Modelos de Soporte No Supervisado

Partitioning Clustering:

K-medoids

K-medoids

- K-medoids es una técnica de clustering particional
- En k-medoids cada cluster es representado por un elemento del cluster, estos puntos (elementos) son los medoids
- El término medoid se refiere a un objeto dentro de un clúster para el cual la disimilitud promedio entre él y todos los demás miembros del clúster es mínima

K-medoids

- Corresponde a el punto más céntrico del grupo.
- Estos objetos (uno por grupo) pueden ser considerados como un ejemplo representativo de los miembros de ese grupo
- Recordemos que, en k-means clustering, el centroide de un cluster se calcula como el valor medio de todos los puntos de datos en el clúster.

K-medoids

- K-medoid es una alternativa robusta a la agrupación por k-means.
- K-medoid es menos sensible al ruido y a los valores atípicos
- Debido a que usa los medoids como centros de conglomerados en lugar de los centroides (utilizados en k-medias).

K-medoids

- K-medoids requiere que se especifique el número de clústeres que se generarán (como en k-means clustering)
- Un método útil para determinar el número óptimo de clusters es el método de siluetas (silhouette method) que veremos más adelante.
- Los métodos de agrupamiento k-medoids más comunes son los PAM algorithm (Particionamiento Alrededor de Medoids, Kaufman y Rousseeuw, 1990).

PAM (Partitioning Around Medoids)

- El algoritmo PAM se basa en la búsqueda de k objetos representativos o medoids entre las observaciones del conjunto de datos
- Después de encontrar un conjunto de k medoids, los conglomerados se construyen asignando cada observación al medoid más cercano
- Luego, cada medoid *m* seleccionado y cada punto de datos no medoid se intercambian y se calcula la función objetivo

PAM (Partitioning Around Medoids)

- La función objetivo corresponde a la suma de las diferencias de todos los objetos con su medoid más cercano.
- El paso *swap* intenta mejorar la calidad de la agrupación mediante el intercambio de objetos seleccionados (medoids) y objetos no seleccionados.
- Si la función objetivo se puede reducir intercambiando un objeto seleccionado con un objeto no seleccionado, entonces se lleva a cabo el intercambio.

PAM (Partitioning Around Medoids)

Algoritmo PAM

1. Seleccione k objetos para convertirse en los medoids, o en caso de que estos objetos se hayan proporcionado serán los medoids
2. Calcule la matriz de disimilaridad
3. Asignar cada objeto a su medoid más cercano
4. Para cada búsqueda de clúster si alguno de los objetos del clúster disminuye el coeficiente de disimilaridad promedio; si lo hace, selecciona la entidad como medoid para este grupo
5. Si al menos un medoid ha cambiado, vaya a (3), de lo contrario, finalice el algoritmo.

PAM (Partitioning Around Medoids)

Como se mencionó anteriormente, el algoritmo PAM funciona con una matriz de disimilitud, y para calcular esta matriz el algoritmo puede usar dos métricas:

1. Distancias euclidianas
2. Distancia de Manhattan

Aplicación de K-medoids en R

Trabajaremos con la misma base de datos USArrests

- data**("USArrests")** # Cargamos los datos
- **df <-** scale**(USArrests)** # Estandarizamos
- head**(df, n = 3)** # Muestra las primeras tres filas

Aplicación de K-medoids en R

Necesitamos las siguiente función

pamk() #Esta función requiere definir k
pam**(x, k, metric = "euclidean", stand = FALSE)**

Y las siguientes librerías:

“cluster”
“factoextra”

Aplicación de K-medoids en R

pam**(x, k, metric = "euclidean", stand = FALSE)**

Donde:

x:

- Matriz de datos numéricos o marco de datos numéricos
- Matriz de disimilaridad: en este caso, x suele ser la salida de daisy () o dist()

k: la cantidad de clusters

• métrica: Las opciones disponibles son "euclidianas" y "Manhattan".

• stand: valor lógico; si es verdadero, las variables (columnas) en x están estandarizadas antes de calcular las disimilaridades

Aplicación de K-medoids en R

- library**(cluster)**
- library**(factoextra)**

Estimamos el número de cluster a utilizar:

- fviz_nbclust**(df, pam, method = "silhouette")+**

theme_classic**()**

El número óptimo de clusters k es el que maximizar el *“average sillhouette”* sobre un rango de valores posibles para k (Kaufman y Rousseeuw [1990]).

Aplicación de K-medoids en R

¿Qué hace el algoritmo Silhouette?

```
library(cluster)
k.max <- 15
data <- iris.scaled
sil <- rep(0, k.max)

# Ejemplo con For
# k = 2 to k = 15
for(i in 2:k.max){
  km.res <- kmeans(data, centers = i, nstart = 25)
  ss <- silhouette(km.res$cluster, dist(data))
  sil[i] <- mean(ss[, 3])
}
# Gráfica del average silhouette
plot(1:k.max, sil, type = "b", pch = 19,
     frame = FALSE, xlab = "Number of clusters k")
abline(v = which.max(sil), lty = 2)
```

Silhuette coefficient[1]

For each object x_i define:

- s_i-mean distance to objects in the same cluster
- d_i-mean distance to objects in the next nearest cluster

Silhouette coefficient for x_i:

$$Silhouette_i = \frac{d_i - s_i}{\max\{d_i, s_i\}}$$

Silhouette coefficient for $x_1, \ldots x_N$:

$$Silhouette = \frac{1}{N} \sum_{i=1}^{N} \frac{d_i - s_i}{\max\{d_i, s_i\}}$$

- Advantages
  - The score is bounded between -1 for incorrect clustering and +1 for highly dense clustering.
  - Scores around zero indicate overlapping clusters.
  - The score is higher when clusters are dense and well separated.

Aplicación de K-medoids en R

- pam.res <- pam(df, 2)
- print(pam.res)
- dd <- cbind(USArrests, cluster = pam.res$cluster)
- head(dd, n = 3)

El resultado impreso muestra:

- Los medoids del clúster: una matriz, cuyas filas son los medoids y las columnas son variables
- El vector de agrupamiento: un vector de números enteros (de 1: k) que indica el clúster al que cada punto está asignado

Aplicación de K-medoids en R

- pam.res$medoids
- head**(pam.res$clustering)**

Para visualizar los resultados de la partición usaremos la función fviz_cluster() [factoextra]

Y representamos los grupos mediante un gráfico de dos dimensiones. Si los datos contienen más de dos variables podemos usar PCA para reducir la dimensión.

Aplicación de K-medoids en R

- fviz_cluster**(pam.res, palette =** c**("#00AFBB", "#FC4E07"),** # Paleta de colores

  ellipse.type = "t", # Elipse de concentración

  repel = TRUE, # Evita el overplotting label

  ggtheme = theme_classic**()**

)

Conclusiones

- El algoritmo K-medoids, PAM, es una alternativa robusta a k-means para particionar un conjunto de datos en grupos.
- En el método k-medoids, cada grupo está representado por un objeto seleccionado dentro del cluster.
- Los objetos seleccionados se denominan medoids y corresponden a los puntos más céntricos localizados dentro del cluster.
- El algoritmo PAM requiere que el usuario conozca los datos e indique los datos apropiados y número de clusters que se producirán
- Esto puede ser estimado usando la función fviz_nbclust [en el paquete factoextra].

Conclusiones

- La función R pam () [cluster] se puede usar para calcular el algoritmo PAM.
- El formato simplificado es pam (x, k), donde "x" es la información y k es la cantidad de clusters
- Después, realizando la agrupación de PAM, la función R fviz_cluster () [factoextra package]
- se puede usar para visualizar los resultados.
- El formato es fviz_cluster (pam.res), donde pam.res es el resultado de PAM.