TP5 : un exemple d'utilisation de la validation croisée

ENSAE 3ème année, 2017

Applications du bootstrap et autres techniques de rééchantillonnage (cours de M. Roquain)

Ce TP concerne un problème de classification de données postales (issues de la base de données `MNIST`), contenant des signes de chiffres manuscrits au format $28 * 28$ pixels, chaque pixel étant représenté par un niveau de gris allant de 0 à 255.

Exercice 0 (Création de votre fichier)

Créer votre fichier `nom_prenom_TP5.R`. N'oubliez pas d'effectuer des commentaires dans votre fichier (que vous utilisiez un notebook ou pas).

Exercice 1 (Chargement et représentation des données)

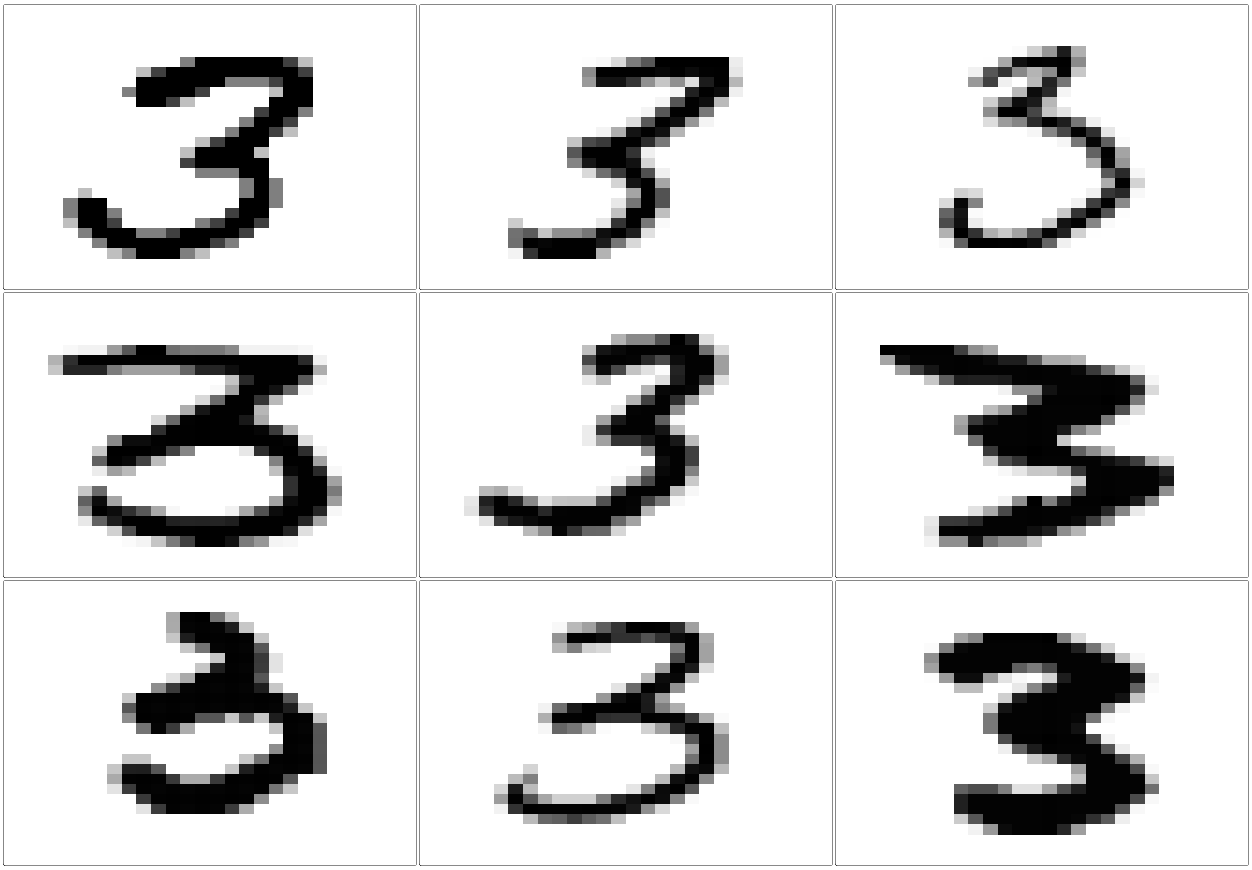
1) Récupérer le fichier `data1_for_TP5` disponible sur ma page http://etienne.roquain.free.fr/teaching.html. Charger ensuite les données de la manière suivante :

```
x=read.table("data1_for_TP5")
resol=28*28
```

Quelle est la dimension de x ?

2) Ces données sont structurées de la façon suivante : chaque ligne contient le label (`3` ou `5`), puis une image du signe manuscrit codée par $28 * 28$ pixels, chaque pixel étant représenté avec un niveau de gris dans $\{0, \dots, 255\}$. Nous pouvons représenter quelques lignes de x de la façon suivante :

```
par(mfrow=c(3,3),mar=c(0, 0, 0, 0) + 0.1,xaxt="n",yaxt="n")
for(i in 1:9){
image(t(matrix(as.integer((x[i,2:(resol+1)])),28,28,byrow=TRUE)[28:1,]),
      col  = gray((255:0)/255),cex=4)
}
```



Vérifier la valeur de `x[1:9,1]`. Faire un graphique similaire pour les lignes de x numérotées de 201 à 209.

Exercice 2 (Méthode des k plus proches voisins)

Le but est de créer une procédure automatique de reconnaissance de signe, ici réduite à prédire `3` ou `5` à partir d'une image d'un signe manuscrit (donc un vecteur de `28*28` niveau de gris). Nous allons utiliser pour cela la méthode classique des k plus proches voisins (kppv).

1) Pour voir comment fonctionne la méthode des kppv, prenons un exemple. Récupérer le fichier `Exemple_for_TP5` disponible sur ma page http://etienne.roquain.free.fr/teaching.html. Charger ensuite les données de la manière suivante :

```
x0=read.table("Exemple_for_TP5")
```

Représenter cette image.

2) Déterminons les images de x qui sont les plus proches de x_0. Une distance simple à utiliser est la norme $L^{1/10}$ entre les niveaux de gris.

```
distancex0=apply(x[,2:(resol+1)],1, FUN=function(line) mean(abs(line -x0[2:(resol+1)])^(1/10)))
orderdistancex0=order(distancex0)
orderlabel=x[orderdistancex0,1]
```

Que contient la variable `orderlabel` ?

3) Représenter les 35 plus proches voisins voisins de x0.
4) Pour déterminer automatiquement le label associé à x_0, la méthode des kppv retourne `res` obtenu ainsi :

```
k=10
res=3
if(mean(orderlabel[1:k])>=4)  res=5
```

Expliquer cette démarche.

Exercice 3 (Choix de k par validation croisée)

Dans la méthode des kppv, le choix de k est crucial. Une façon de le calibrer est de minimiser la function

$$R^{VC} : k \mapsto n^{-1} \sum_{i=1}^{n} \mathbf{1}_{\{\phi_k^{-i}(X_i) \neq Y_i\}},$$

où $n = 400$, X_i (resp. Y_i) est le codage (resp. le label) de l'image i (c'est-à-dire de la ligne i) de la base de données x, et ϕ_k^{-i} est le classifieur des kppv obtenu avec la base de données x privée de l'image numéro i.

1) Construire la matrice des distances entre lignes de x de la façon suivante

```
Distancematrice =apply(x[,2:785],1,FUN=function(line)
  apply(x[,2:785],1, FUN=function(line2) mean(abs(line2 -line)^(1/10))))
```

2) Pour chaque $i \in \{1, \ldots, 400\}$ fixé, que fait la fonction suivante ?

```
phimoinsi=function(i,k){
  distance=Distancematrice[i,]
  orderdistance=order(distance)
  orderlabel=x[orderdistance,1]
  res=3
  if(mean(orderlabel[2:(k+1)])>=4)  res=5
  return(res)
}
```

3) Utiliser cette fonction pour créer la fonction R^{VC}, que l'on nommera `ComputeRisqueVC`.
4) Quel est l'entier $k = \hat{k}^{CV}$ qui minimise `ComputeRisqueVC` ?

Exercice 4 (Performance de la classification)

1) Récupérer le fichier `data2_for_TP5` disponible sur ma page http://etienne.roquain.free.fr/teaching.html. Charger ensuite les données de la manière suivante (même format que x) :

```
y=read.table("data2_for_TP5")
```

2) Calculer la matrice des distances entre les images de x et celles de y comme suit :

```
Distancematricexy =apply(x[,2:785],1,FUN=function(line) apply(y[,2:785],1,
     FUN=function(line2) mean(abs(line2 -line)^(1/10))))
```

3) Pour le classifieur des kppv ϕ^{CV} avec le choix de k de l'exercice 3 (calibré uniquement à partir de x), calculer le taux d'erreur de classification

$$n^{-1}\sum_{i=1}^{n}\mathbf{1}_{\{\phi^{CV}(X_i')\neq Y_i'\}},$$

où X_i' (resp. Y_i') désigne le codage (resp. le label) de l'image i du jeu de données y. Commenter.