Stratégie d'échantillonnage pour l'apprentissage supervisé

Tutoriel Tanagra

1. Introduction

Ce tutoriel fait suite au support de cours consacré aux algorithmes d'échantillonnage. Nous nous intéressons en particulier aux stratégies d'échantillonnage pour la modélisation prédictive (Algorithmes d'échantillonnage).

L'idée est relativement simple. Modéliser à partir d'ensemble de données très volumineux nécessite des ressources (machines performantes et/ou organisées en cluster) et des compétences (maîtrise des technologies big data) qui ne sont pas toujours à notre portée. Plutôt que de s'enferrer dans des voies sans issues (chercher à accéder à des ressources de plus en plus onéreuses par exemple), pourquoi ne pas travailler à partir d'échantillons représentatifs pour construire les modèles prédictifs ? L'inférence statistique repose justement ce postulat : il est possible d'induire les caractéristiques inconnues d'une population à partir d'un échantillon qui en est issu (Wikipédia). La démarche sera d'autant plus efficace qu'il y a une forme de redondance dans les données. Plus elle sera forte, moins on a besoin d'observations pour généraliser sur le reste de la base.

La transposition au cadre de l'apprentissage supervisé semble séduisante. Mais elle n'est pas spontanée. Elle soulève en effet le problème crucial de l'identification la taille d'échantillon permettant d'assurer un niveau de performance équivalent à la modélisation sur la totalité de la base. Nous ne sommes pas dans un cadre paramétrique où les hypothèses de distributions statistiques nous permettraient de définir analytiquement le nombre d'observations requis pour respecter une marge d'erreur choisie à l'avance. Les approches développées sont forcément empiriques, dépendantes des caractéristiques des algorithmes de machine learning utilisées et des caractéristiques de la base traitée.

Dans ce document, nous étudions expérimentalement le comportement des deux stratégies décrites dans notre support de cours de référence. L'approche "random sampling" consiste à démarrer à partir d'une taille d'échantillon définie a priori, puis de l'augmenter graduellement tout en surveillant les performances en test (taux d'erreur). La méthode "windowing" procède du même principe mais cherche à sélectionner judicieusement les observations additionnelles à chaque étape pour améliorer la convergence.

Nos algorithmes et bases de référence seront respectivement l'analyse discriminante linéaire et les données WAVEFORM bien connus des data scientists. L'étude a été menée sous R. Mais le portage du code dans d'autres langages comme Python ne pose aucun problème conceptuel.

2. Cadre de l'expérimentation

Données WAVEFORM

La base WAVEFORM est composée de 21 variables prédictives et d'une variable cible à trois modalités.

Elle convient à notre étude car nous disposons d'un générateur de données. Nous pouvons moduler à discrétion les volumétries.

Ci-dessous, nous chargeons l'ensemble d'apprentissage et affichons la liste des variables.

```
dfTrain <- read.table("wave20k_train.txt",header=T,sep="\t",dec=".")
print(str(dfTrain))
```

```
## 'data.frame':    20000 obs. of  22 variables:
##  $ onde: Factor w/ 3 levels "A","B","C": 1 2 2 3 3 1 3 1 3 3 ...
##  $ v01 : num  1.39 0.1 -0.64 0.94 -0.03 1.12 -0.29 -0.72 1.01 2.58 ...
##  $ v02 : num  1.25 1.7 -1.15 -0.51 0.62 0.2 0.5 0.96 -0.46 0.5 ...
##  $ v03 : num  0.38 2.37 1.33 -1.24 -1.28 0.15 0.39 2.35 -0.28 0.89 ...
##  $ v04 : num  1.62 0.5 -0.35 -2.07 -0.44 -1.84 -0.71 2.23 -1.1 -0.12 ...
##  $ v05 : num  2.85 1.4 -0.17 0.23 -1.04 0.63 -1.05 2.97 -0.75 1.48 ...
##  $ v06 : num  4.1 1.56 2.84 -0.32 0.36 2.32 0.26 3.62 -0.8 1.27 ...
##  $ v07 : num  5.68 1.98 2.44 3.11 0.08 1.99 0.65 3.94 2.52 0.26 ...
##  $ v08 : num  2.98 4.46 3.71 2.3 0.22 0.22 -0.09 4.75 1.89 1.14 ...
##  $ v09 : num  3.03 3.65 4.87 3.33 2.87 0.75 3.02 1.82 2.75 0.56 ...
##  $ v10 : num  2.39 4.61 4.64 3.84 3.37 1.04 4.07 1.81 2.78 0.6 ...
##  $ v11 : num  0.63 6.91 4.56 6.97 3.75 0.93 4.26 1.51 4.37 1.82 ...
##  $ v12 : num  3.07 4.86 2.62 3.52 3.65 3.42 4.31 -0.77 3.31 3.63 ...
##  $ v13 : num  0.09 0.29 2.94 2.02 5.46 4.04 5.73 -0.02 3.98 3.04 ...
##  $ v14 : num  1.66 1.02 1.7 3.12 3.35 3.68 4.75 1.72 4.4 5.78 ...
##  $ v15 : num  1.25 3.74 2.57 1.55 5.14 4.73 5.33 0.65 3.88 5.13 ...
##  $ v16 : num  3.04 0.58 1.83 2.12 4.43 2.33 3.19 0.9 4.1 4.32 ...
##  $ v17 : num  1.33 1.87 -1.4 -0.21 0.8 3.03 0.78 0.49 2.31 2.31 ...
##  $ v18 : num  0.96 -0.27 -0.2 0.44 2.42 2.68 0.92 -0.49 1.38 3.68 ...
##  $ v19 : num  3.3 0.63 -0.15 -0.04 0.5 2.51 1.38 -0.55 -0.06 0.73 ...
##  $ v20 : num  0.68 1.19 0.02 0.12 2.18 -0.64 0.07 0.14 1.31 0.8 ...
##  $ v21 : num  -0.48 -0.18 0.71 -0.52 0.81 -0.58 1.39 0.69 -0.13 -0.81 ...
## NULL
```

"onde" est la variable cible, nous disposons de n_a = 20.000 observations.

```
print(dim(dfTrain))
```

```
## [1] 20000    22
```

Notre ensemble de test est composé de n_t = 80.000 observations, il servira à mesurer le taux d'erreur de nos différents modèles.

```
dfTest <- read.table("wave80k_test.txt",header=T,sep="\t",dec=".")
print(dim(dfTest))

## [1] 80000    22
```

Performances de référence

L'objectif de l'étude est de déterminer une taille d'échantillon qui permettrait d'atteindre une qualité prédictive équivalente au travail sur la totalité de la base.

Mesurons cette performance de référence en créant un premier modèle sur l'intégralité de l'ensemble d'apprentissage.

```
library(MASS)

#construction du modèle
modeleRef <- lda(onde ~ ., data = dfTrain)

#prédiction sur l'ensemble de test
pred <- predict(modeleRef, newdata = dfTest)

#taux d'erreur
err_ref <- mean(dfTest$onde != pred$class)
print(err_ref)

## [1] 0.1355
```

Notre objectif est d'utiliser d'une fraction de dfTrain aussi petite que possible permettant d'atteindre un taux d'erreur équivalent sur dfTest (err_ref = 0.1355).

3. Random sampling

L'approche "Random Sampling" consiste à modéliser à partir de sous-ensembles de dfTrain dont nous augmentons graduellement la taille. Nous ne sommes pas censés disposer de dfTest. Pour suivre l'évolution des performances, nous pouvons utiliser la fraction de dfTrain qui n'est pas utilisée à chaque étape. Sa taille diminue à chaque itération. C'est un inconvénient dont il faut être conscient.

Dans notre expérimentation, nous utilisons quand même l'échantillon test dfTest uniquement pour suivre l'amélioration réelle des performances consécutive à l'adjonction d'observations supplémentaires dans le sous-ensemble d'apprentissage.

Nous simplifions également le processus en spécifiant un nombre prédéfini d'itérations.

Dans un premier temps, nous initialisons les paramètres de l'étude :

```
#taille initiale et nombre d'observations additionnelles à chaque étape
n_add <- 200

#nombre d'étapes
n_steps <- 25
```

Trois vecteurs nous permettent de suivre le processus :

```
#taille de la base d'apprentissage
taille <- c()

#erreur sur la fraction non-utilisée de dfTrain
err_train <- c()

#erreur sur dfTest
err_test <- c()
```

Nous sommes parés pour lancer le processus :

```
#data frame courant pour la modélisation
dfCur <- data.frame()

#data frame formé par les observations qui ne sont pas dans dfCur
dfReste <- dfTrain

#boucler
for (i in 1:n_steps){
  #échantillonner dans les observations non sélectionnées
  #tirage sans remise
  id <- sample(1:nrow(dfReste),n_add,replace=FALSE)

  #data frame intermédiaire pour la modélisation
  dfCur <- rbind(dfCur,dfReste[id,])

  #data frame contenant les observations restantes
  dfReste <- dfReste[-id,]

  #modélisation
  modele <- lda(onde ~ ., data = dfCur)

  #taille d'échantillon utilisé
  taille <- c(taille,nrow(dfCur))

  #erreur sur le train restant
  err_train <- c(err_train,mean(dfReste$onde != predict(modele,dfReste)$class))

  #erreur sur l'échantillon test
  err_test <- c(err_test,mean(dfTest$onde != predict(modele,dfTest)$class))
}
```

Voyons le détail des résultats :

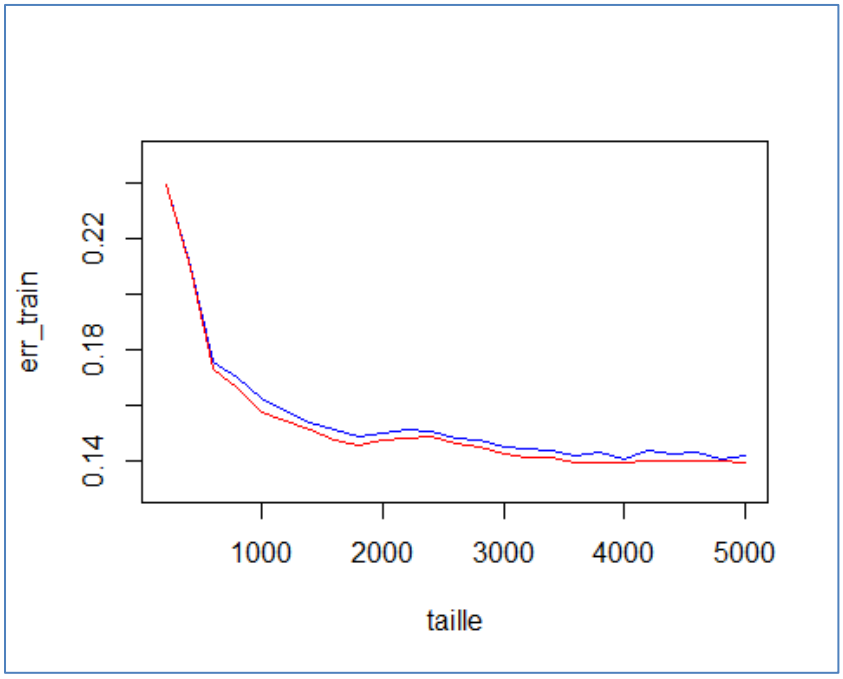
```r
print(cbind(taille,err_train,err_test))
```

```
##       taille err_train  err_test
##  [1,]    200 0.2395455 0.2390125
##  [2,]    400 0.2102041 0.2093125
##  [3,]    600 0.1756186 0.1730375
##  [4,]    800 0.1698437 0.1662875
##  [5,]   1000 0.1625263 0.1576875
##  [6,]   1200 0.1581915 0.1543750
##  [7,]   1400 0.1534409 0.1509625
##  [8,]   1600 0.1510326 0.1472375
##  [9,]   1800 0.1486264 0.1457750
## [10,]   2000 0.1496667 0.1472250
## [11,]   2200 0.1512360 0.1483250
## [12,]   2400 0.1503977 0.1487750
## [13,]   2600 0.1479885 0.1459750
## [14,]   2800 0.1472674 0.1451625
## [15,]   3000 0.1447059 0.1427000
## [16,]   3200 0.1441667 0.1414500
## [17,]   3400 0.1434337 0.1411500
## [18,]   3600 0.1420122 0.1393375
## [19,]   3800 0.1427778 0.1395375
## [20,]   4000 0.1408125 0.1394125
## [21,]   4200 0.1434810 0.1402375
## [22,]   4400 0.1426923 0.1401750
## [23,]   4600 0.1429221 0.1402000
## [24,]   4800 0.1403947 0.1399000
## [25,]   5000 0.1415333 0.1394875
```

Que nous pouvons représenter graphiquement :

```r
#erreur sur les fractions non utilisées de dfTrain
plot(taille,err_train,col="blue",type="l",ylim=c(0.13,0.25))

#erreur sur l'échantillon test à part
lines(taille,err_test,col="red")
```



Courbe rouge. Avec une taille d'échantillon de n = 5000 (pris parmi les 20000 de dfTrain), nous obtenons un niveau de performances (erreur = 0.1369) quasi-équivalent à celui obtenu avec un modèle exploitant l'intégralité de dfTrain (err_ref = 0.1355). Notons que dès n = 4000, le gain marginal à chaque adjonction d'observations est faible. S'arrêter à ce stade n'aurait pas été choquant.

Courbe bleue. En pratique, nous ne disposons que de la courbe bleue pour piloter le processus. A partir de 4000 observations, l'erreur est sur un plateau laissant à penser que nous avons d'ores et déjà exploité l'information "utile" pour la modélisation.

Courbes bleues et rouges sont plutôt cohérentes finalement. Dans les deux cas, pour cette base de données tout du moins et avec l'analyse discriminante linéaire, nous constatons qu'une taille d'échantillon inférieure à 1000 observations est à l'évidence insuffisante pour modéliser correctement les relations entre la cible et les variables explicatives.

4. Windowing

L'approche "windowing" s'appuie sur un principe similaire à "random sampling". A la différence que nous incluons dans la fraction d'apprentissage les individus mal classés à l'étape précédente (le nombre d'observations concernés n'est pas connu à l'avance). En concentrant les efforts sur les observations "difficiles" à chaque étape, nous espérons pouvoir converger plus rapidement vers la solution la plus performante.

Voyons ce qu'il en est en pratique.

Ici également, plusieurs initialisations sont nécessaires :

```
#taille initiale de la fenêtre
n_init <- 200

#taille de la base d'apprentissage
taille <- c()

#erreur sur la fraction non-utilisée de dfTrain
err_train <- c()

#erreur sur dfTest
err_test <- c()

#nombre d'individus mal classés
nb_misc <- c()

#échantillonnage
id <- sample(1:nrow(dfTrain),n_init,replace=FALSE)

#data frame courant pour la modélisation
dfCur <- dfTrain[id,]

#data frame formé par les observations qui ne sont pas dans dfCur
dfReste <- dfTrain[-id,]
```

Nous pouvons boucler jusqu'à ce que tous les individus de dfTRain soient : soit inclus dans la fraction d'apprentissage, soit correctement classés.

```
#boucler
while (TRUE){

  #taille de la base
  taille <- c(taille,nrow(dfCur))

  #modélisation
  modele <- lda(onde ~ ., data = dfCur)

  #erreur sur l'échantillon test
  err_test <- c(err_test,mean(dfTest$onde != predict(modele,dfTest)$class))

  #prédiction sur le reste de train
  predReste <- predict(modele,dfReste)$class

  #erreur sur la fraction non utilisée de dfTRain
  err_train <- c(err_train,mean(dfReste$onde != predReste))

  #identification des individus mal classés sur le reste de dfTrain
  misclassified <- which(predReste != dfReste$onde)

  #nombre correspondant
  n_misc <- length(misclassified)
```

```
  #stockage
  nb_misc <- c(nb_misc,n_misc)

  #si amélioration possible
  if (n_misc > 0){
    #obs. pour l'apprentissage
    dfCur <- rbind(dfCur,dfReste[misclassified,])

    #obs. pour la vérififcation
    dfReste <- dfReste[-misclassified,]

  } else {
    #tous les individus restants sont bien classés - arrêt
    break
  }

}
```

Affichons le détail des résultats :

```
print(cbind(taille,err_train,nb_misc,err_test))
```

```
##      taille    err_train nb_misc  err_test
## [1,]    200 0.2321212121    4596 0.2296250
## [2,]   4796 0.4679689555    7115 0.3928500
## [3,]  11911 0.1097787118     888 0.2049625
## [4,]  12799 0.0004166088       3 0.1582375
## [5,]  12802 0.0000000000       0 0.1580625
```

Le processus s'est arrêté parce que tous les individus non-inclus dans l'apprentissage de dfTrain ont été correctement classés. La convergence est très rapide avec 4 itérations seulement.

En revanche, et c'est une déception, l'erreur en test finalement obtenue (0.1567) avec 12152 observations n'est pas à la hauteur de la référence (err_ref = 0.1355). Parfois, en se focalisant sur les individus à problèmes (bruités ou outliers), l'approche "windowing" peut se fourvoyer vers des solutions non-optimales.

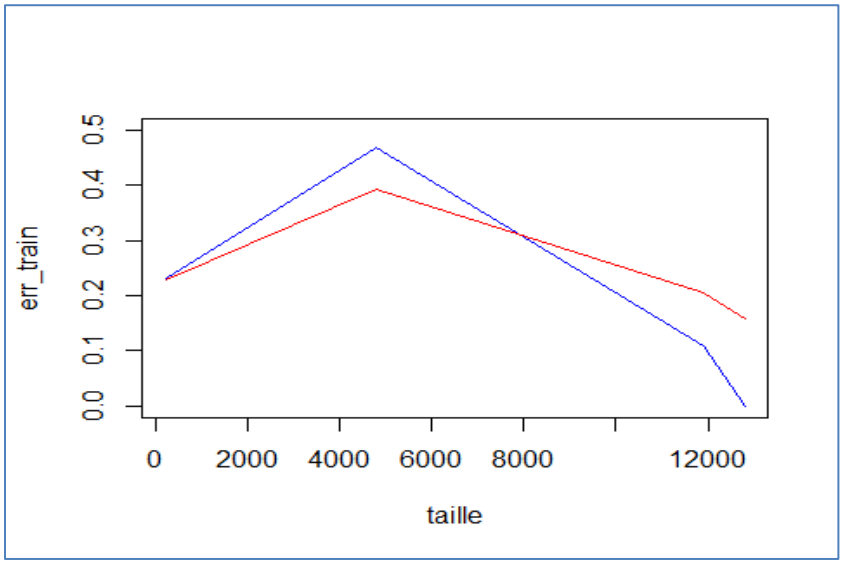
La seconde itération où l'on note une forte dégradation de l'erreur est assez symptomatique de cet écueil d'ailleurs. L'intégration d'un nombre élevés d'individus mal classés (4486) au détriment de l'échantillon de départ (200 obs.) désoriente la modélisation. Tendance corrigée à l'étape suivante où les 7032 individus supplémentaires intégrés dans l'apprentissage améliore significativement les résultats.

Des expérimentations menées sur d'autres bases confirment ce comportement caractéristique.

Ici également, il est possible de représenter graphiquement l'évolution de l'erreur.

```r
#erreur sur les fractions non utilisées de dfTrain
plot(taille,err_train,col="blue",type="l",ylim=c(0.0,0.5))

#erreur sur l'échantillon test à part
lines(taille,err_test,col="red")
```



5. Conclusion

"Big data par-ci, big data par-là..." me disait un collègue, limite moqueur je trouve. Mais il a raison, la profusion des données est certes un merveilleux terreau pour le développement de nouvelles technologies à la pointe de l'informatique. Ce que comprend l'informaticien que je suis. Mais cela ne doit cependant pas nous faire oublier que dans la partie analytique, l'objectif est - trivialement dirais-je - de produire les modèles les plus efficaces possibles. Si on peut se contenter de ne travailler que sur une fraction des données pour obtenir un niveau de performance satisfaisant, pourquoi s'en priver ? Le data miner que je suis comprend parfaitement ce discours.