

# PECL4 - Fundamentos de la ciencia de datos

Mario Adán Herrero      Alberto González Martínez  
Branimir Stefanov Yanev      Diego Gutiérrez Marco

22 de diciembre de 2020

## **Resumen**

En el siguiente documento se presentan los resultados y la solución de la PECL4 del laboratorio de Fundamentos de la Ciencia de los Datos. En esta ocasión realizaremos una clasificación no supervisada de los datos siguiendo el ejercicio realizado en teoría. Para la resolución del ejercicio utilizaremos el método k-means, que nos permite clasificar estos elementos en función de diferentes centroides.

# Índice

<b>1. Introducción - Clasificación no supervisada</b>	<b>3</b>
<b>2. Apartado 1 - Análisis de clasificación no supervisada</b>	<b>3</b>
<b>3. Apartado 2 - Clasificación no supervisada modificado</b>	<b>4</b>
<b>4. Apartado 3 - Análisis paquete de R</b>	<b>5</b>
4.1. función AgglomerativeHC . . . . .	6
4.2. función getCluster . . . . .	7
4.3. función chebyshevDistance . . . . .	7
4.4. función normalizeWeight . . . . .	7
4.5. función newCluster . . . . .	8
<b>5. Conclusiones</b>	<b>9</b>

## 1. Introducción - Clasificación no supervisada

En esta práctica utilizaremos el método de clasificación no supervisada k-means como algoritmo para realizar la clasificación. Partiendo de una muestra dada, utilizaremos la media aritmética para definir los diferentes centroides de los datos, clasificando así los elementos en varios conjuntos, cada uno asociado a uno de los centroides utilizados.

## 2. Apartado 1 - Análisis de clasificación no supervisada

En este apartado seguiremos el ejercicio de clasificación no supervisada visto en clase, donde clasificamos los datos de la muestra en dos clusters. Se tienen los siguientes datos en forma (x,y):

1. (4, 4)
2. (3, 5)
3. (1, 2)
4. (5, 5)
5. (0, 1)
6. (2, 2)
7. (4, 5)
8. (2, 1)

Lo primero que debemos hacer es cargar los datos de la muestra. Para ello los introducimos directamente en la declaración de la matriz: No necesitamos cargar ninguna librería adicional ya que nos sirve con el paquete base.

```
> #Cargamos datos matriz
> matriz_datos_iniciales<-matrix(c(4,4,
+                               3,5,
+                               1,2,
+                               5,5,
+                               0,1,
+                               2,2,
+                               4,5,
+                               2,1),
+                               # Filas, columnas
+                               2,8)
> matriz_datos_iniciales<-t(matriz_datos_iniciales)
```

Ahora podemos establecer los centroides:

```
> #Establecemos los centroides
> centroides <- matrix(c(0,1,2,2),2,2)
> centroides <- t(centroides)
```

Realizamos la clasificación llamando al método k-means. Con 4 iteraciones es suficiente para clasificar los datos:

```
> clasificacion_kmeans = kmeans(matriz_datos_iniciales, centroides, 4)
```

Al imprimir vemos una tabla cuya primera columna indica el centroide asociado al par de datos, mostrados en la segunda y tercera columna:

```
> matriz_datos_clasificados <- cbind(clasificacion_kmeans$cluster,
+                                         matriz_datos_iniciales)
```

Para separar los datos en función del clúster al que pertenecen, creamos dos vectores y vamos añadiendo datos a cada uno de los clústers.

```
> #Matriz de datos del primer cluster
> matriz_cluster1 <- subset(matriz_datos_clasificados,
+                               matriz_datos_clasificados[,1]==1)
> #Matriz de datos del segundo cluster
> matriz_cluster2 <- subset(matriz_datos_clasificados,
+                               matriz_datos_clasificados[,1]==2)
> matriz_cluster1<- matriz_cluster1[,-1]
> matriz_cluster2<- matriz_cluster2[,-1]
> print(matriz_cluster1)

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

> print(matriz_cluster2)

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

### 3. Apartado 2 - Clasificación no supervisada modificado

En este apartado, desarrollaremos nuestro propio enunciado y la solución de este, realizando un análisis de clasificación no supervisada con R.

Vamos a realizar la clasificación de los clientes de una empresa, con el fin de realizar estrategias de marketing enfocadas a sus clientes más fieles. Los datos que tenemos serán las edades de los clientes y el total de compras que han realizado.

```
> # Introducimos los datos
> clientes <- data.frame(edad = c(21,35,26,53,23,43,32,42,45,31,19,53),
+                           compras = c(5,4,6,3,8,3,4,4,2,2,5,1))
```

Realizamos la clasificación por medio del algoritmo k-means, con 3 clústers.

```
> # Aplicamos k-means  
> resultado_kmeans <- kmeans(clientes, 3)
```

Visualizamos, en una tabla, resultado del algoritmo k-means. Podemos observar que la primera columna representa el clúster y las otras dos columnas, los datos (edad y total de compras).

```
> matriz_resultado <- cbind(resultado_kmeans$cluster, clientes)
```

Separamos, en vectores, la matriz resultante del algortimo k-means, según el clúster al que pertenezcan los datos.

```
> m1 <- subset(matriz_resultado,  
+                         matriz_resultado[,1]==1)  
> m2 <- subset(matriz_resultado,  
+                         matriz_resultado[,1]==2)  
> m3 <- subset(matriz_resultado,  
+                         matriz_resultado[,1]==3)  
> m1<- m1[,-1]  
> m2<- m2[,-1]  
> m3<- m3[,-1]  
> print(m1)  
  
      edad compras  
2     35      4  
7     32      4  
10    31      2  
  
> print(m2)  
  
      edad compras  
4     53      3  
6     43      3  
8     42      4  
9     45      2  
12    53      1  
  
> print(m3)  
  
      edad compras  
1     21      5  
3     26      6  
5     23      8  
11    19      5
```

## 4. Apartado 3 - Análisis paquete de R

Para la tercera y última parte de la práctica, analizaremos y utilizaremos el paquete LearnCrust, el cual, permite hacer cálculos mediante algoritmos de clustering jerárquico.

Para poder utilizar el paquete primero lo instalamos y cargamos en nuestro entorno de trabajo:

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

A continuación, haremos uso de algunas funciones de este paquete, las cuales son las siguientes:

1. AgglomerativeHC
2. getCluster
3. chebyshevDistance
4. normalizeWeight
5. newCluster

Prueba de funciones del paquete con el enunciado del ejercicio del primer apartado. (Meter mas funciones y comentar lo que hacen)

#### 4.1. función AgglomerativeHC

Esta función ejecuta el algoritmo de agrupamiento jerárquico especificando la distancia y tipo de acercamiento. El algoritmo convierte los datos que creamos en un objeto, crea los clusters y calcula una matriz de distancias con los clusters creados aplicando la distancia y acercamiento indicados a la función. Agrupa los clusters en uno nuevo y repite el mismo proceso hasta que existe un único cluster.

```
> matriz<-matrix(c(4,4,3,5,1,2,5,5,0,1,2,2,4,5,2,1),2,8)
> agglomerativeHC(matriz, 'EUC', 'MAX')

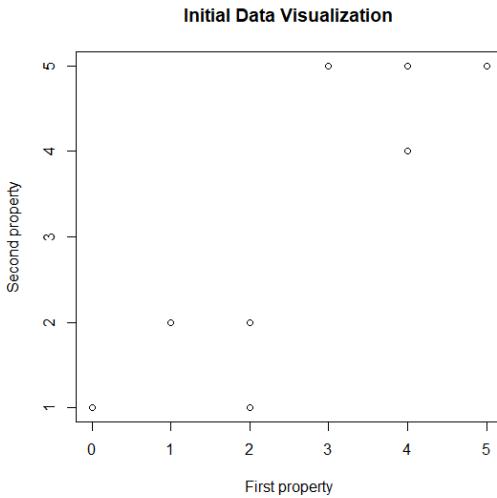
$dendrogram
Number of objects: 2

$clusters
$clusters[[1]]
  X1 X2
1   4   3

$clusters[[2]]
  X1 X2
1   4   5

$clusters[[3]]
  X1 X2
1   4   3
2   4   5

$groupedClusters
  cluster1 cluster2
1           1           2
> lista<-toList(matriz)
```



#### 4.2. función getCluster

Esta función forma parte del método de agrupamiento jerárquico. Usando el valor distancia que le pasamos a la función, la función obtiene el cluster con la menor distancia.

```
> distance = 3
> getCluster(distance,matriz)
```

#### 4.3. función chebyshevDistance

Esta función forma parte del método de agrupamiento jerárquico. Calcula la distancia de Chebyshev entre dos clusters.

```
> x<-c(1,2,3,4)
> y<-c(5,6,7,8)
> matrizX<-matrix(x,ncol=4)
> matrizY<-matrix(y,ncol=4)
> chebyshevDistance(x,y)

[1] 4

> chebyshevDistance(matrizX,matrizY)

[1] 4
```

#### 4.4. función normalizeWeight

Esta función permite normalizar pesos. Si el primer parámetro toma valor TRUE, la función convertirá cualquier valor de paso como un valor "[0:1]". En caso de que el primer parámetro sea FALSE, la función no hará ningún cambio y los pesos serán los mismos.

```

> datos <- data.frame(matrix(c(1:10), ncol = 2))
> weight1 <- c(6.3, 3.2)
> weight2 <- c(4.5, 3.0)
> normalizeWeight(TRUE, weight1, datos)
[1] 0.6631579 0.3368421
> normalizeWeight(FALSE, weight1, datos)
[1] 6.3 3.2
> normalizeWeight(TRUE, weight2, datos)
[1] 0.6 0.4
> normalizeWeight(FALSE, weight2, datos)
[1] 4.5 3.0

```

#### 4.5. función newCluster

Esta función forma parte del método de agrupamiento jerárquico. Crea un cluster a partir de dos clusters dados. Añade el nuevo cluster a una lista.

```

> data <- c(1:10)
> list <- toList(data)
> clusters <- c(1,2)
> newCluster(list,clusters)

[[1]]
 [,1] [,2] [,3]
[1,]    1    2    0

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

[[3]]
 [,1] [,2] [,3]
[1,]    5    6    1

[[4]]
 [,1] [,2] [,3]
[1,]    7    8    1

[[5]]
 [,1] [,2] [,3]
[1,]    9   10    1

[[6]]
 [,1] [,2] [,3]
[1,]    1    2    1
[2,]    3    4    1

```

## **5. Conclusiones**

En esta práctica, se han realizado análisis de clasificación no supervisada de datos con R, aplicando el algoritmo k-means, algoritmo de clusterización encargado de agrupar objetos en k grupos basándose en sus características. También, se ha aplicado y analizado el paquete de R LearnClust, un paquete que contiene funciones que aplican algoritmos clásicos del agrupamiento jerárquico además de funciones adicionales que sirve para explicar estos algoritmos paso a paso.