R-PL4

Gabriel López, Sergio Sanz, Álvaro Zamorano

21 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 primer 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"
 [2] "package:foreign"
 [3] "package:stats"
 [4] "package:graphics"
 [5] "package:grDevices"
 [6] "package:utils"
 [7] "package:datasets"
 [8] "package:methods"
 [9] "Autoloads"
[10] "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"
```

```
[3] "totss"        "withinss"
[5] "tot.withinss" "betweenss"
[7] "size"         "iter"
[9] "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 obtenios 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. 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.

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) {
```

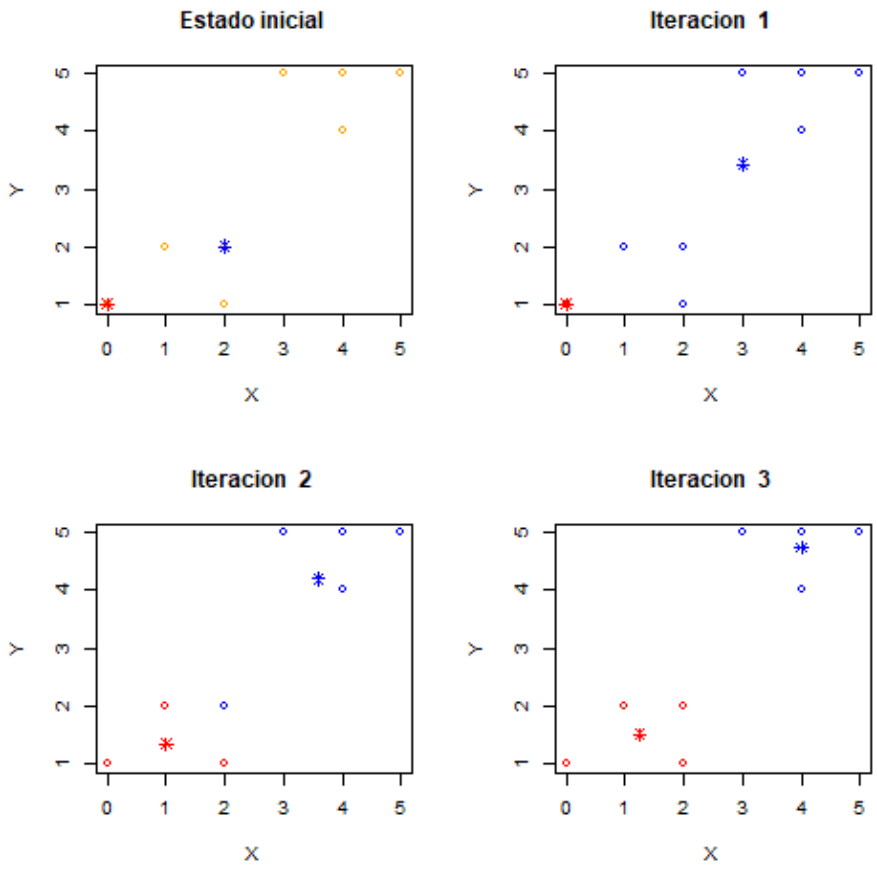
```
            indiceColor <- 1
        }
    }
}
```

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))
> (centroides<-KMeans(m,c))

     [,1] [,2]
[1,] 1.25 1.50
[2,] 4.00 4.75
```



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