Implementing the K-means algorithm in R

Zhen Yang

2022-11-27

The K-means method

K-means is an unsupervised machine learning clustering algorithm. For a given data, we assume that there are k groups (i.e. k clusters), and each cluster is represented by its center which corresponds to the mean of points assigned to the cluster. K-means clustering partitions the data points into k groups where each data point belongs to only one group. The result of clustering is to maximise the homogeneity within each group (i.e. objects within the same cluster are as similar as possible) and maximise the heterogeneity between the different groups (i.e. objects from different clusters are as dissimilar as possible).

Implementation in R

My K-means algorithm works as follows:

Step 1: Specify the number of clusters k and maximum number of iterations.
Step 2: Select randomly k objects from the data set as the initial cluster centers.
Step 3: Assigns each data point to its closest centroid based on the Euclidean distance between the object and the centroid. Then determine which cluster each object belongs to.
Step 4: For each of the k clusters update the cluster center by calculating the new mean values of all the data points in the cluster.
Step 5: Iterate steps 3 and 4 until there is no change to the cluster centers or the maximum number of iterations is reached.
Step 6: Output the result.

The formulas used in my algorithm are shown below:
Euclidean distance:

$$d(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

The total within-group sum of squares SSW:

$$SSW = \sum_{k=1}^{K}\sum_{i\in G_k}(x_i - \hat{\mu}_k)^T(x_i - \hat{\mu}_k)$$

The between-group sum of squares SSB:

$$SSB = \sum_{k=1}^{K} n_k(\hat{\mu}_k - \hat{\mu}_0)^T(\hat{\mu}_k - \hat{\mu}_0)$$

The total sum of squares:

$$SST = \sum_{i=1}^{n}(x_i - \hat{\mu}_0)^T(x_i - \hat{\mu}_0)$$

Results and Discussion

We use the Iris flower data as simulated data and assume that the number of clusters is 3. By comparing my K-means function with the built-in functions, we find that the cluster, centers, totss, withinss, tot.withinss, betweenss and size are very similar or even the same as the values calculate by the built-in functions. The inconsistency is due to the random selection of the initial cluster centers. By plotting the within and between cluster variation in dependence of K, we can see that $K = 3$ clusters seem appropriate since more than 3 clusters will not substantially reduce unexplained variation.

```
data <- iris[,1:4] # Load iris data
k=3  # Specify the number of clusters
max.iter=100  # Maximum number of iterations is 100
rows <- nrow(data)  # Get the number of rows
cols <- ncol(data)  # Get the number of columns
iter = 0  # Number of iterations

nameMatrix <- matrix(0,rows,2)
# The first column is the cluster in which each data is located,
# and the second column is the distance of each data to its centroid
centers <- matrix(0,k,cols)  # For storage cluster centers

# Set the initial cluster centers by generating random numbers
randomNum <- as.matrix(sample(1:rows,k))
for(i in 1:k){
  nameMatrix[randomNum[i],1] <- i
  centers[i,] <- as.matrix(data[randomNum[i],])
  centers <- matrix(centers,k,cols)
}

changed <- TRUE
while(changed){
  if(iter >= max.iter)
    break
  changed <- FALSE

  # For each data point, calculate the distance to each cluster center
  # Assign each observation to their closest centroid
  # Determine which cluster each object belongs to
  for(i in 1:rows){
    initialDistance <- 10000
    formerCluster <- nameMatrix[i,1]
    for(j in 1:k){
      currentDistance <- (sum((data[i,]-centers[j,])^2))^0.5
      if(currentDistance < initialDistance){
        initialDistance <- currentDistance
        nameMatrix[i,1] <- j
        nameMatrix[i,2] <- currentDistance
      }
    }
    # If the cluster to which the data belongs has changed,
    # set 'changed' to TRUE and the algorithm continues
    if(formerCluster!=nameMatrix[i,1])
      changed <- TRUE
  }
```

```
  # For each of the k clusters update the cluster center
  # by calculating the new mean values of all the data points in the cluster.
  for(i in 1:k){
    clusterMatrix <- as.matrix(data[nameMatrix[,1]==i,])
    if(nrow(clusterMatrix)>0){
      centers[i,] <- colMeans(clusterMatrix)
    }
    else{
      centers[i,] <- centers[i,]
    }
  }
  iter = (iter+1)
}
colnames(centers) <- colnames(data)

# Calculate the total sum of squares
totss <- sum((t(data[,])-colMeans(data))^2)

# Calculate the total within-group sum of squares
withinss <- numeric()
for(i in 1:k){
  withinss[i] <- sum((t(data[nameMatrix[,1]==i,])-(centers[i,]))^2)
}
tot.withinss <- sum(withinss)

# Calculate the between-group sum of squares
betweenss <- 0
for(i in 1:k){
  betweenss <- betweenss +
    sum(nrow(data[nameMatrix[,1]==i,])*(rowMeans(t(data[nameMatrix[,1]==i,]))
                                        -colMeans(data))^2)
}

# Calculate the size of each cluster
size <- aggregate(nameMatrix[,1], by=list(nameMatrix[,1]),length)[,2]

# List all the components
result <- list(cluster = nameMatrix[,1],centers = centers,totss = totss,
               withinss = withinss,tot.withinss = tot.withinss,
               betweenss = betweenss,size = size,iter = iter)

result # output
```

```
## $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 1
##  [38] 1 1 1 1 1 1 1 1 1 1 1 1 2 3 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
##  [75] 3 3 3 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 3 2 2 2 2 3 2 2 2 2
## [112] 2 2 3 3 2 2 2 2 3 2 3 2 3 2 2 3 3 2 2 2 2 2 3 2 2 2 2 3 2 2 2 3 2 2 2 3 2
## [149] 2 3
##
## $centers
##      Sepal.Length Sepal.Width Petal.Length Petal.Width
## [1,]     5.006000    3.428000     1.462000    0.246000
## [2,]     6.853846    3.076923     5.715385    2.053846
```

```
## [3,]     5.883607    2.740984     4.388525    1.434426
##
## $totss
## [1] 681.3706
##
## $withinss
## [1] 15.15100 25.41385 38.29082
##
## $tot.withinss
## [1] 78.85567
##
## $betweenss
## [1] 602.5149
##
## $size
## [1] 50 39 61
##
## $iter
## [1] 12
```

```
# Compare with built-in functions for K-means
kmeans.out <- kmeans(data,3)
kmeans.out$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
##  [38] 1 1 1 1 1 1 1 1 1 1 1 1 1 3 3 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
##  [75] 3 3 3 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 3 2 2 2 2 3 2 2 2 2
## [112] 2 2 3 3 2 2 2 2 3 2 3 2 3 2 2 3 3 2 2 2 2 2 3 2 2 2 2 3 2 2 2 3 2 2 2 3 2
## [149] 2 3
```

```
kmeans.out$centers
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1     5.006000    3.428000     1.462000    0.246000
## 2     6.850000    3.073684     5.742105    2.071053
## 3     5.901613    2.748387     4.393548    1.433871
```

```
kmeans.out$totss
```

```
## [1] 681.3706
```

```
kmeans.out$withinss
```

```
## [1] 15.15100 23.87947 39.82097
```

```
kmeans.out$tot.withinss
```

```
## [1] 78.85144
```

```
kmeans.out$betweenss
```

```
## [1] 602.5192
```

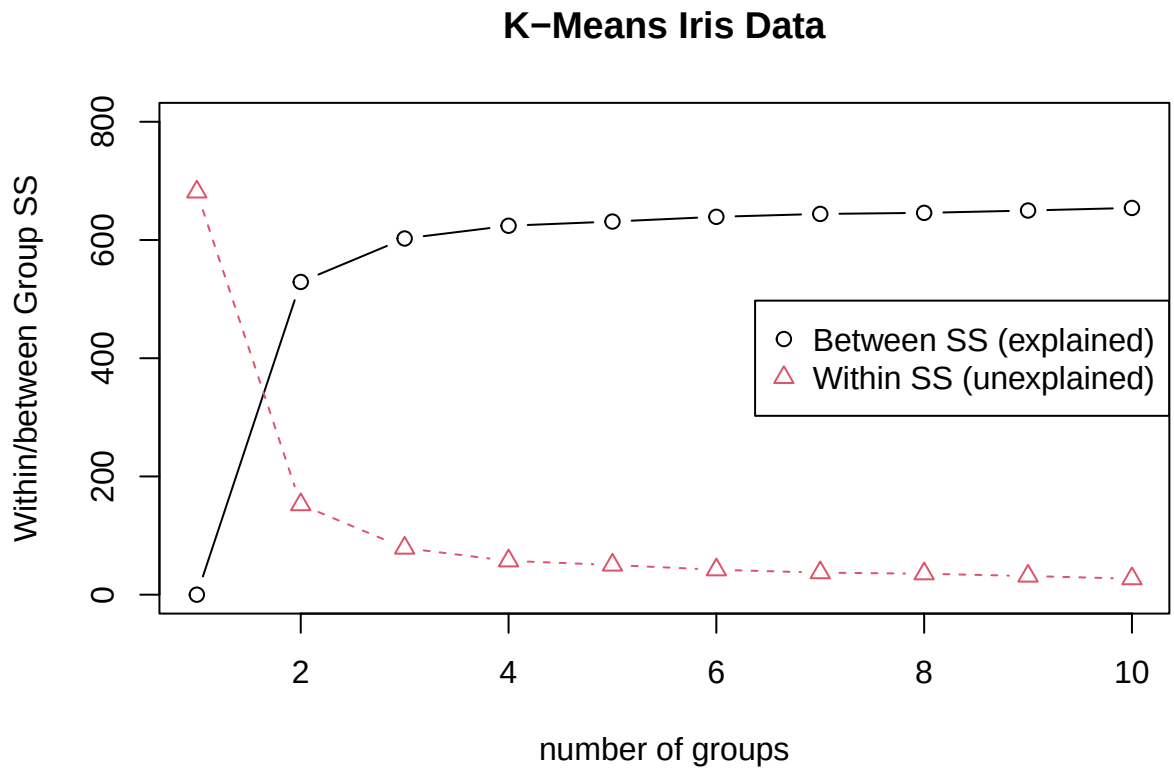
```
kmeans.out$size
```

```
## [1] 50 38 62
```

```
kmeans.out$iter
```

```
## [1] 3
```

```r
# Within and between cluster variation in dependence of K
kmax=10
bvec=numeric(kmax)
wvec=numeric(kmax)
for (k in 1:kmax)
{
  kmeans.out = kmeans(data, k)
  bvec[k] = kmeans.out$betweenss
  wvec[k] = kmeans.out$tot.withinss
}
plot(1:kmax, bvec, type="b", ylim=c(0, 800), xlab="number of groups",
     ylab="Within/between Group SS", main="K-Means Iris Data")
points(1:kmax, wvec, type="b", col=2, lty="dashed", pch=2)
legend("right", c("Between SS (explained)", "Within SS (unexplained)"), col=c(1,2), pch=c(1,2))
```



References

1:https://en.wikipedia.org/wiki/Euclidean_distance
2:https://uc-r.github.io/kmeans_clustering
3:https://towardsdatascience.com/k-means-clustering-algorithm-applications-evaluation-methods-and-drawbacks-aa03e644b48a