Université Grenoble Alpes

Master SSD, BDA et IC
Enseignant : Jonathan El Methni
Apprentissage Statistique

TP 1 : Régression, ACP et loi normale multivariée

Exercice 1 : Régression Linéaire

Dans cet exercice nous allons travailler sur une problématique de régression à partir du jeu de données `regression-dataset.Rdata` qui contient les données suivantes :

- deux vecteurs `x` et `y`, qui vont nous servir de données d'apprentissage pour construire un modèle de régression de `y` à partir de `x`.
- deux vecteurs `x.test` et `y.test` qui nous serviront d'échantillons indépendants (ou de test) pour évaluer les performances de prédiction.

1. Charger le jeu de données et représenter y en fonction de x.
2. Effectuer la régression linéaire de y en fonction de x et représenter le modèle obtenu sur le même graphique. Ce modèle vous semble-t-il satisfaisant ?
3. Considérer maintenant des modèles polynomiaux d'ordre 3, 10 et 20. Que constatez-vous ?
4. Considérer des modèles polynomiaux d'ordre 1, 2, 3, 4, 5, 8, 10, 15 et 20 :
   - Apprendre les modèles à partir du couple (x, y).
   - Evaluer la qualité de la régression sur le jeu d'apprentissage (i.e., sur les données `(x,y)`) à partir de l'erreur quadratique moyenne (MSE), définie comme :

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

     où n est le nombre d'observations (60 dans notre cas) et $\hat{y}_i$ est la valeur prédite par le modèle pour le point x_i.
   - Tracer l'évolution de la MSE en fonction du degré du polynôme.

   Que constatez-vous ? Quel modèle vous semble réaliser la meilleure performance ?
5. Evaluer ces modèles sur le jeu de test :
   - Utiliser ces modèles pour prédire les réponses associées aux valeurs de `x.test`.
   - Calculer l'erreur de prédiction obtenue par ces différents modèles sur le jeu de test en terme de MSE.
   - Représenter sur un même graphique les MSE obtenues sur le jeu d'apprentissage et le jeu de test en fonction du degré du polynôme. Que constatez-vous ?

Rappels R :

- pour effectuer une régression linéaire on utilisera la fonction `lm`.
- pour effectuer une régression polynomiale, on utilisera la fonction `poly` directement dans l'appel à la fonction `lm` : `fit.poly = lm(y ~ poly(x,d))`
- pour obtenir des prédictions à partir d'un modèle obtenu par la fonction `lm` on peut directement utiliser la fonction `predict` associée en utilisant l'argument `newdata`.

  Attention, il faut prendre soin à utiliser le même nom de variable que celui utilisé lors de l'apprentissage du modèle, par exemple :
  - `fit = lm(y ~ x)`
  - `pred = predict(fit, newdata = data.frame("x"=x.test))`

Exercice 2 : ACP

Dans cet exercice nous travaillerons sur des données de puces à ADN obtenues sur 64 tumeurs cancéreuses d'origines diverses (e.g., prostate, rein) par le National Cancer Institute (NCI). Chaque échantillon est caractérisé par 6830 variables, correspondant à une mesure d'expression de gènes.

1. Chargez le jeu de données à l'aide des instructions suivantes

   ```
   y =as.character(read.table("datasets/nci.label")$V1)
   X =read.table("datasets/nci.data")
   X =t(X)
   ```

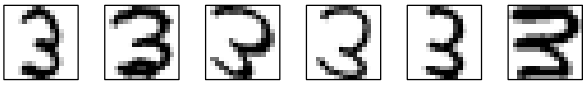
   puis représentez le nombre d'observations disponibles dans les différentes catégories. Le jeu de données se présente sous la forme de deux fichiers, `nci.data` et `nci.labels`, contenant respectivement la matrice de données et la catégorie des échantillons (types de cancer).
2. Supprimez l'échantillon de classe inconnue et les échantillons relatifs à des expériences de reproductibilité (leur nom contient le terme "repro").

   ```
   ind1 = grep("repro", y)
   ind2  = which(y == "UNKNOWN")
   ind.rm = c(ind1, ind2)
   y = y[-ind.rm]
   y = factor(y)
   X = X[-ind.rm,]
   ```

3. Réalisez une Analyse en Composantes Principales sur les données restantes en utilisant la commande `prcomp` ou `PCA` du package `FactoMineR` si vous préférez. Combien de composantes principales obtenez-vous ?
4. Représentez le jeu de données selon ses deux premières composantes principales en utilisant une couleur par type de cancer. Qu'observez-vous ?
5. Proposez une représentation permettant de conforter vos observations.
6. Calculer la proportion de variance expliquée par chacun des composantes et représentez la en fonction de l'indice de la composante principale.
7. Représenter cette information sous forme cumulée. Combien de composantes principales faut-il conserver pour expliquer 90% de la variance présente dans le jeu de données ? Quel taux de réduction de dimension cela représente t'il ? A l'inverse, combien de variance peut-on expliquer avec 30 composantes ?

Exercice 3 : ACP (bis)

Dans cet exercice nous allons analyser un ensemble d'images de ce type représentant le chiffre 3 :



L'objectif sera de réaliser une analyse en composantes principales afin d'expliciter les sources de variabilité prépondérantes au sein de ces caractères manuscrits.

1. Charger le jeu de données `digits-3.Rdata` et représentez quelques images.

   — le jeu de données se présente sous la forme d'un tableau à 3 dimensions nommé `I`. Les deux premières correspondent à la taille des images (16x16 pixels) et la troisième aux différentes observations (657 images).

   — on peut représenter une image en `R` via la fonction `image()`.

   — on définira un code couleur en niveau de gris ainsi : `cols = gray(seq(1,0,length.out=256))` la couleur noir correspondra à des valeurs élevées et la couleur blanche à des valeurs faibles.

   — on le passera à la fonction `image()` via l'argument `col` : `image(x, col = cols)`.

2. Transformer ce jeu de données en matrice en vue d'en réaliser une ACP.

```
X = apply(I, 3, function(x){as.vector(x)})
X = t(X)
```

3. Effectuer l'ACP et représenter le jeu de données selon ses deux composantes principales.

4. Visualiser un sous-ensemble d'images réparties régulièrement dans l'espace ACP.

   — Vous pourrez pour cela utiliser les indices contenus dans la matrice `ind.grid`.

   — Commencer par mettre en évidence les points correspondant dans le graphe ACP précédent (par exemple en les affichant d'une autre couleur), puis représenter les 25 images correspondantes dans une même figure, en suivant la structure (i.e., les lignes et les colonnes) de la matrice `ind.grid`.

   — Donnez une première interprétation des deux premières composantes principales en explicitant l'information capturée par ces dernières.

5. Représenter comme des images les coefficients définissant les deux premières composantes principales. Confortez votre précédente interprétation.

6. (Optionnel) Pour aller plus loin, utiliser la fonction `identify` pour aller choisir vos propres images dans l'espace ACP.

Exercice 4 : loi Normale multivariée

Dans cet exercice nous allons manipuler des fonctions `R` permettant de générer des observations selon des lois normales multivariées, et d'estimer leurs paramètres.

1. Générer $n = 200$ observations en deux dimensions distribuées selon des lois normales multivariées sphérique, diagonale et ellipsoïdale en utilisant la fonction `mvrnorm` du pagkage `MASS`. Vous pourrez par exemple utiliser les matrices de covariance suivantes :

$$\Sigma_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad \Sigma_2 = \begin{bmatrix} 4 & 0 \\ 0 & 1 \end{bmatrix}, \text{ et } \Sigma_3 = \begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix}.$$

2. Estimer leurs paramètres en utilisant la fonction `mvn` du package `mclust`, et en utilisant le modèle adapté (option `modelName`). Comparer les valeurs obtenues aux vrais valeurs des paramètres.
3. On se servira du code donné ci-dessous pour représenter la densité d'une gaussienne bivariée et superposer les tirages obtenus précédemment avec les distributions théoriques et/ou estimées. Pour cela on commence par implémenter une fonction calculant la densité de la loi normale multivariée.

```
mv = function(X, mu, S){
 p = length(mu)
 return( 1/( sqrt((2*pi)^p*det(S)) ) *  exp( -0.5 * t(X-mu) %*% solve(S) %*% (X-mu)) )
}
```

On peut visualiser la densité en utilisant la fonction `contour`. Pour cela il suffit de calculer sa valeur sur une grille discrétisant l'espace (x, y). Dans cet exemple nous calculons deux densités : les densités théoriques et estimées du modèle ellipsoïdal précédent.

```
# define grid
x = seq(-2,6, by = 0.02)
y = x
# init density
D = matrix(0, nrow = length(x), ncol = length(y))
D.hat = D
# extract estimated parameters
mu.hat= as.vector(fit3$parameters$mean)
S3.hat = fit3$parameters$variance$Sigma
# compute densities
for(i in 1:nrow(D)){
  for(j in 1:ncol(D)){
    X = matrix( c(x[i],y[j]), nrow = 2, ncol = 1)
    D[i,j] = mv(X, mu, S3)
    D.hat[i,j]  = mv(X, mu.hat, S3.hat)
  }
}
```

Enfin, on trace les deux densités sur le même graphique, avec les points tirés précédemment.

```
plot(X3[,1], X3[,2], pch = 19, col = "grey")
contour(x,y,D, add = TRUE)
contour(x,y,D.hat, add = TRUE, col = "red")
legend("bottomright", c("densité théorique","densité estimée"), col = c("black","red"), lwd = 1)
```

4. Combien de paramètres faut-il estimer avec chacun de ces modèles pour un problème en d dimensions ? Tracer l'évolution de ces nombres de paramètres quand la dimension d augmente de 2 à 10. Commenter.