Handmade K-Means Clustering

Álvaro Orgaz Expósito

THEORY

The aim of the *K-Means Clustering* algorithm is to create K groups or clusters in the data using its features $x = \{x_1, \dots, x_P\}$. This is an *unsupervised machine learning* algorithm because the aim is not to predict a target variable y, but exploring the data.

Firstly, let's see the main steps of this *unsupervised machine learning* algorithm.

1. Define the input data $x_{[NxP]}$ and K or number of desired clusters.
2. For each cluster K initialize its *centroid*, which means the center of each cluster as a P dimensional vector.
$$C_k = \begin{bmatrix} C_1^k & \cdots & C_P^k \end{bmatrix} \quad with \quad 1 \leq k \leq K$$
3. For each N observation in the data $x_{[NxP]}$, compute the mathematical distance (*Euclidean* in this paper) to each K *centroid*.
4. For each N observation in the data $x_{[NxP]}$, assign it to the cluster with the nearest *centroid* (lower mathematical distance). Then, the result is K disjoint splits of the data x.
$$x = \bigcup_{k=1}^{K} x^k \quad with \quad x^k = \{x_1^k, \dots, x_P^k\}$$
5. For each K clusters, update its *centroid* as the mean of all observations assigned to this cluster by variabes.
$$C_k^{new} = \begin{bmatrix} mean(x_1 \in x^k) & \cdots & mean(x_P \in x^k) \end{bmatrix} \quad with \quad 1 \leq k \leq K$$
6. Repeat steps 3, 4 and 5 until no relocations occur in step 4. It means when all the observations in the data are assigned to the same cluster than the actual and then any *centroid* change.

CODE

The data used is the popular dataset *iris*. Let's create 3 groups or clusters in data using its features: *Petal.Length* and *Petal.Width*. In the paper *Handmade_KNN.pdf*, the *KNN* algorithm tries to predict the variable *Species* in *iris* dataset (which has 3 categories) using the same features. But now the aim is to create 3 clusters only using the features (without *Species*), and let's see if the observations in each cluster correspond to the same category in *Species* variable.

Note: It is important to *normalize* (every observation minus the feature minimum value divided by the feature range) the features so that their units do not affect the mathematical distance. For example, if you compute the mathematical *Euclidean* distance from a person $A = (1.80m, 70kg)$ and another $B = (1.9m, 80kg)$ the result is
$$distance(A, B) = \sqrt{(1.8 - 1.9)^2 + (70 - 80)^2} = 10$$

But if you change the units to $A = (180cm, 70000g)$ and $B = (190cm, 80000g)$ the result is
$$distance(A, B) = \sqrt{(180 - 190)^2 + (70000 - 80000)^2} = 10000$$

Let's define the input data $x_{[NxP]}$. *Note*: The function *duplicated* is used to delete the observations with duplicated values in the needed variables.

```r
duplicates <- duplicated(iris[,c("Petal.Length","Petal.Width")])
x <- iris[!duplicates,c("Petal.Length","Petal.Width")]
range_x1 <- max(x[,1])-min(x[,1])
range_x2 <- max(x[,2])-min(x[,2])
x[,1] <- (x[,1]-min(x[,1]))/range_x1
x[,2] <- (x[,2]-min(x[,2]))/range_x2
```

Let's create the *KMeans* algorithm function.

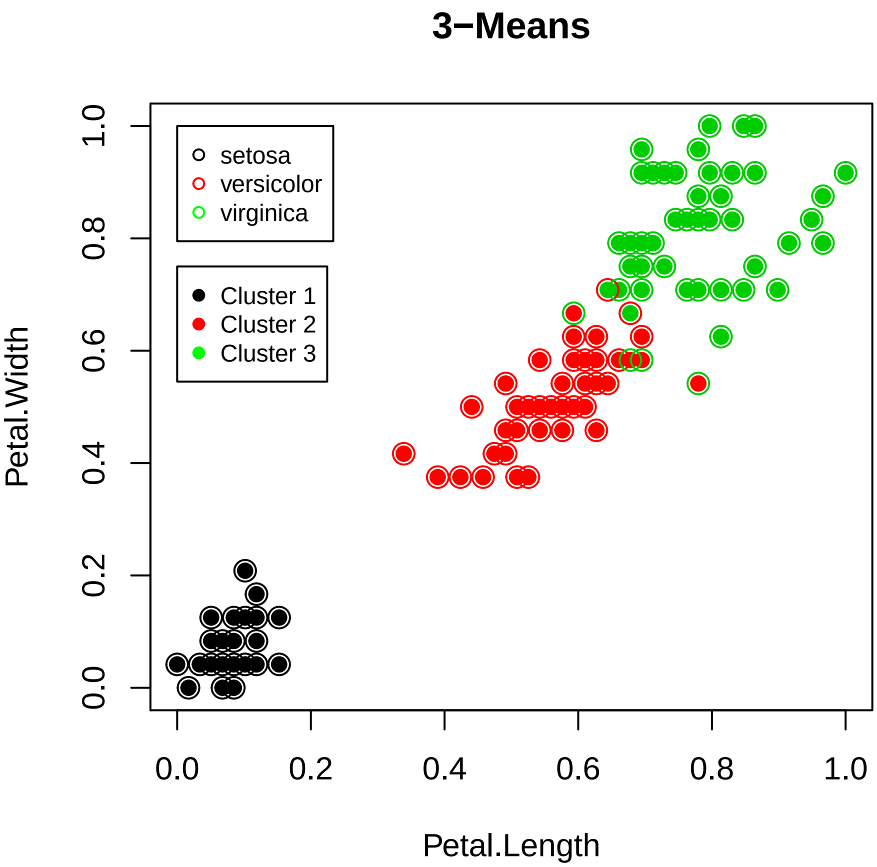
```r
KMeans <- function(x,K,centroids){
  # Define function for mathematical distance between observations
  euclidean <- function(a,b){return(sqrt(sum((a-b)^2)))}
  # Iteration until no relocations occur
  clusters <- rep(0,nrow(x))
  any_relocation <- TRUE
  while(any_relocation){
    any_relocation <- FALSE
    # Iterate all data observations
    distances <- matrix(NA,nrow=nrow(x),ncol=K)
    for(i in 1:nrow(x)){
      # Iterate all clusters and compute the distance from its centroid to observation i
      for(k in 1:K){
        distances[i,k] <- euclidean(x[i,],centroids[k,])
      }
      # Assign the data observation i to the cluster with the nearest centroid (if change)
      nearest <- which.min(distances[i,])
      if(nearest!=clusters[i]){
        clusters[i] <- nearest
        any_relocation <- TRUE
      }
    }
    # Update centroids
    for(k in 1:K){
      for(p in 1:ncol(x)){
        centroids[k,p] <- mean(x[clusters==k,p])
      }
    }
  }
  return(clusters)
}
```

Now, let's apply the *KMeans* function with $K = 3$. Note that we need to initialize the *centroids*, and we will use the means of *Petal.Length* and *Petal.Width* by categories in *Species* (3 equal to K).

```r
K <- 3
centroids <- matrix(NA,nrow=K,ncol=ncol(x))
species <- iris[!duplicates,"Species"]
for(k in 1:K){
  for(p in 1:ncol(x)){
    centroids[k,p] <- mean(x[species==unique(species)[k],p])
  }
}
clusters <- KMeans(x,K,centroids)
```

Once the clusters of the data observations are calculated, let's plot the clusters and also, the category of each observation in the variable *Species* (contouring by colours every observation point).

```
plot(x[,1],x[,2],col=clusters,pch=19,xlab="Petal.Length",ylab="Petal.Width",main="3-Means")
points(x[,1],x[,2],col=species,pch=1,cex=1.5)
legend(0,1,legend=unique(species),col=c("black","red","green"),pch=1,cex=0.75)
legend(0,0.75,legend=paste("Cluster",unique(clusters)),col=c("black","red","green"),pch=19,
       cex=0.75)
```



In conclusion, we can observe that the clusters created by the *K-Means* algorithm, using the features *Petal.Length* and *Petal.Width*, are quite similar to the distribution of the variable *Species*. It seems that, except some observation with different point and contour colour, the cluster 1 corresponds to the category *setosa*, the cluster 2 corresponds to the category *versicolor*, and the cluster 3 corresponds to the category *virginica*.

Note: Remember that the *KMeans* algorithm only uses the features *Petal.Length* and *Petal.Width*. Once the clusters are computed, we analyze them comparing with the variable *Species*, since it is an *unsupervised machine learning* technique that just explores data instead of predicting a target variable.