R-PL4

Gabriel López, Sergio Sanz, Álvaro Zamorano

24 de noviembre de 2019

1. Ejercicio realizado en clase.

A partir del siguiente conjunto de calificaciones académicas, pertenecientes a dos grupos de alumnos (mañana y tarde), formados por dos notas: teoría y laboratorio, las notas de teoría y laboratorio tendrán valores entre 0 y 5, realizar un análisis de clasificación no supervisada utilizando el algoritmo **K-Means**.

Alumno	Teoría	Laboratorio
A1	4	4
A2	3	5
A3	1	2
A4	5	5
A5	0	1
A6	2	2
A7	4	4
A8	2	1

En primer lugar se introducirán los datos en forma de matriz y se hará la traspuesta de esta.

```
> m<-matrix(c(4,4,3,5,1,2,5,5,0,1,2,2,4,5,2,1),2,8)
> (m<-t(m))

     [,1] [,2]
[1,]    4    4
[2,]    3    5
[3,]    1    2
[4,]    5    5
[5,]    0    1
[6,]    2    2
[7,]    4    5
[8,]    2    1
```

En segundo lugar se deben seleccionar el número de clusters en los que se van a agrupar los datos, en este caso serán 2. Además es necesario indicar los

centroides iniciales de cada uno de ellos, en este caso son C1{0,1} y C2{2,2}. Todo ello es elegido de forma arbitraria.

Introducimos los centroides en una matriz y se realiza la traspuesta.

```
> c<-matrix(c(0,1,2,2),2,2)
> (c<-t(c))

     [,1] [,2]
[1,]    0    1
[2,]    2    2
```

La función `K-Means` se encuentra en el paquete **stats**. Dicho paquete se carga por defecto al arrancar R; para comprobarlo se hace uso de la función `search()`.

```
> search()

 [1] ".GlobalEnv"         "package:factoextra" "package:ggplot2"
 [4] "package:readr"      "package:foreign"    "package:stats"
 [7] "package:graphics"   "package:grDevices"  "package:utils"
[10] "package:datasets"   "package:methods"    "Autoloads"
[13] "package:base"
```

Por último hacemos uso de la función y obtenemos los centroides finales. Indicamos que el número máximo de iteraciones es 4.

```
> (clasificacionns<-kmeans(m,c,4))

K-means clustering with 2 clusters of sizes 4, 4

Cluster means:
  [,1] [,2]
1 1.25 1.50
2 4.00 4.75

Clustering vector:
[1] 2 2 1 2 1 1 2 1

Within cluster sum of squares by cluster:
[1] 3.75 2.75
 (between_SS / total_SS =  84.8 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
[6] "betweenss"    "size"         "iter"         "ifault"
```

Los resultados obtenidos son los mismos que los de clase, es decir, **C1{1.25,1.5} y C2{4,4.75}.**

A continuación usaremos los clusters obtenidos para separar los datos de la muestra en dos grupos. Para ello se hace uso de la función `cbind` la cuál añade (por delante) una columna a la matriz de datos. Dicha columna se corresponde con la clasificación obtenida, será 1 ó 2 dependiendo del cluster al que pertenezca cada muestra.

```
> (m = cbind(clasificacionns$cluster,m))
```

```
     [,1] [,2] [,3]
[1,]    2    4    4
[2,]    2    3    5
[3,]    1    1    2
[4,]    2    5    5
[5,]    1    0    1
[6,]    1    2    2
[7,]    2    4    5
[8,]    1    2    1
```

Una vez se tiene el cluster al que pertenece cada muestra, se separa la matriz siguiendo el criterio anterior.

```
> mc1=subset(m,m[,1]==1)
> mc2=subset(m,m[,1]==2)
```

Por último, limpiamos la columna introducida para el fin buscado y mostramos los dos conjuntos de datos clusterizados.

```
> (mc1=mc1[,-1])
```

```
     [,1] [,2]
[1,]    1    2
[2,]    0    1
[3,]    2    2
[4,]    2    1
```

```
> (mc2=mc2[,-1])
```

```
     [,1] [,2]
[1,]    4    4
[2,]    3    5
[3,]    5    5
[4,]    4    5
```

Se puede observar que las muestras 3,5,6,8 pertenecen al mismo grupo, mientras que las muestras 1,2,4,7 se encuentran en el restante.

2. Desarrollo por parte del alumno.

En primer lugar hemos realizado un conjunto de funciones que realizan el algoritmo **K-Means**. Los parámetros de esta función deben ser la matriz de muestras y la matriz con los centroides iniciales, al igual que se indica anteriormente. Como resultado, proporciona las coordenadas finales del centroide de cada uno de los clusters y varias listas (número de clusters) incluyendo en cada una de ellas las muestras que pertenece a cada uno de los clusters. Además, esta función, en caso de que los datos tengan 2 dimensiones, nos generará un gráfico donde se muestra la evolución del algoritmo a la largo de las diferentes iteraciones realizadas.

Cabe destacar que el uso de la función se hace mediante matrices.

Procedemos a cargar dichas funciones.

```
> source("./Funciones/KMeans.R")
```

Las principales funciones codificadas son:

```
> calcularMatrizDistancias

function (matrizMuestras, matrizCentroides,dimensiones) {
    sizeCentroides <- length(matrizCentroides)/dimensiones
    sizeMuestras <- length(matrizMuestras)/dimensiones
    x<-c()

    for (i in 1:sizeCentroides) {
        for (j in 1:sizeMuestras)
            x<-c(x,distanciaEuclidea(matrizCentroides[i,],matrizMuestras[j,]))
    }

    m<-matrix(x,nrow=sizeCentroides,ncol=sizeMuestras,byrow=T)

    return(m)
}

> calcularMatrizPertenencia

function (matrizDistancias, nCentroides) {
    sizeDistancias <- length(matrizDistancias)/nCentroides
    x<-c()

    for (i in 1:sizeDistancias) {
        posMinimo <- filaMinimo(matrizDistancias[,i])
        for (j in 1:nCentroides)
            if (j==posMinimo){
                x<-c(x,1)
            } else {
                x<-c(x,0)
            }
```

```
    }

    m<-matrix(x,nrow=nCentroides,ncol=sizeDistancias)

    return(m)
}
```

> muestrasPorCluster

```
function (matrizPertenencia, nCentroides) {
    sizeDistancias <- length(matrizPertenencia)/nCentroides
    finalList<-c()
    for (i in 1:nCentroides){
        finalList<-c(finalList, list(muestrasPorFila(matrizPertenencia[i,])))
    }

    m<-matrix(finalList,nrow=nCentroides,ncol=sizeDistancias)
    return(finalList)
}
```

> obtenerMuestrasSeparadas

```
function (matrizMuestras, muestrasPorCluster, dimensiones) {
    muestras<-t(matrizMuestras)
    sizeMuestras <- length(matrizMuestras)/dimensiones
    separadas<-list()

    for (i in 1:length(muestrasPorCluster)){
        nMuestrasCluster<-length(muestrasPorCluster[[i]])
        temp<-c()
        for (j in 1:sizeMuestras) {
            if (j %in% muestrasPorCluster[[i]]) {
                temp<-c(temp, muestras[,j])
            }
        }

        separadas[[i]]<-matrix(temp,nrow=dimensiones,ncol=nMuestrasCluster)

    }

    return(separadas)
}
```

> obtenerNuevosCentroides

```
function (muestrasSeparadas, dimensiones) {
    sizeMuestras <- length(muestrasSeparadas)
    centros<-c()

    for (i in 1:sizeMuestras){
        matriz<-muestrasSeparadas[[i]]
```

```
        centros<-c(centros,nuevoCentroide(t(matriz),dimensiones))
    }

    return(matrix(centros,nrow=sizeMuestras,ncol=dimensiones,byrow=T))
}
```

> comprobarMatrizPertenencia

```
function (matriz1, matriz2) {
    iguales<-TRUE
    i<-1

    while(iguales && i<=length(matriz1)){
        if (matriz1[i]==matriz2[i]){
            i<-i+1
        } else {
            iguales<-FALSE
        }
    }

    return(iguales)
}
```

Todas ellas se encuentran dentro de un bucle while realizado siempre y cuando cambie la matriz de pertenencia.

Respecto a la parte de **representación** se ha realizado otro conjunto de funciones con dicho fin. La principal de ellas es:

> representar

```
function(muestrasSeparadas,matrizMuestras,matrizCentroides,i){
    colores <- c("red","blue","green","black","purple")

    titulo<-paste("Iteracion ",i)

    m<-muestrasSeparadas[[1]]
    limites <- obtenerLimites(matrizMuestras)
    plot(m[1,],m[2,],pch=1,col=colores[1],xlim=limites$x,ylim=limites$y,
        main=titulo,xlab="X",ylab="Y")

    centroide<-matrizCentroides[1,]
    points(centroide[1],centroide[2],pch=8,col=colores[1])

    indiceColor <- 2

    nClusters<-length(muestrasSeparadas)
    for (i in 2:nClusters) {
        m<- muestrasSeparadas[[i]]
        points(m[1,],m[2,],pch=1,col=colores[i])
```

```
        centroide<-matrizCentroides[i,]
        points(centroide[1],centroide[2],pch=8,col=colores[indiceColor])

        indiceColor <- indiceColor + 1
        if (indiceColor==5) {
            indiceColor <- 1
        }
    }
}
```

Cabe indicar que únicamente se dispone de 5 colores diferentes, es decir, en caso de haber más de 5 clusters, los colores de la gráfica volverían a repetirse.

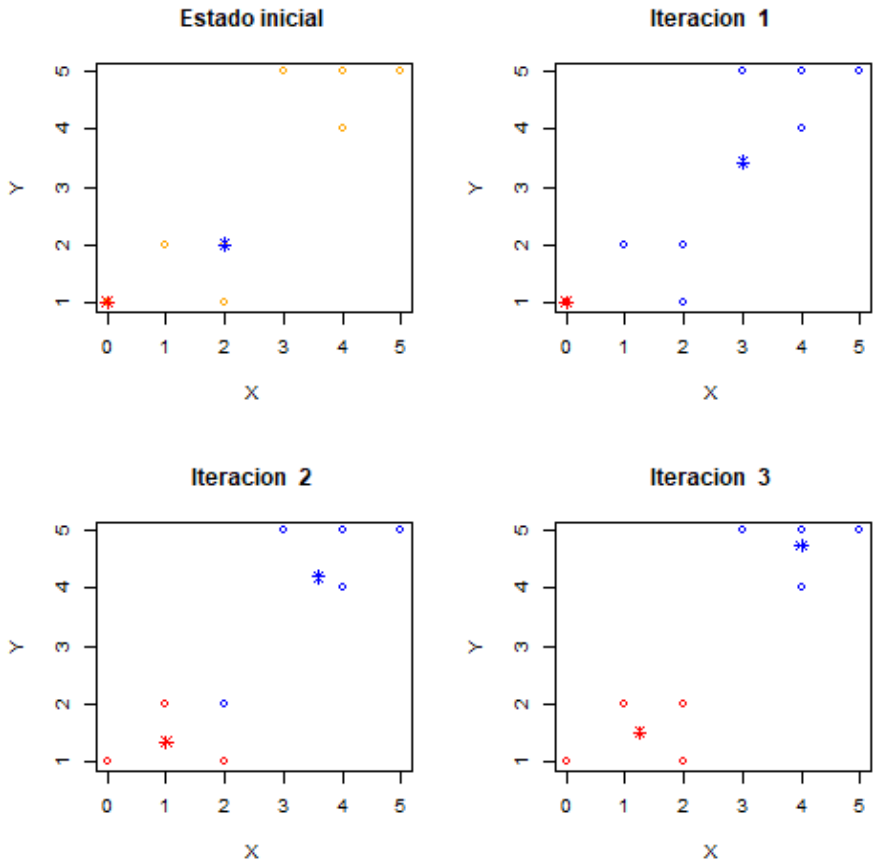
Probamos nuestras funciones con los datos del apartado anterior y obtenemos la evolución del algoritmo.

```
> m<-t(matrix(c(4,4,3,5,1,2,5,5,0,1,2,2,4,5,2,1),2,8))
> c<-t(matrix(c(0,1,2,2),2,2))
> (resultado<-KMeans(m,c,"resultadoKMeans.png"))

$centroides
     [,1] [,2]
[1,] 1.25 1.50
[2,] 4.00 4.75

$muestrasPorCluster
$muestrasPorCluster[[1]]
[1] 3 5 6 8

$muestrasPorCluster[[2]]
[1] 1 2 4 7
```



Se observa que los resultados obtenidos son los mismos que los conseguidos en clase.

Con el uso de estas funciones se realizará una clasterización no jerárquica de datos obtenidos de `Kaggle` sobre información de diferentes países. Procedemos a leer dichos datos.

```
> library("readr")
> datos<-read.csv("./Datos/countries.csv")
```

Se usarán los datos porcentuales pertenecientes a los sectores de agricultura, industria y servicios para realizar k-Means.

País	Id	Sector Primario	Sector Secundario	Sector Terciario
Argentina	1	0.095	0.358	0.547
España	2	0.040	0.295	0.665
Portugal	3	0.053	0.274	0.673
Reino Unido	4	0.005	0.237	0.758
Estados Unidos	5	0.010	0.204	0.787
Rusia	6	0.054	0.371	0.575
Alemania	7	0.009	0.296	0.695
Francia	8	0.022	0.214	0.764
Brasil	9	0.084	0.400	0.516
Italia	10	0.021	0.291	0.688
China	11	0.125	0.473	0.403
Chile	12	0.060	0.493	0.447
Turquía	13	0.117	0.298	0.585
Mexico	14	0.038	0.259	0.702
Canadá	15	0.022	0.294	0.684

```
> datosEvaluar<-c(datos$Agriculture,datos$Industry,datos$Service)
> datosEvaluar<-matrix(datosEvaluar,15,3)
> cIniciales<-matrix(c(0.04,0.08,0.25,0.35,0.60,0.65),2,3)
> (resultado2<-KMeans(datosEvaluar,cIniciales,"resultadoKMeans.png"))

$centroides
           [,1]      [,2]      [,3]
[1,] 0.02444444 0.2626667 0.7128889
[2,] 0.08916667 0.3988333 0.5121667

$muestrasPorCluster
$muestrasPorCluster[[1]]
[1]  2  3  4  5  7  8 10 14 15

$muestrasPorCluster[[2]]
[1]  1  6  9 11 12 13
```

A la vista de los resultados obtenidos, los países se clusterizan de la siguiente manera:

1. España, Portugal, Reino Unido, Estados Unidos, Alemania, Francia, Italia, Mexico, Canada
2. Argentina, Rusia, Brasil, China, Chile, Turquía

Una de las diferencias de los países del primer cluster respecto a los restantes es el mayor porcentaje en el sector servicios y menor en el sector primario.

Con estos mismos datos realizaremos una clusterización **jerárquica** mediante el uso de los paquetes `stats` y `factorextra`. Procedemos a cargar el segundo de ellos.

```
> install.packages("factoextra")
> library(factoextra)
```

Previo a la clusterización es necesario escalar los datos mediante la función `scale`.

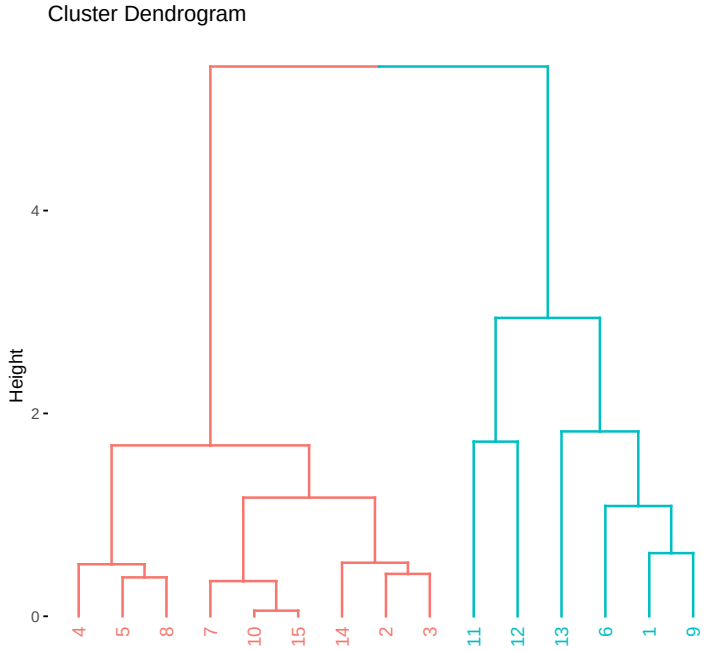
```
> datosCJ<-scale(datosEvaluar)
```

Por último, usando la función `hclust` se realiza dicha clusterización.

```
> hCJ<-hclust(d = dist(x=datosCJ, method="euclidean"), method="complete")
```

Para mostrar los resultados obtenidos se usa una función definida por nosotros y así poder pasar estos a un formato de foto. Es necesario indicar el número de clusters que queremos obtener.

```
> fviz_dend(x=hCJ, k=2)
```



Se puede observar que la clusterización de países se corresponde con la obtenida anteriormente mediante el uso de KMeans.