DAUPHINEPSL★ UNIVERSITÉ PARIS RESEARCH UNIVERSITY

Machine Learning et Data Mining
2015 / 2016

Travaux Pratiques n° 1 : Approches non-supervisées

Objectifs : mise en pratique sous R des concepts de l'apprentissage supervisé vus en cours - Réduction de dimensionalité, Clustering

1 Avant goût

1.1 ACP

Considerer la matrice X de données brutes :

$$X = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 2 \\ 2 & 2 & 1 \\ 1 & 0 & 0 \\ 2 & 3 & 2 \end{pmatrix}$$

(i) Créer un fichier **.txt** avec la matrice X et importer les données, e.g. :

```
X = read.table("Nom.txt", header = FALSE)
```

(ii) Calculer le centre de gravité (individu moyen), la matrice Y des données centrées.
(iii) Calculer la matrice V des variances-covariances.
(iv) Calculer la variance de la première variable avec la commande **var**. Commenter.
(v) Utiliser la commande **eigen** pour obtenir les vecteurs propes de V.
(vi) Projeter les individus sur le plan factoriel principal. Utiliser la commande **plot** pour les visualiser.
(vii) Charger la librairie "FactoMineR" et utiliser la commande **PCA** pour faire une ACP sur X.

1.2 Analyse sémantique latente

Le but dans cette section est d'introduire le paquet `lsa` et de son utilisation. Commencez par télécharger celui et exécutez pas à pas le code suivant :

```
library(lsa)
# create some files
td = tempfile()
dir.create(td)
```

```
write( c("dog", "cat", "mouse"), file=paste(td, "D1", sep="/"))
write( c("hamster", "mouse", "sushi"), file=paste(td, "D2", sep="/"))
write( c("dog", "monster", "monster"), file=paste(td, "D3", sep="/"))
write( c("dog", "mouse", "dog"), file=paste(td, "D4", sep="/"))

# read files into a document-term matrix
myMatrix = textmatrix(td, minWordLength=1)

# create the latent semantic space
myLSAspace = lsa(myMatrix, dims=dimcalc_raw())

# display it as a textmatrix again
round(as.textmatrix(myLSAspace),2) # should give the original

# create the latent semantic space
myLSAspace = lsa(myMatrix, dims=dimcalc_share())

# display it as a textmatrix again
myNewMatrix = as.textmatrix(myLSAspace)
myNewMatrix # should look be different!
# compare two terms with the cosine measure
cosine(myNewMatrix["dog",], myNewMatrix["cat",])
# compare two documents with pearson
cor(myNewMatrix[,1], myNewMatrix[,2], method="pearson")

##Find closest terms

# calc associations for mouse
associate(myNewMatrix, "mouse")

# clean up
unlink(td, recursive=TRUE)
```

1.3 K-means

Le but ici est de vous donner un exemple simple d'utilisation de l'algorithme k-moyennes :

```
x = c(-2,-2,0,2,-2,3)
y= c(2, -1,-1,2,3,0)
don = matrix(data=c(x,y), nr=6, nc=2)
ctre = c(-1,2,-1,3)
ctre1 =matrix(data=ctre, nr=2, nc=2)
cl1 = kmeans(don,ctre1,algorithm="Lloyd")
plot(don, col = cl1$cluster)
points(cl1$centers, col = 1 : 2, pch = 8, cex=2)
```

1.4 CAH

Même objectif que pour la k-means pour en utilisant l'algorithm CAH. La suggestion dans le code suivant est d'utiliser l'indice du lien moyen. Vous pouvez tester différents

types d'indices tels que le lien minimal ou maximal.

```
> data(iris)
> don=iris[,1 :4]
#
# Classification par la C.A.H.
#
> hclust(dist(don), "ave")
> plot(hc)
> plot(hc, hang = −1)}
```

2 La vraie vie : Text Mining

Nous allons considérer dans ce TP l'analyse d'un corpus de texte. Commencez par récupérer l'archive `TP_TextMining.zip` envoyée par mail ou disponible sur `MyCourse`

1. Commencer par décompresser l'archive envoyée par email dans votre espace de travail local (e.g. `~/DataMining/TP1/`)
2. Changer d'espace de travail : `setwd("~/DataMining/TP1/")`
3. Commencer par installer les librairies nécessaires :

```
Needed <- c("tm", "SnowballCC", "RColorBrewer", "ggplot2", "wordcloud", "biclust",
    "cluster", "igraph", "fpc", "corrplot")
install.packages(Needed, dependencies = TRUE)

install.packages("Rcampdf", repos = "http://datacube.wu.ac.at/", type = "source")
```

4. Charger la librairie **tm** :

```
library(tm)
```

5. Importer les données dans une variable "docs", et inspecter e.g. :

```
docs <-VCorpus(DirSource("./")
summary(docs)
```

6. Vous pouvez visualiser le content d'un élément du corpus (ex. le premier) : `docs[[1]]$content` ou `writeLines(as.character(docs[1]))`

7. Nettoyer le corpus pour pouvoir le traiter :

   — Enlever les ponctuations

   ```
   docs <- tm_map(docs,removePunctuation)
   ```

   — Mettre en minuscule

   ```
   docs <- tm_map(docs, tolower)
   docs <- tm_map(docs, PlainTextDocument)
   ```

— Enlever les chiffres
```
docs <- tm_map(docs, removeNumbers)
```
— Autres nettoyage (facultatif)

```
for (j in seq(docs)) {
    docs[[j]] <- gsub("/", " ", docs[[j]])
    docs[[j]] <- gsub("@", " ", docs[[j]])
    docs[[j]] <- gsub("\\|", " ", docs[[j]])
    docs[[j]] <- gsub("\u2028", " ", docs[[j]])
}
```
— Enlever les mots fréquents :

```
docs <- tm_map(docs, removeWords, stopwords("english"))
docs <- tm_map(docs, PlainTextDocument)
```
— Enlever les mots particuliers :

```
docs <- tm_map(docs, removeWords, c("syllogism", "tautology"))
```
— Fusionner des termes :

```
for (j in seq(docs))
{
  docs[[j]] <- gsub("fake news", "fake_news", docs[[j]])
  docs[[j]] <- gsub("inner city", "inner-city", docs[[j]])
  docs[[j]] <- gsub("politically correct", "politically_correct", docs[[j]])
}
docs <- tm_map(docs, PlainTextDocument)
```
— Extraire les racines (? manier avec précaution)

```
docs_st <- tm_map(docs, stemDocument)
docs_st <- tm_map(docs_st, PlainTextDocument)
writeLines(as.character(docs_st[1])) # inspection.
```

— Enlever les blancs s'il en reste
```
docs <- tm_map(docs, stripWhitespace) docs <- tm_map(docs, PlainTextDocument)
```
— Test
```
docs[[2]]$content
```

8. Créer la matrice documents-termes

```
dtm <- DocumentTermMatrix(docs)
dtm
inspect(dtm[1:5, 1:20])
```

9. Créer la matrice termes-documents-

```
tdm <- TermDocumentMatrix(docs)
tdm
inspect(tdm[1:5, 1:20])
```

10. Réduction de la matrice documents-terms

```
inspect(removeSparseTerms(dtm, 0.2))
inspect(removeSparseTerms(dtm, 0.4))
```

11. Visualisation de la matrice termes-documents

```
# Créer une DTM avec des poids normalisées
library(corrplot)
dtm.norm <- as.matrix(removeSparseTerms(
  DocumentTermMatrix(docs, control=list(weighting=weightTf)),
  0.2))
corrplot (dtm.norm, is.corr=FALSE)
```

12. Création de la matrice TFidF

```
dtm.TfIdf2 <- as.matrix(removeSparseTerms(
  DocumentTermMatrix(docs, control=list(weighting=weightTfIdf)),
  0.2))
# Plot simple
corrplot (dtm.TfIdf2, is.corr=FALSE)
```

13. Dans cet exemple on se contente d'utiliser une version non normalisée

```
dtms <- removeSparseTerms(dtm, 0.2)
dtms
```

14. Inspection des mots les plus fréquents

```
freq <- colSums(as.matrix(dtm))
head(table(freq), 20) # inspecter les 20 mots les plus fréquents

tail(table(freq), 20)#inspecter les 20 mots les moins fréquents

#Meme chose sur la matrice seuillee et d'une autre maniere

findFreqTerms(dtm, lowfreq=50)
```

15. Visualisation

```
library(ggplot2)
p <- ggplot(subset(wf, freq>50), aes(x = reorder(word, -freq), y = freq)) +
          geom_bar(stat = "identity") +
          theme(axis.text.x=element_text(angle=45, hjust=1))
p
```

16. Calculer les corrélations entre terms

```
library(ggplot2)
findAssocs(dtm, c("country" , "american"), corlimit=0.85)
               # limite de correlation egale a 0.85


findAssocs(dtms, "think", corlimit=0.70)
```

17. Nuage de mots

```
library(RColorBrewer)
set.seed(142)
wordcloud(names(freq), freq, min.freq=25)
```

```
              # affiche les mots qui apparaissent au moins 25 fois.

set.seed(142)
wordcloud(names(freq), freq, max.words=100)
     # affiche les 100 mots les + frequents

#avec un peu de couleur

set.seed(142)
wordcloud(names(freq), freq, min.freq=20, scale=c(5, .1),
         colors=brewer.pal(6, "Dark2"))

set.seed(142)
dark2 <- brewer.pal(6, "Dark2")
wordcloud(names(freq), freq, max.words=100,
      rot.per=0.2, colors=dark2)
```

18. Application d'une classification hiérarchique

```
library(cluster)
d <- dist(t(dtmss), method="euclidian")
fit <- hclust(d=d, method="complete")
          # on peut changer de linkage: method="ward.D"
fit

# interpretation du dendogramme

plot.new()
plot(fit, hang=-1)
groups <- cutree(fit, k=6)   # "k=" defines the number of clusters you are using
rect.hclust(fit, k=6, border="red") # draw dendogram with red borders around the 6 c
```

19. Application du k-means sur les termes (vous pouvez essayer sur les documents)

```
library(fpc)
d <- dist(t(dtmss), method="euclidian")
kfit <- kmeans(d, 2)
clusplot(as.matrix(d), kfit$cluster, color=T, shade=T, labels=2, lines=0)
```

20. Appliquer une Décomposition en Valeurs Singulières à la matrice `dtms`. Expliquez ? (commande `svd`)
21. Appliquer une ACP à cette matrice de données. Interprétez !
22. Appliquer la méthode k-moyennes sur la matrice `dtms` après application de la Décomposition en Valeurs Singulières. Commentez ! Rque : Vous pouvez aussi passer par le paquet LSA.
23. Appliquer la méthode k-moyennes sur la matrice `dtms` après application de l'ACP. Commentez !