

TP 2 : APPRENTISSAGE NON-SUPERVISÉ

Exercice 1 : Multi-Dimensional Scaling

Dans cet exercice, nous allons apprendre à utiliser les outils de R permettant de manipuler des matrices de distance (ou plus généralement, de dissimilarité). Pour cela nous allons travailler sur un jeu de données dites fonctionnelles, où chaque observation est une courbe.

1. Chargez le jeu de données **spectra.Rdata**. Il se constitue :
 - d'une matrice **X** de taille 215 x 100, contenant 215 courbes faites de 100 points
 - d'un vecteur **y** indiquant la catégorie (1 ou 2) associée à chacune des observations.
2. Représentez sur une même figure l'ensemble des 215 courbes, avec un code couleur indiquant leur catégorie. En pratique on ignore la catégorie des groupes car on est dans un cas d'apprentissage non-supervisé. Dans le cadre du TP on souhaite évaluer la qualité des méthodes on connaît donc les catégories. Les données sont dites labellisées.
3. Utilisez la fonction **dist** pour calculer la matrice de distance Euclidienne du jeu de données. Quelles autres distances peut-on calculer avec cette fonction ? Quelle est le lien entre la distance Euclidienne et la distance de Minkowski ?
4. Effectuez une ACP sur les données et visualisez le nuage de points des individus dans le premier plan factoriel. Vous pouvez ajouter un code couleur reflétant la catégorie des individus.
5. La fonction **cmdscale** permet de transformer une matrice de distance en un ensemble de points en k dimensions (k étant un paramètre à spécifier) tels que deux points seront d'autant plus proches dans cet espace que leur distance dans la matrice est faible. Appliquez cette fonction à la matrice de distance calculée précédemment pour la transformer en un ensemble de points en 2 dimensions, et visualiser le nuage de points obtenus avec un code couleur reflétant leur catégorie. Qu'observez-vous ?
6. Reproduire cet analyse en utilisant la corrélation de Pearson pour quantifier la similarité entre les courbes :
 - calculer la matrice de corrélation entre les courbes du jeu de données.
 - en utilisant la fonction **as.dist**, construire un objet de type **dist** à partir de cette matrice, en utilisant 1 moins la corrélation comme critère de dissimilarité.
 - projeter la matrice de distance obtenue en 2 dimensions avec la fonction **cmdscale** et visualiser le nuage de points obtenu.
Qu'observez-vous ? Comment interprétez-vous ces résultats ?
7. Comparez les distributions des distances/dissimilarités intra- et inter-classes (i.e., entre observations de la même catégorie et de catégories distinctes) pour conforter votre interprétation.

Exercice 2 : SVM à une classe

Dans cet exercice, nous allons apprendre à manipuler des outils R permettant d'estimer de manière non-paramétrique le support d'une distribution par l'algorithme des SVMs à une classe ("one-class SVM"). Cet algorithme met en jeu deux paramètres principaux :

- un paramètre noté $\nu \in [0, 1]$ correspondant à la proportion de points du jeu d'apprentissage que l'on souhaite laisser en dehors du support.
- un (ensemble de) paramètre(s) entrant dans la définition de la fonction "noyau" de la SVM, qui définit la nature de la frontière du support. Dans notre cas, nous utiliserons un noyau gaussien, contrôlé par un paramètre γ , correspondant à l'inverse d'un paramètre de variance.

L'objectif de l'exercice sera d'évaluer l'impact de ces deux paramètres sur la nature du support estimé. Nous travaillerons pour cela sur un jeu de données artificiel en deux dimensions, et utiliserons le package `e1071`.

1. Charger et représenter le jeu de données `exo-4.Rdata`. Il contient une matrice `X` contenant 210 instances en deux dimensions.
2. Charger la fonction `plot_with_contours()` disponible dans le fichier `utils-exo4.R`. Cette fonction permet de tracer un nuage de points en deux dimensions ainsi que les lignes de niveau (ou "contours") d'une fonction, tous deux passés en paramètres (via les arguments `X` et `my_func`).
3. Reproduire l'exemple suivant qui permet d'estimer et de représenter le support de la distribution :

```
library(e1071)
source("utils.R")
# fit model
svm.model = svm(X, type='one-classification', nu=0.10, kernel="radial")
# make predictions
y.pred = predict(svm.model, X)
# show outliers
plot(X[,1], X[,2], col = factor(y.pred), xlab = "x1", ylab = "x2")
# show decision boundary
svm_pred = function(x){
  return( predict(svm.model, x))
}
plot_with_contours(X, svm_pred, y = factor(y.pred))
```

4. Reproduire cette analyse pour des valeurs de $\nu \in [0.01, 0.05, 0.1, 0.25]$. Analyser la nature du support obtenu et vérifier que la proportion de points faisant partie du support est conforme à ce qui est attendu.
5. Choisir la valeur de ν vous semblant la plus appropriée et reproduire cette analyse en faisant varier le paramètre γ . Quel est l'impact du paramètre γ sur la régularité du sup-

port ? Les "outliers" sont-ils toujours les mêmes quand le paramètre γ varie (à ν fixé) ?

6. Pour aller plus loin, construire le support pour la configuration (ν, γ) qui vous semble la plus appropriée et comparer vos résultats avec une approche paramétrique basée sur l'estimation d'une gaussienne multivariée.

Les "outliers" identifiés par la SVM à une classe tombent-ils systématiquement dans des zones où la densité gaussienne est faible ?

- pour estimer les paramètres de la gaussienne multivariée vous pourrez par exemple utiliser la fonction `mvn()` du package `mclust` via la commande

```
fit.mvn = mvn("XXX", X)
```

- pour représenter cette gaussienne multivariée, vous pourrez utiliser la fonction

```
plot_with_contours()
```

comme précédemment. Il suffira pour cela de créer une fonction calculant la fonction densité de la gaussienne multivariée à partir des paramètres estimés.

Exercice 3 : classification hiérarchique

Dans cet exercice nous allons apprendre à manipuler les outils de R permettant d'effectuer une classification hiérarchique. Nous travaillerons sur un jeu de données de caractères manuscrits tel que celui utilisé dans le TP précédent, mais contenant des caractères allant de 0 à 9.

1. Chargez le données `digits.Rdata`. Il contient deux matrices `X` et `X.large` contenant respectivement 100 et 5000 caractères. Les caractères sont stockés comme des lignes de ces matrices, et sont représentés par 256 variables (16 x 16 pixels). Les vecteurs `y` et `y.large` indiquent la valeur des caractères.
2. On commencera par travailler sur le petit jeu de données :
 - (a) Calculez une matrice de distance en utilisant la fonction `dist` et réaliser le clustering hiérarchique en utilisant la fonction `hclust`. On utilisera ici leur paramétrage par défaut (distance Euclidienne et agglomération "complete").
 - (b) Représentez le dendrogramme obtenu en utilisant la fonction `plot` appliqué à l'objet renvoyé par la fonction `hclust`.
 - (c) Faites en sorte d'afficher la valeur des caractères (i.e., les chiffres entre 0 et 9) dans le dendrogramme.
 - R utilise en fait la fonction `plot.hclust` pour représenter l'objet renvoyé par la fonction `hclust`. Référez-vous à l'aide de cette fonction si besoin.
 - (d) La fonction `cutree` permet d'obtenir un clustering en "coupant" le dendrogramme à une certaine hauteur ou en un certain nombre de clusters. Tracez l'évolution du nombre de clusters que l'on obtient quand on fait varier la hauteur de coupe `h` entre 0 et 25 (par pas de 1). A l'inverse, générez le clustering que l'on obtient quand on choisit d'avoir 10 groupes. A quelle hauteur cela correspond t'il ? Représentez la sur le dendrogramme.
 - (e) Construire le tableau de contingence reliant les indices des clusters et les valeurs de caractère. Ce clustering vous semble t'il satisfaisant ? Interprétez ces résultats à la lueur du dendrogramme pour voir si certaines confusions sont plus surprenantes que d'autre.

- (f) (Optionnel) Pour aller plus loin, explorez le choix d'autres distances et/ou d'autres stratégies d'agrégation lors du clustering hiérarchique pour tâcher d'améliorer les résultats.
3. On va ensuite reproduire ce type d'analyse sur le jeu de données plus conséquent :
 - (a) Calculez une matrice de distance, réalisez le clustering hiérarchique et coupez le dendrogramme obtenu pour obtenir 10 clusters.
 - (b) Construire la table de contingence reliant les indices de cluster et les valeurs de caractères. Représentez la sous la forme d'un diagramme en bâton avec la fonction `barplot`.
 - On introduira une barre par cluster, la hauteur des barres représentant le nombres de caractères des différentes valeurs obtenu au sein de chaque cluster.
 - (c) Représentez la même information sous forme de proportions conditionnelles : les proportions de caractères de différents types obtenus au sein de chaque cluster.
 - pour convertir la table d'effectifs en table de proportion, on pourra par exemple utiliser la fonction `prop.table`.

Exercice 4 : heatmap

Dans cet exercice nous allons apprendre à représenter un jeu de données sous la forme d'une "heatmap", que l'on obtient en effectuant à la fois un clustering hierarchique des lignes et des colonnes d'une matrice de données. Nous allons pour cela travailler à partir du jeu de données de puces à ADN relatif aux tumeurs cancéreuses utilisé lors du dernier TP, et utiliserons la fonction `aheatmap` du package NMF.

1. Installez le package NMF.
2. Chargez le données `nci_small.Rdata`.

Ce jeu de données contient 59 des 64 observations disponibles après suppression des échantillons relatifs à des expériences de reproductibilité et d'origine inconnue (cf TP1), et restreint à un sous ensemble de 675 variables choisies parmi les 6830 initiales.

3. Réalisez un clustering hiérarchique et afficher le dendrogramme correspondant. On gardera le paramétrage par défaut des fonctions `dist` et `hclust`.
4. Réalisez un "heatmap" en appelant la fonction `aheatmap` à partir de la matrice `X`. Utilisez le vecteur `y` comme option `annRow` pour représenter sur le heatmap l'origine des tumeurs. Qu'observez-vous ? Le dendrogramme obtenu sur les tumeurs est-il le même que celui obtenu par la fonction `hclust` ?
5. Réalisez un second heatmap en désactivant le clustering des variables (gènes) en définissant l'option `Colv = NA`. Qu'observez-vous ?

Exercice bonus : Multi-Dimensional Scaling

- Coder la méthode MDS pas à pas suivant l'algorithme donné dans le cours et comparer vos résultats avec la fonction `cmdscale`. Vous pouvez l'appliquer au jeu de données de votre choix.