POLYTECHNIQUE MONTRÉAL
UNIVERSITÉ D'INGÉNIERIE

MTH8304 - Apprentissage non-supervisé et séries chronologiques

Département de Mathématiques et Génie Industriel
Hiver 2024
www.moodle.polymtl.ca

Devoir 4 : remise le 28 mars avant minuit

Les objectifs de ce devoir sont de se familiariser avec la classification non-supervisée lorsque l'on veut déterminer des caractéristiques de façon automatique dans les deux cas suivants :

- pour réduire la dimension avant une tâche de classification supervisée ;
- pour segmenter une image RGB.

Ce devoir noté sur 10 comporte deux parties valant respectivement 5 points et 3 points, 1 point étant attribué à la qualité de la présentation du rapport (sous forme de présentation) et 1 point pour le code à fournir en annexe.

Dans tout le devoir, assurez-vous de ne pas utiliser des librairies autres que celles qui sont précisées dans les énoncés.

Partie A : extraction automatique de caractéristiques de Fashion-MNIST

Dans cette partie, l'évaluation porte sur

(1) l'analyse des caractéristiques obtenues par clustering hiérarchique sur les images de Fashion-MNIST ;

(2) la sélection d'hyperparamètres optimaux associés à des caractérisques optimales sur un ensemble de validation ;

(3) l'illustration des partitions créées par les arbres de décisions et leur variance (c'est-à-dire leur sensibilité aux données d'entraînement).

Les données Fashion-MNIST sont du même format que les données MNIST : chaque image est de dimension 28×28. Chaque image est donc représentée par un vecteur de dimension 784 avec des intensités entre 0 et 255 qui seront ramenées dans l'intervalle $[0, 1]$. Les étiquettes associées à chaque image correspondent à un type de vêtement. Commencez par télécharger les données et lisez-les dans `R` avec le code suivant :

```
load.image.file <- function(filename) {
        ret <- list()
        f <- file(filename, "rb")
        readBin(f, "integer", n = 1, size = 4, endian = "big")
        ret$n <- readBin(f, "integer", n = 1, size = 4, endian = "big")
        nrow <- readBin(f, "integer", n = 1, size = 4, endian = "big")
        ncol <- readBin(f, "integer", n = 1, size = 4, endian = "big")
        x <- readBin(f, "integer", n = ret$n * nrow * ncol, size = 1,
            signed = F)
        ret$x <- matrix(x, ncol = nrow * ncol, byrow = T)
        close(f)
        ret
}
load.label.file <- function(filename) {
    f = file(filename, "rb")
    readBin(f, "integer", n = 1, size = 4, endian = "big")
    n = readBin(f, "integer", n = 1, size = 4, endian = "big")
    y = readBin(f, "integer", n = n, size = 1, signed = F)
    close(f)
    y
}
```

```
## remplacer "dir" par le nom du répertoire où se trouvent les données sur votre ordinateur:
dir <- "~/Data/cours/Fashion-MNIST/"

## lecture des données
mnist.train <- load.image.file(paste(dir, "train-images-idx3-ubyte", sep = ""))
mnist.train.lab <- load.label.file(paste(dir, "train-labels-idx1-ubyte", sep = ""))

## noms des étiquettes de classes
fashion.lab <- c("T-shirt/top", "Trouser", "Pullover", "Dress", "Coat", "Sandal", "Shirt",
                 "Sneaker", "Bag", "Ankle boot")

library(RColorBrewer); library(ggplot2); theme_set(theme_light(18))
## création d'un ensemble d'entraînement ayant 5000 exemples
set.seed(monmatricule) ## remplacer par son numéro de matricule
shuffle <- sample(mnist.train$n, mnist.train$n, replace = FALSE)
mnist.train$x <- mnist.train$x[shuffle,]
mnist.train.lab <- mnist.train.lab[shuffle]
df.train <- data.frame(x = mnist.train$x[1:5000,]/255,
                       lab = factor(fashion.lab[mnist.train.lab[1:5000]+1], levels = fashion.lab))
## coordonnées des pixels
img.y <- matrix(rep(1:28, 28),28,28, byrow = FALSE)
img.x <- matrix(rep(1:28, 28),28,28, byrow = TRUE)
coord.xy <- (data.frame(x = c(img.y[,28:1]), y = c(img.x[,28:1]))-1)/27
i <- 1
dig.i <- unlist(df.train[i,-785])
dig.xy.i <- data.frame(coord.xy, orig = dig.i)
ggplot(data = dig.xy.i, mapping = aes(x = x, y = y, col = orig)) + geom_point(size=1)+
         scale_colour_gradientn(colors=brewer.pal(n =9, name = "YlGnBu"), name = "") +
         coord_fixed() + xlab("") + ylab("")
```

Dans ce devoir, on entend par caractéristique la valeur moyenne des intensités de gris prises par les pixels d'un cluster donné (un sous-groupe de pixels dans l'image). S'il n'y a qu'un seul cluster, la caractéristique associée est la moyenne des intensités sur les 784 pixels. S'il y a 784 clusters, il y a 784 caractéristiques correspondant aux intensités de chaque pixel.

Pour obtenir les caractérisques des images de Fashion-MNIST, les descripteurs fournis en entrée au clustering sont les intensités de gris sur les 5000 images de l'ensemble d'entraînement ainsi que les coordonnées de chaque pixel. La distance utilisée dans le clustering entre deux pixels i et j est donnée par :

$$d(i,j) = \sqrt{\sum_{n=1}^{5000} \underbrace{(g_n^{(i)} - g_n^{(j)})^2}_{\text{intensités}} + \lambda^2 \underbrace{\left((x^{(i)} - x^{(j)})^2 + (y^{(i)} - y^{(j)})^2\right)}_{\text{coordonnées}}}. \quad (1)$$

Pour le pixel i, pour une image n donné, $g_n^{(i)}$ dénote l'intensité de gris du pixel et $(x^{(i)}, y^{(i)})$ sont ses coordonnées. De même pour le pixel j. La pondération λ est un hyperparamètre qui affecte l'importance des coordonnées relativement aux intensités dans la distance.
Pour extraire les caractéristiques de façon automatique, vous allez utiliser le clustering hierarchique tel que mis en oeuvre par la fonction agnes de la librairie `cluster` :

```
library(cluster)
lambda  <- 10
## descripteurs de l'ensemble d'entraînement à utiliser pour le calcul de la distance
train.loc <- cbind(t(df.train[,-785]), lambda * coord.xy)

## clustering hierarchique - prends quelques minutes à exécuter
hclust.train <- agnes(train.loc)
```

```
## extraction d'un clustering d'une taille donnée à partir du clustering hierarchique
k <- 40
clust.k <- cutree(hclust.train, k)

## transformation en variables catégoriques
clust.k.group <- factor(clust.k, levels=1:k)

## graphique du clustering
ggplot(data = data.frame(coord.xy, class=clust.k.group), mapping = aes(x = x, y = y, col = class)) +
    geom_point(size=5)+ scale_colour_manual(values = rep(brewer.pal(n =10, name = "Paired"),
    ceiling(k/10)), guide = "none") + coord_fixed() + xlab("") + ylab("")
```

Les caractéristiques sont alors données par la moyenne des intensités dans chaque cluster :

```
 ## calcul des intensités moyennes par cluster
dig.i.clust.k <- aggregate(dig.i,by = list(clust.k.group), FUN = "mean")

## attribution de l'intensité moyenne du cluster à chaque pixel du cluster
dig.i.clust.k.merge <- data.frame(cluster = clust.k.group, val = rep(NA, 784))
for (j in 1:k)
    dig.i.clust.k.merge$val[dig.i.clust.k.merge$cluster == j] <- dig.i.clust.k[j,2]

dig.xy.i <- data.frame(coord.xy, orig = dig.i, clust = dig.i.clust.k.merge[,2])

## graphiques des intensités moyennes pour l'image i
ggplot(data = dig.xy.i, mapping = aes(x = x, y = y, col = clust)) + geom_point(size=5)+
    scale_colour_gradientn(colors=brewer.pal(n =9, name = "YlGnBu"), name = "", lim = c(0,1)) +
    coord_fixed()+ xlab("") + ylab("")
```

Pour classifier, vous allez vous servir de la forêt aléatoire telle que mise en oeuvre dans la librairie `ranger` :

```
library(ranger)
## création de l'ensemble de validation ayant 5000 exemples
df.valid <- data.frame(x = mnist.train$x[5001:10000,]/255,
               lab = factor(fashion.lab[mnist.train.lab[5001:10000]+1], levels = fashion.lab))

## entraînement du classifieur sur les données originales:
## le nombre de clusters = 784
f0 <- ranger(y~., data = df.train)
pred.f0 <- predict(f0, data = df.valid)  ## estimation en validation

## erreur de classification totale sur l'ensemble de validation
mean(pred.f0$pred != df.valid$y)

## entraînement sur les caractéristiques automatiques
## le clustering est utilisé pour calculer les intensités moyennes par cluster
## sur l'ensemble d'entraînement:
trainclust.k <- aggregate(t(df.train[,-785]),by = list(clust.k.group), FUN = "mean")
fea.train.k <- data.frame(x = t(trainclust.k[,-1]), y=df.train$y)

## sur l'ensemble de validation:
validclust.k <- aggregate(t(df.valid[,-785]),by = list(clust.k.group), FUN = "mean")
fea.valid.k <- data.frame(x = t(validclust.k[,-1]), y=df.valid$y)
```

```
## entraînement du classifieur:
fk <- ranger(y~., data = fea.train.k)
pred.fk <- predict(fk, data = fea.valid.k) ## estimation en validation

## erreur de classification totale sur l'ensemble de validation
mean(pred.fk$pred != fea.valid.k$y)
```

Pour répondre à l'élément (1) de l'évaluation, vous allez :

1. Représenter un échantillon des images contenues dans le jeu de données Fashion-MNIST ;
2. Illustrer les caractéristiques automatiques extraites des images de Fashion-MNIST pour différentes valeurs de λ et pour différents nombres de clusters ;
3. Discuter, à l'aide des graphiques ci-dessus, de l'effet de l'hyper-paramètre λ et du nombre de clusters k sur le clustering et sur les caractéristiques extraites.

Pour répondre à l'élément (2) de l'évaluation, vous allez :

1. Déterminer des valeurs optimales pour les deux hyperparamètres affectant les caractéristiques automatiques : le nombre de clusters k et la pondération λ des coordonnées par rapport aux intensités de gris.

   Pour sélectionner des valeurs optimales pour ces deux hyperparamètres, vous allez utiliser le principe de maximisation de la capacité à généraliser sur la tâche de classification : les caractéristiques automatiques, pour des valeurs de λ et de k données, servent en entrée d'un algorithme de classification et la performance du classifieur est estimé sur un ensemble de validation.

   Adaptez le code ci-dessus qui met en oeuvre la librairie `ranger` pour effectuer l'entraînement pour différentes valeurs de λ et de k (notez que le même clustering hiérarchique peut être utilisé pour différentes valeurs de k sans re-exécuter la fonction `agnes`).
2. En étudiant les courbes d'erreur de validation, identifiez les valeurs optimales de λ et de k et commentez. En particulier, expliquez en quoi les valeurs de λ et k affectent la complexité du classifieur.

Pour répondre à l'élément (3) de l'évaluation, vous allez utiliser le jeu de données `iris` pour :

1. Choisir deux sous-ensembles des données et construire deux arbres de décision sur ces sous-ensembles en utilisant la librairie `tree` vue en classe et en ne retenant que deux descripteurs d'iris ;
2. Illustrer les arbres et les partitions obtenues sur les deux sous-ensembles et commenter.

Faites en sorte que les sous-ensembles choisis donnent lieu à des arbres différents.

Partie B : segmentation d'une image RGB

Dans cette partie, l'évaluation porte sur :

(1) la compréhension de l'initialisation du clustering avec une mélange de gaussiennes mis en oeuvre dans la librairie `mclust` ;

(2) le choix et la discussion du meilleur modèle pour faire la segmentation d'une image RGB ;

(3) l'illustration des étapes pour l'algorithme des K-moyennes et le clustering hiérarchique.

Pour répondre à l'élément (1) de l'évaluation, vous allez expliquer dans vos mots comment la fonction `Mclust` de la librairie `mclust` procède pour faire l'initialisation des paramètres du mélange de gaussiennes avant d'itérer avec l'algorithme EM.

Pour répondre à l'élément (2) de l'évaluation, vous allez suivre les étapes ci-dessous.

1. Choissisez aléatoirement une image parmi la base d'images mise à disposition à l'aide du code ci-dessous :

```
set.seed(monmatricule) # remplacez par votre matricule
mypath <- "~/Data/cours/images/" # adaptez le chemin à votre ordinateur
img.list <- list.files(path=mypath)
mon.img <- sample(img.list, 1)
img.path <- paste0(mypath, mon.img)
```

```
img1 <- imgRGB(img.path)

## pour faire le graphique associé à l'image choisie
library(jpeg);library(ggplot2)

imgRGB <- function(img.path){
    img <- readJPEG(img.path)
    imgDm <- dim(img)
# Assign RGB channels to data frame
   data.frame(
        x = rep(1:imgDm[2], each = imgDm[1]),
        y = rep(imgDm[1]:1, imgDm[2]),
        R = as.vector(img[,,1]),
        G = as.vector(img[,,2]),
        B = as.vector(img[,,3])
   )

}

# ggplot theme to be used
plotTheme <- function() {
  theme(
    panel.background = element_rect(
      linewidth = 3,
      colour = "black",
      fill = "white"),
    axis.ticks = element_line(
      linewidth = 2),
    panel.grid.major = element_line(
      colour = "gray80",
      linetype = "dotted"),
    panel.grid.minor = element_line(
      colour = "gray90",
      linetype = "dashed"),
    axis.title.x = element_text(
      size = rel(1.2),
      face = "bold"),
    axis.title.y = element_text(
      size = rel(1.2),
      face = "bold"),
    plot.title = element_text(
      size = 20,
      face = "bold",
      vjust = 1.5)
  )
}

# Plot the image
ggplot(data = img1, aes(x = x, y = y)) + geom_point(colour = rgb(img1[c("R", "G", "B")])) +
  labs(title = "Original Image") + xlab("") +  ylab("") +
    plotTheme()+ coord_fixed()
```

2. Faites d'abord la segmentation de votre image avec un appel à la fonction `Mclust` dédiée au clustering avec les paramètres par défaut.

   (a) Faites un graphique des critères d'information bayesien (BIC) et notez le choix de modèle optimal selon le BIC :

```
plot(img.mclust, what = "BIC")
```

où `img.mclust` est l'objet créé par un appel à `Mclust`

(b) Faites un graphique de la segmentation associée à ce clustering. Pour cela, utilisez le code ci-dessous :

```
i <- 1
col.i <- data.frame(t(apply(img1[c("R", "G", "B")][img.mclust$classification == i,], 2, mean)))

for (i in 2:img.mclust$G){
    col.i <- rbind(col.i,
              data.frame(t(apply(img1[c("R", "G", "B")][img.mclust$classification == i,], 2, mean))))
}

col.clust <- rgb(col.i)

ggplot(data = img1, aes(x = x, y = y)) +
  geom_point(colour = col.clust[img.mclust$classif]) +
  labs(title = "Segmented Image") +
  xlab("") + ylab("") + plotTheme()+ coord_fixed()
```

3. Choisissez deux modèles alternatifs à celui sélectionné par le BIC de l'étape précédente. Privilégiez des modèles parsimonieux (le moins de paramètres possibles) et avec un critère BIC élevé. Expliquez vos choix et illustrez la segmentation associée à chaque modèle. Discutez la présence ou l'absence de différences avec les segmentations précédentes.

Pour répondre à l'élément (3) de l'évaluation, vous allez :

1. sélectionner de façon aléatoire un sous-ensemble de 5 exemples des données Old faithful ;
2. calculer et illustrer les étapes de l'algorithme EM des K-moyennes sur ces 5 exemples ;
3. calculer et illustrer les étapes du clustering hiérarchique avec liaison simple sur ces 5 exemples.