# TP 4 Logistic Regression

*Mamadou Lamine DIAMBAN*

*15 Novembre 2019*

## Contexte

Nous cherchons à classer des observations en deux classes: $\mathcal{O} = \{-1, +1\}$ par régression logistique dont la fonction de perte logistique s'écrit :

$$\min_{x \in \mathbb{R}^d} f(x) := \frac{1}{m} \sum_{i=1}^{m} \log(1 + \exp(-b_i \langle a_i, x \rangle)) + \lambda_1 \|x\|_1 + \frac{\lambda_2}{2} \|x\|_2^2.$$

où:
– $a_i$ : est une observation et $b_i$ la classe correspondate.
– les deux derniers termes s'écrivent sont une régularisation de type elastic-net.

Sous certaines hypothèses évoquées ici, $x^\star = arg\min f(x)$ maximise la vraisemblance régularisées des labels sachant les observations. Ainsi pour une nouvelle observation $a$ de classe $d$ (inconnue),

$$p_1(a) = \mathbb{P}[d \in class + 1] = \frac{1}{1 + \exp(-\langle a; x^\star \rangle)}$$

Ainsi, à partir de $a$ , si $p_1(a)$ est proche de 1 , il est raisonnable de décider que l'example arrivant appartient à la classe 1 ; et l'inverse si $p(a)$ est proche de 0 . Entre les deux, l'appréciation est laissée à l'analyste selon l'application.

## Données

```
library(CVXR)
library(kableExtra)
```

```
# Importation des données de student
st <- read.csv2("../Data/student-mat.csv", header = TRUE)
```

```
# Retypage de toutes les variables en variable quantitative
student <- as.data.frame(sapply(st, as.numeric))
```

```
y <- ifelse(student$G3 < 12, -1, 1)

# On crée un jeux de données à partir de student en enlevant G3
X <- as.matrix(subset(student, select = -G3))
```

### 1. Implementer une fonction de minimisation de la perte logistique régularisée en utilisant CVXR

```
perte_logist_reg <- function(X,y,lambda1,lambda2){
    m <- ncol(X)
    n <- nrow(X)
    x = Variable(m)

    f <- (1/n)*(sum(logistic(-y*(X%*%x))))+lambda1*cvxr_norm(x,1)+(lambda2/2)*cvxr_norm(x,2)^2

    objective <- Minimize(f)
    problem <- Problem(objective)
    result <- solve(problem)

    # On cache les sorties pour ne pas toujours avoir l'affichage "optimal"
    invisible(list(status =result$status, coef = result$getValue(x)))
}
```

```
# Les valeurs de la fonction sont stockées dans la variable res_logist
res_logist <- perte_logist_reg(X, y, 0.1, 0.06)
res_logist$status
```

```
## [1] "optimal"
```

```
coef_perte <- res_logist$coef

# On affiche que les valeurs estimés qui sont supérieurs à 10e-4 ou inférieures à -10e-4
val <- as.matrix(coef_perte[coef_perte > 0.0001 | coef_perte < -0.0001])
rownames(val) <- colnames(X)[c(which(coef_perte > 0.0001 | coef_perte < -0.0001))]
kable(val, booktab = TRUE)
```

| | |
|---|---|
| age | -0.4653711 |
| absences | -0.0097520 |
| G1 | 0.2436522 |
| G2 | 0.4261646 |

En prenant $\lambda_1 = 0.1$ et $\lambda_2 = 0.06$, nous avons une solution optimale et nous obtenons le tableau des coefficients avec les colonnes correspondantes. On peut voir que la variable G3 est assez bien corrélée positivement avec la variable G2 (0.42), assez bien corrélée avec G1(0.24) et est corrélée négativement avec la variable age(-0.46); Nous pouvons donc en déduire que plus l'étudiant a de bonnes notes en G1 et G2 plus il a des chances d'avoir de bonnes notes en G3 et de passer en classe supérieure; et plus il est jeune plus il a de bonnes notes en G3 aussi;

## 2. Écrire une fonction de prédiction des labels pour un nouveau vecteur à partir de la solution obtenue par la fonction précédente

```
prediction <- function(labs, coef){
    1 / (1 + exp(- (labs %*% coef)))
}
```

```
set.seed(1)
train_index <- sample(nrow(X), .8*nrow(X))

# On crée des jeux d'apprentissage et de teste
X_train <- X[train_index, ]; X_test <- X[-train_index,]
y_train <- y[train_index]; y_test <- y[-train_index]
```

```r
# On estime les lables sur les données d'apprentissage
perte_logist_reg(X_train, y_train, .1, .1)$status
```

```
## [1] "optimal"
```

```r
coef_perte_train <- perte_logist_reg(X_train, y_train, 0.1, 0.06)$coef
res_pred_train <- prediction(X_train, coef_perte_train)
label_train <- ifelse(res_pred_train < .5, -1, 1)

# On estime les labels sur les données de teste
perte_logist_reg(X_test, y_test, .1, .1)$status
```

```
## [1] "optimal"
```

```r
coef_perte_test <- perte_logist_reg(X_test, y_test, 0.1, 0.06)$coef
res_pred_test <- prediction(X_test, coef_perte_test)
label_test <- ifelse(res_pred_test < .5, -1, 1)

# On crée une fonction pour calculer le taux de bon classement
taux <- function(y, label){
    tab <- table(y, label)
    round((tab[1,1] + tab[2,2]) / length(y), 3)
}

taux_class1 <- cbind(train = taux(y_train, label_train),
                     test = taux(y_test, label_test))

kable(taux_class1, booktab = TRUE)
```

| train | test |
|---|---|
| 0.927 | 0.899 |

La fonction nous permet de prédire la classe des étudiants (-1 et 1). Nous avons une probabilité de l'échantillon Test(0.89) qui légérement différente de l'échantillon Train(0.92), mais dans l'ensemble nous obtenons un bon modèle.

### 3. Motivez un choix pour les paramètres du modèle $\lambda_1, \lambda_2$ et de la prédiction (seuil entre +1 et -1)

```
lambda1 <- lambda2 <- seq(0.01,.1, by = .01)

taux_class1 <- matrix(ncol = 4, nrow = 1)

for(i in 1:length(lambda1)){
    for(j in 1:length(lambda2)){
        res_train <- perte_logist_reg(X_train, y_train, i, j)
        coef_train <- res_train$coef
        res_pred_train<- prediction(X_train, coef_train)

        res_test <- perte_logist_reg(X_test, y_test, i, j)
        coef_test <- res_test$coef
        res_pred_test <- prediction(X_test, coef_test)

        if(res_train$status=="optimal" & res_test$status=="optimal"){
            label_train <- ifelse(res_pred_train < .5, -1, 1)
            label_test <- ifelse(res_pred_test < .5, -1, 1)

            prop_label_train <- sum((y_train - label_train)==0)/length(y_train)
            prop_label_test <- sum((y_test - label_test)==0)/length(y_test)
            prop <- cbind(lambda1[i],lambda2[j], prop_label_train, prop_label_test)
            if(prop_label_train > .9){
                taux_class1 <- rbind(taux_class1, prop)
            }
        }
    }
}
colnames(taux_class1) <- c("lambda 1", "lambda 2", "prop train", "prop test")
#kable(taux_class1[-1,], booktab = TRUE)
taux_class1[-1,]
```

```
##      lambda 1 lambda 2 prop train prop test
## [1,]     0.04     0.02  0.9303797 0.6835443
## [2,]     0.05     0.09  0.9240506 0.6835443
## [3,]     0.06     0.02  0.9082278 0.6835443
## [4,]     0.07     0.03  0.9208861 0.6835443
## [5,]     0.10     0.06  0.9240506 0.9113924
```

En faisant les valeurs de $\lambda_1$ et $\lambda_2$ de 0.01 à 0.1 par séquence 0.01, on observe que le bon pourcentage de prédiction est atteint pour $\lambda_1 = 0.1$ et $\lambda_2 = 0.06$