

Afin d'ajuster un modèle qui explique la variable compressive_strength en fonction des autres variables explicatives du jeu de données, nous allons tester toutes les familles de modèles linéaires multiples pénalisés par elasticnet juqu'à la famille de modèles allant jusqu'à l'interaction d'ordre 5 des variables explicatives pénalisés par elasticnet. Pour déterminer l'ordre d'interaction des variables à retenir, nous allons séparer le jeu de données en apprentissage et test de tailles 75% et 25% de la taille du jeu de données de départ concrete.

Q1- Partager le jeu de données concrete en jeux de données apprentissage-test concrete_tr et concrete_te de tailles 75% et 25% respectivement.

Afin de prendre en compte les interactions allant jusqu'à un certain ordre, nous pouvons tester des formules à l'aide de la fonction model.matrix. Examinons les deux formules suivantes

```
model.matrix(compressive_strength ~ cement * age * fly_ash, concrete) %>% head()
model.matrix(compressive_strength ~ (cement + age + fly_ash)^3, concrete) %>% head()
```

Q2- Déduire une manière efficace pour configurer une formule qui prend en compte toutes les interactions de toutes les variables explicatives allant jusqu'à l'ordre 5.

Dans la suite, nous allons faire appel à la librairie caret pour implémenter une recherche des meilleurs hyperparamètres λ et α de pénalité elasticnet

```
fit_enet = train(....., # formule à compléter
                 data = ....., # jeu de données d'apprentissage
                 method = "glmnet", # famille de modèles ici glmnet
                 tuneGrid = ....., # grille d'hyperparamètres à compléter
                 metric = "RMSE", # métrique  de la validation croisée
                 preProcess = c("center", "scale"), # prétraitement des X
                 trControl = .....) # à déclarer à l'aide de la fonction trainControl
```

Q3- Pour chaque ordre d'interaction k allant de 1 à 5

- Utiliser la fonction model.matrix pour construire la matrice de design X_k
- Faire un appel à la fonction glmnet pour ajuster un modèle LASSO sur y et X_k pour calculer le chemin de régularisation lambda_k (grille de λ).
- Construire la grille d'hyperparamètres enet_grid_k composée d'une colonne lambda correspondant à lambdas_k et une colonne alpha correspondant à une séquence allant de 0.1 à 0.9 par pas de 0.1. **Attention** : pour avoir une exploration optimale, il faut que la grille enet_grid_k contienne tous les couples de valeurs λ et α qu'on peut former à partir des deux séquences.
- Déclarer ctrl à l'aide de la fonction trainControl correspondant à une validation croisée à 5 folds répétée 5 fois. (Cette déclaration est la même à toutes les valeurs de k donc à faire qu'une seule fois.)
- Utiliser la fonction train pour ajuster une famille de modèles linéaires allant juqu'aux interactions d'ordre k et stocker le résultat dans fit_enet_k.
- Calculer le vecteur pred_k de prédiction du jeu de données test concrete_te à partir de fit_enet_k.
- Calculer le RMSE de prédiction du jeu de données test concrete_te.

Q4- Peut-on conclure à un ordre d'interaction optimal ?

Q5- Peut-on forcer les grilles lambdas_1 à lambdas_5 à avoir la même taille ? Si oui, Comment ?

Q6- Peut-on refaire toutes les étapes de la question 3 en utilisant une spline cubique naturelle de degré de liberté 4 sur la variable age en interaction avec les variables cemen, water, superplasticizer.

Q7 (question bonus)- Peut-on paralléliser les calculs qui se déroulent durant les appels à la fonction train à l'aide de la librairie doParallel ? Tester les temps de calcul avec une parallélisation avec 10 cpus. Consulter l'aide de la fonction registerDoParallel et utiliser la fonction proc.time().

Jeu de données wallabies

Le jeu wallabies est disponible via la librairie mplot

```
data("wallabies", package = "mplot")
```

Q8- Utiliser la fonction bestglm de la librairie du même nom pour faire une sélection de modèles exhaustive à l'aide des critères AIC et BIC.

Q9- Proposer une démarche de sélection de variables avec interaction allant jusqu'à l'ordre *5*. L'objectif ici est d'explorer les interactions de variables qui peuvent expliquer la présence de l'espèce étudiée d'où l'absence de partition apprentissage-test. Vous pouvez utiliser ce que vous avez appris des questions précédentes ou une toute autre démarche.