ACT6100	Analyse de données
H2019	Série 7

Agrégation de modèles

1. Une base de données définie dans $\mathbb{R}^2 \times \{A, B\}$ est présentée à la Table 1. On considère comme *weak learner* un arbre avec un seul noeud, c'est-à-dire qu'à chaque étape, la règle de décision est $x_j < c$ ou $x_j > c$, $j = 1$ ou 2, où la valeur de c est sélectionnée afin de minimiser l'erreur de classification sur la base de données complète. On utilise une version de l'algorithme *AdaBoost* sans *bootstrap*, c'est-à-dire que $\mathcal{D}_t$ est la base de données complète, $\forall t$.

i	X_{i1}	X_{i2}	Y_i	p_{i1}
1	-3	-1	-1	1/8
2	-3	1	-1	1/8
3	3	-1	-1	1/8
4	3	1	-1	1/8
5	-1	-1	1	1/8
6	-1	1	1	1/8
7	1	-1	1	1/8
8	1	1	1	1/8

TABLE 1 – Base de données.

   (a) Quelle sera la première règle de décision (r_1) ?

   (b) En posant $T = 2$, déterminer $R(\mathbf{X}_i)$.

2. On considère la base de données *dataSPAM* disponible sur le site *Moodle* du cours [1]. Elle contient une variable réponse **V58** binaire et 57 variables explicatives **V1** à **V57**.

   (a) Diviser aléatoirement la base de données en une base de données d'entrainement (50 %) et une base de données de test (50 %).

   (b) Ajuster un modèle des K plus proches voisins en choisissant la valeur de K qui minimise le taux de mauvais classement sur la base de données de test.

   (c) En utilisant la base de données d'entrainement, ajuster un modèle d'arbres de décision. Calculer le taux de mauvais classement sur la base de données de test.

   (d) En utilisant la base de données d'entrainement, utiliser l'algorithme d'agrégation de modèles du *Random Forests* avec 2 500 arbres. Ajouter les arguments

   ```
   xtest = dataTEST[,-58]
   ```

   et

   ```
   ytest = as.factor(dataTEST[,58]
   ```

   à la fonction randomForest afin de calculer à chaque itération le taux de mauvais classement sur la base de données de test (dataTEST ici).

   (e) Utiliser le code présenté à la Figure 1 afin d'utiliser l'algorithme d'agrégation de modèles *AdaBoost*. Ce code utilise la fonction *gbm* de la librairie du même nom. Présenter les résultats obtenus aux différentes sous-questions sur le même graphique. Commenter.

1. La base de données provient du livre *The Elements of Statistical Learning*.

```
model <- gbm(Y~., data = dataTRAIN, distribution = "adaboost",
             n.trees = 2500, interaction.depth = 2, shrinkage = 0.05)

B <- 2500
errapp <- rep(0, B)
err_boost <- errapp
boucle <- seq(1, B, by = 50)
errtest <- rep(0, length(boucle))
k <- 0
for (i in boucle)
    {
    k<- k+1
    prev_test <- predict(model, newdata = dataTEST, n.trees = i)
    err_boost[k] <- sum(as.numeric(prev_test > 0) != dataTEST$Y)/nrow(dataTEST)
    }
```

FIGURE 1 – Code informatique.

3. On considère une version légèrement simplifiée de l'algorithme *AdaBoost* introduit dans les notes des cours. On s'intéresse à un problème de classification binaire (-1 ou 1) à partir d'une base de données $\mathcal{D} = \{Y_i; \mathbf{X}_i\}_{i=1,\ldots,n}$.
   1. Toutes les observations reçoivent un poids identique : $p_1(\mathbf{X}_i) = 1/n$, $i = 1, \ldots, n$.
   2. Pour l'étape $t = 1, \ldots, T$,
      2a. Apprendre une règle de classification r_t sur cette base de données en minimisant le taux de mauvaise classification, c'est-à-dire

$$r_t = \arg\min_{r \in \mathcal{R}} \frac{1}{n} \sum_{i: r(\mathbf{X}_i) \neq Y_i} p_t(\mathbf{X}_i),$$

      où $\mathcal{R}$ est l'espace des choix possibles pour la règle r.

      2c. Calculer l'erreur apparente

$$\epsilon_t = p_t\left(r_t(\mathbf{X}_i) \neq Y_i\right) = \sum_{i: r_t(\mathbf{X}_i) \neq Y_i} p_t(\mathbf{X}_i)$$

      pour $\mathcal{D}$ et évaluer $\alpha_t = 0.5 \ln((1 - \epsilon_t)/\epsilon_t)$.

      2d. Mettre à jour les poids :

$$p_{t+1}(\mathbf{X}_i) = \begin{cases} p_t(\mathbf{X}_i) e^{-\alpha_t}, & r_t(\mathbf{X}_i) = Y_i \\ p_t(\mathbf{X}_i) e^{+\alpha_t}, & r_t(\mathbf{X}_i) \neq Y_i. \end{cases}$$

   On va vérifier que la règle de classification à l'étape t peut s'écrire comme étant

$$r_t(\mathbf{X_i}) = r_{t-1}(\mathbf{X}_i) + \alpha_t R_t(\mathbf{X}_i),$$

   où

$$(\alpha_t, R_t) = \arg\min_{(\alpha, r) \in \mathbb{R} \times \mathcal{R}} \sum_{i=1}^{n} e^{-Y_i (r_{t-1}(\mathbf{X}_i) + \alpha r(\mathbf{X}_i))}.$$

(a) Vérifier que (α_t, R_t) peuvent être obtenus de façon équivalente en solutionnant

$$(\alpha_t, R_t) = \arg\min_{(\alpha,r)\in\mathbb{R}\times\mathcal{R}} \left(e^{\alpha} - e^{-\alpha}\right) \sum_{i:r_t(\mathbf{X}_i\neq Y_i} w_i^t + e^{-\beta} \sum_{i=1}^{n} w_i^t,$$

où w_i^t est un poids qui ne dépend ni de α, ni de r.

(b) Vérifier que l'optimisation peut se faire en deux étapes :

i. en supposant $\alpha > 0$ fixé, on obtient

$$R_t(\mathbf{X}_i) = \arg\min_{r\in\mathcal{R}} \sum_{i=1}^{n} w_i^t \mathbb{I}_{r(\mathbf{X}_i)\neq Y_i};$$

ii. en supposant r fixée on obtient

$$\beta_t = 0.5 \ln\left(\frac{\sum_{i:r(\mathbf{X}_i)=Y_i} w_i^t}{\sum_{i:r(\mathbf{X}_i)\neq Y_i} w_i^t}\right).$$

(c) Vérifier que cette procédure est équivalente à l'algorithme *Adaboost*.

4. En utilisant le code présenté à la Figure 2, générer un échantillon d'entrainement et un échantillon de validation. En utilisant la fonction *gbm* (voir Figure 1), ajuster un modèle

```
set.seed(1025)
n <- 100
X <- matrix(rnorm(n, 0, 1), ncol = 1)
U <- runif(n, 0, 1)
Y <- matrix((X <= 0) * (U <= 0.25) + (X > 0) * (U > 0.25), ncol = 1)
data <- data.frame(X = X, Y = Y)

m <- 400
Xvalid <- matrix(rnorm(m, 0, 1), ncol = 1)
Uvalid <- runif(m, 0, 1)
Yvalid <- matrix((Xvalid <= 0) * (Uvalid <= 0.25) + (Xvalid > 0)
                 * (Uvalid > 0.25), ncol = 1)
datavalid <- data.frame(X = Xvalid, Y = Yvalid)
```

FIGURE 2 – Code informatique.

avec un taux d'apprentissage élevé (par exemple 1) et un modèle avec un taux d'apprentissage faible (par exemple 0.01). Pour chacun des modèles, réaliser un graphique de la proportion de mauvaise classification calculée sur la base d'entrainement et de l'erreur calculée sur la base de validation en fonction du nombre d'itérations de l'algorithme. Commenter.