Clustering PAM - K-Medoides

Método Particional

A diferença básica em relação ao k-means está na utilização de uma das observações do conjunto original como elemento representativo, chamado medoid, localizado mais no meio do cluster, ao invés da tradicional escolha do centro de gravidade do grupo.

Nos algoritmos k-medoids os grupos são definidos como subconjuntos de pontos que estão mais próximos dos seus elementos representativos, que são chamados de medoides. O medoid pode ser definido como o objeto do grupo, cuja soma das dissimilaridades a todos os objetos do mesmo grupo seja mínima, o que é equivalente a média dessa soma ser mínima.

Assim como o k-means, esse método é bem adequado na hipótese dos grupos serem esféricos, ocupando cada medoide uma observação mais central do grupo. Entretanto, esse método é menos sensível a ruídos de que o k-means, pois não avalia os desvios das observações aos centróides ao quadrado, como é feito no k-means. Além disso, possui a característica de ser capaz de lidar com qualquer tipo de atributo.

O algoritmo Partitioning Around Medoids (PAM) é um algoritmo clássico da família dos métodos k-medoids. Passos que o algoritmo reliza: 1. Inicialização: seleciona aleatoriamente q das m observações do conjunto de dados como medoids; 2. Associa cada observação do conjunto de dados ao medoid mais próximo usando distânacia métrica válida. 3. Para cada medoid: Para cada não medoid: calcule o custo total da configuração do grupo como se esta observação fosse o medoid. 4. selecionar a configuração com o menor custo. 5. Repita os passos 2-5 até que não haja mudança nos medoids.

E, relação ao tempo de processamento, o k-medoid é menos eficiente do que o k-means, pois o calculo do medoid é mais custoso computacionamente do que o cáculo do centro de gravidade, resultando num maior tempo de processamento.

Algoritmo Utiliza a Função de distância **Manhattan**

```
manhattan.dist<-function(x, y){
    sum(abs(x-y))
}
```

Os dados são os próprios dados medoides são uma lista de índices de todos os medoides (que é um conjunto de dados reais).

```
distanceToMedoid<-function(data, medoids){
    distMatrix<-matrix(nrow=nrow(data), ncol=length(medoids))
    for(i in 1:length(medoids))
        distMatrix[,i] = apply(data, 1, manhattan.dist, y=data[medoids[i],])

    distMatrix
}
```

clustering função Partioning Around Medoids (**PAM**).

```
pam.cluster<-function(data, k){
  # inicializa os centros
    centers = sample(nrow(data),k)
    # calcula a disntaância
    distMat = distanceToMedoid(data, centers)

    clusters = apply(distMat, 1, which.min)

    min.dist = sum(apply(distMat, 1, min))
```

```
    while(TRUE){
        new.centers = c()
        for(i in 1:k){
            #DOHHH...não tão inteligente nova escolha de centro...
            new.centers[i] = sample(which(clusters == i), 1)
        }

        distMat = distanceToMedoid(data, new.centers)
        new.clusters = apply(distMat, 1, which.min)
        new.dist = sum(apply(distMat, 1, min))

        if(new.dist < min.dist){
            clusters = new.clusters
            centers = new.centers
            min.dist = new.dist
            cat("-")
        }else if(new.dist == min.dist){
            break
        }
    }

    resp<-list()
    resp$cluster=clusters
    resp$center=centers

    resp
}
```

Testando o algoritmo no conjunto de dados *Iris*

```
# read the dataset
iris <- read.csv('../data/Iris.csv')

# pegando somente as features do meu conjunto de dados, variável contínua
iris.features <- iris[,2:5]

# era bom escolher o K pelo método do cotovelo

# obtem os centroides do meu cluster com o agrupamento kmeans
cluster <- pam.cluster(iris.features, 2)
```

```
## ----
```

HEAD

```
head(cluster)
```

```
## $cluster
##   [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
##  [36] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 1 2 2 1 2 2 2 2 2 2 2 2 2
##  [71] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 1 2 2 2 2 2 2
## [106] 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
## [141] 2 2 2 2 2 2 2 2 2 2
##
## $center
```

```
## [1]  40 128
```

Plotando o gráfico.

```
library("cluster")
clusplot(iris.features, cluster$cluster, color = TRUE, shade= TRUE, labels = 2, lines = 0)
```