MAD_project_kmeanspp

2022-12-07

Do Thanh Dat LE

Gan WANG

Exercise 1: L'algorithme des kmeans++

##1. Programmer l'algorithme d´ecrit dans la section 2.2 de l'article scientifique joint au sujet.

First we will program a function of K-means++ algorithm with 2 argument: X is the input data and k is the number of clusters.

```r
# We create a function for the K-means++ algorithm
# with X is the input data and K is the number of clusters
kmeans_plus_plus <- function(X, k) {
  centers <- data.frame() #create an empty vector to store k centers
  #Step 1a: Choose an initial center c_1 uniformly at random from X
  centers <- rbind(centers, X[floor(runif(1,min = 1,max = nrow(X))),]) #Store the first center
  #Step 1b,1c: Choose the next center c_i in X with probability according to D(x)
  for (id in (2:k)) {
    # create an empty vector to store all the D(x)
    distances <- c()
    # Compute the D(x) is the smallest distance from a data point x to the closest center
    for (j in (1:nrow(X))) {
      D <- +Inf #
      for (i in (1:nrow(centers))){
        D_temp <- sqrt(sum((X[j,] - centers[i,])^2))
        D <- min(D, D_temp)
      }
      distances <- append(distances, D) #add new D(x) for each points
    }
    proba <- (distances^2)/sum(distances^2)
    index <- sample(c(1:length(proba)), 1, prob = proba)
    centers <- rbind(centers, X[index,])
  }
  #Step 2: Proceed as with the standard k-means algorithm.
  result <- kmeans(X, centers)
  return(result)
}
```

##2.Simuler des donn´ees NORM-10 et NORM-25 suivant le protocole d´ecrit par la section 6.1

First, we write the function to simulate the data NORM-10 and NORM-25

```r
# Simulation function
# c: number of centers
# d: dimension
# n: number of points to simulate
require(mnormt)
```

```
## Loading required package: mnormt
```

```r
simulation_func <- function(c, d, n){
  norm_data <- data.frame()
  for (i in 1:c) {
    center <- runif(d,0,500)
    sigma <- diag(x = 1, nrow = d, ncol = d)
    data.points <- rmnorm(n/c, mean = center, varcov = sigma)
    norm_data <- rbind(norm_data, data.points)
  }
  return(norm_data)
}
```

We use the above function to simulate 2 data NORM-10 and NORM-25 with 1000 data points

```r
# Create the Norm-10 data
norm_10 <- simulation_func(c = 10,d = 15,n = 1000)
# Create the Norm-25 data
norm_25 <- simulation_func(c = 25,d = 15,n = 1000)
```

##3.Comparer votre algorithme au kmeans classique en vous inspirant du tableau 1 sur les jeux de donnees simulees.

We use the K-means++, K-means to cluster the data NORM-10 and NORM-25 in order to compare these two algorithms

```r
# Use k-means++ for NORM-10 for 10 clusters
kmeanspp.res10 <- kmeans_plus_plus(X = norm_10, k = 10)
kmeanspp.res10$tot.withinss #The potential
```

```
## [1] 14852.85
```

```r
# Use k-means for NORM-10 for 10 clusters
kmeans.res10 <- kmeans(norm_10, centers = 10)
kmeans.res10$tot.withinss #The potential
```

```
## [1] 44367347
```

We can see that the K-means++ algorithm returns the result with potential ϕ which is smaller than K-means classic when clustering on the NORM-10.

```r
# Use k-means++ for NORM-25 for 10 clusters
kmeanspp.res25 <- kmeans_plus_plus(X = norm_25, k = 10)
kmeanspp.res25$tot.withinss #The potential
```

```
## [1] 110738245
```

```r
# Use k-means for NORM-25 for 25 clusters
kmeans.res25 <- kmeans(norm_25, centers = 10)
kmeans.res25$tot.withinss #The potential
```

```
## [1] 128179066
```

We can see that the K-means++ algorithm returns the result with potential ϕ which is smaller than K-means classic when clustering on the NORM-25.

#Exercise 2. Donnees iris

#1. Appliquer votre algorithme au jeu de donnees iris et comparer aux k-means et a l'algorithme mclust.

We use the K-means++, K-means and Mclust to cluster the data iris in order to compare these algorithm

```r
# K-means++ for 3 clusters
data("iris")
kmeanspp.iris.res <- kmeans_plus_plus(X = iris[,-5], k = 2)
kmeanspp.iris.res$centers
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1     6.301031    2.886598     4.958763    1.695876
## 2     5.005660    3.369811     1.560377    0.290566
```

```r
kmeanspp.iris.res$cluster
```

```
##   [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [38] 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [75] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1
## [112] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [149] 1 1
```

```r
potential.kmeanspp <- kmeanspp.iris.res$tot.withinss
potential.kmeanspp
```

```
## [1] 152.348
```

```r
# K-means for 3 clusters
kmeans.iris.res <- kmeans(iris[,-c(5)],centers = 2)
kmeans.iris.res$centers
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1     6.301031    2.886598     4.958763    1.695876
## 2     5.005660    3.369811     1.560377    0.290566
```

```r
kmeans.iris.res$cluster
```

```
##   [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [38] 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [75] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1
## [112] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [149] 1 1
```

```r
potential.kmeans <- kmeans.iris.res$tot.withinss
potential.kmeans
```

```
## [1] 152.348
```

```r
# Mclust for 3 clusters
library(mclust)
```

```
## Package 'mclust' version 5.4.10
## Type 'citation("mclust")' for citing this R package in publications.
```

```r
mclust.iris.res <- Mclust(iris[,-5], G = 3)
summary(mclust.iris.res, parameters = T)
```

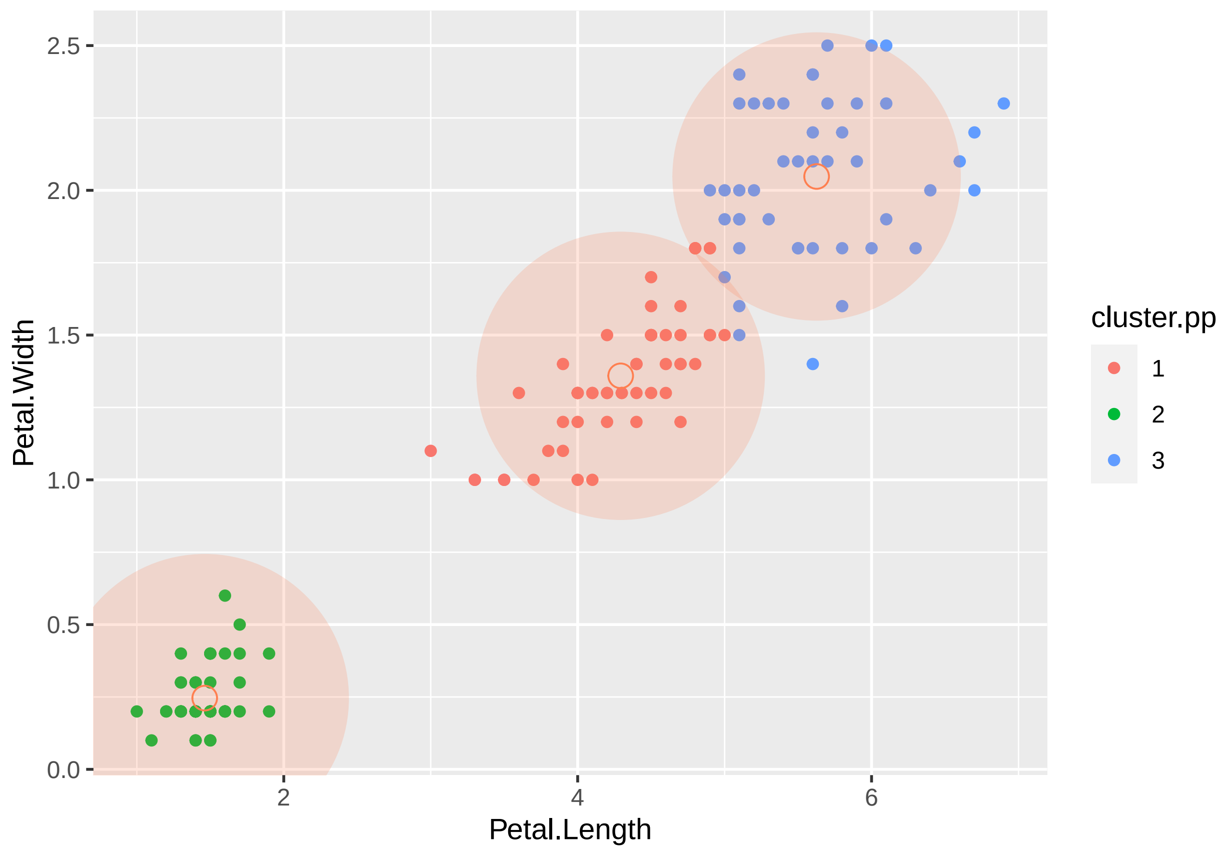
```
## ----------------------------------------------------
## Gaussian finite mixture model fitted by EM algorithm
## ----------------------------------------------------
##
## Mclust VEV (ellipsoidal, equal shape) model with 3 components:
##
##  log-likelihood   n df       BIC       ICL
##        -186.074 150 38 -562.5522 -566.4673
##
## Clustering table:
##  1  2  3
## 50 45 55
##
## Mixing probabilities:
##         1         2         3
## 0.3333333 0.3005423 0.3661243
##
## Means:
##               [,1]     [,2]     [,3]
## Sepal.Length 5.006 5.915044 6.546807
## Sepal.Width  3.428 2.777451 2.949613
## Petal.Length 1.462 4.204002 5.482252
## Petal.Width  0.246 1.298935 1.985523
##
## Variances:
## [,,1]
##              Sepal.Length Sepal.Width Petal.Length Petal.Width
## Sepal.Length   0.13320850  0.10938369  0.019191764 0.011585649
## Sepal.Width    0.10938369  0.15495369  0.012096999 0.010010130
## Petal.Length   0.01919176  0.01209700  0.028275400 0.005818274
## Petal.Width    0.01158565  0.01001013  0.005818274 0.010695632
## [,,2]
##              Sepal.Length Sepal.Width Petal.Length Petal.Width
## Sepal.Length   0.22572159  0.07613348   0.14689934  0.04335826
## Sepal.Width    0.07613348  0.08024338   0.07372331  0.03435893
## Petal.Length   0.14689934  0.07372331   0.16613979  0.04953078
## Petal.Width    0.04335826  0.03435893   0.04953078  0.03338619
## [,,3]
##              Sepal.Length Sepal.Width Petal.Length Petal.Width
## Sepal.Length   0.42943106  0.10784274   0.33452389  0.06538369
## Sepal.Width    0.10784274  0.11596343   0.08905176  0.06134034
## Petal.Length   0.33452389  0.08905176   0.36422115  0.08706895
## Petal.Width    0.06538369  0.06134034   0.08706895  0.08663823
```

Regarding the results of three algorithms, we can see that the K-means++ and K_means has the same result, these two return the same centers and they have equal potential, which is 78.85144.

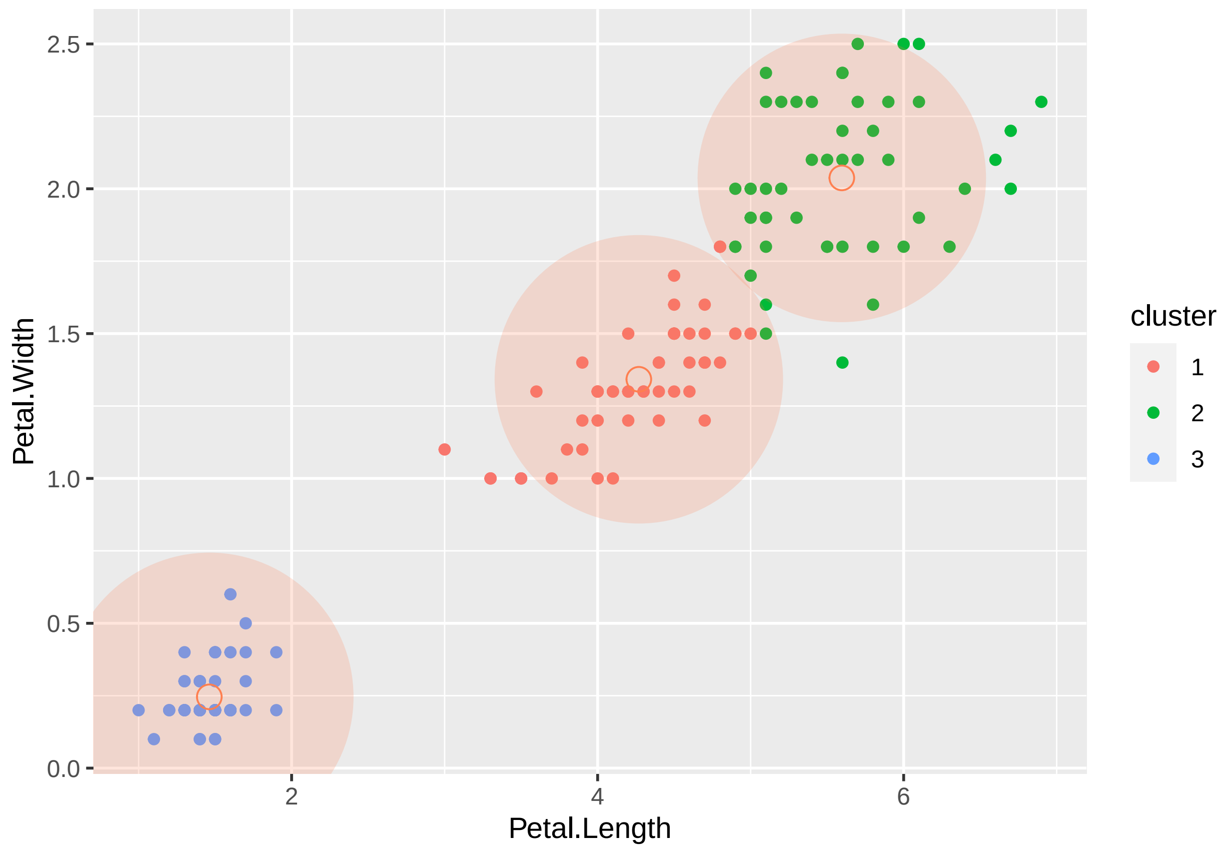
However, the mclust has different result with different centers.

Next, we can just keep the data of the Petal.Length and Petal.Width to do the clustering with K-means++ and K-means. Then we plot the graph to show the center and 3 cluster in two-dimensional space.

```
# Draw plot
library(ggplot2)
#k-means++
kmeanspp.res <- kmeans_plus_plus(X = iris[,-c(1,2,5)], k = 3)
cluster.pp <-as.factor(kmeanspp.res$cluster)
centers.pp <- as.data.frame(kmeanspp.res$centers)
ggplot(iris, aes(x=Petal.Length, y=Petal.Width, color=cluster.pp)) + geom_point() +
geom_point(data=centers.pp, color='coral',size=4,pch=21)+
geom_point(data=centers.pp, color='coral',size=50,alpha=0.2)
```



```
#kmeans
#k-means++
kmeans.res <- kmeans(iris[,-c(1,2,5)], centers = 3)
cluster<-as.factor(kmeans.res$cluster)
centers <- as.data.frame(kmeans.res$centers)
ggplot(iris, aes(x=Petal.Length, y=Petal.Width, color=cluster)) +
geom_point() +
geom_point(data=centers, color='coral',size=4,pch=21)+
geom_point(data=centers, color='coral',size=50,alpha=0.2)
```

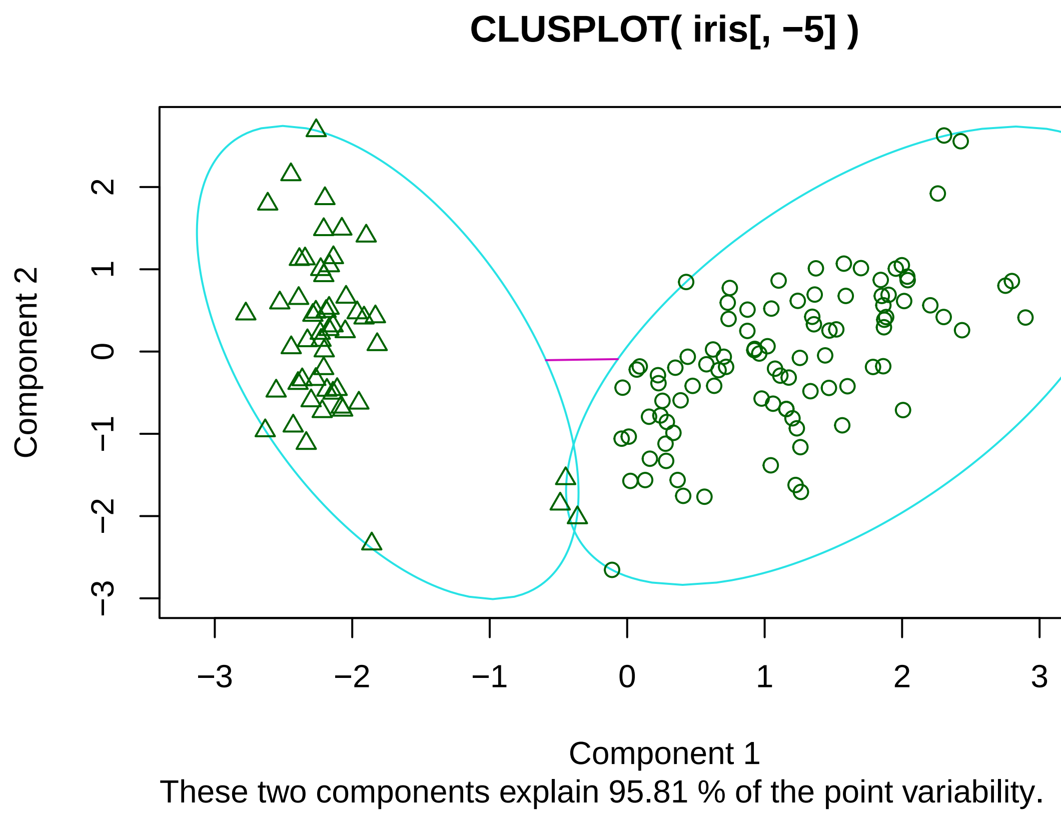


As we can see in the graph, the K-means++ and K-means return the same graph with 3 clusters and the same 3 centers. So when we use these 2 algorithms to cluster the data Iris, we will obtain the same result.

##2.Visualiser les differentes partitions sur les premiers plans d'une analyse en composantes principale.

We use the function clusplot() in the library cluster to visualize 3 different partitions (clusters) on the first plan of PCA (Principal Components Analysis)

```
library(cluster)
clusplot(iris[,-5], clus = kmeanspp.iris.res$cluster)
clusplot(iris[,-5], clus = kmeans.iris.res$cluster)
```



As we can see in the graph, we visualize 3 cluster with 2 components that explain 95.81% information. The K-means++ and K-means has the same graph.