Program – 9

AIM: Write a program to implement clustering in R programming.

Introduction and Theory

Clustering

Clustering is the task of dividing the population or data points into a number of groups such that data points in the same groups are more similar to other data points in the same group than those in other groups. In simple words, the aim is to segregate groups with similar traits and assign them into clusters.

Broadly speaking, clustering can be divided into two subgroups :

- **Hard Clustering:** In hard clustering, each data point either belongs to a cluster completely or not. For example, in the above example each customer is put into one group out of the 10 groups.
- **Soft Clustering**: In soft clustering, instead of putting each data point into a separate cluster, a probability or likelihood of that data point to be in those clusters is assigned. For example, from the above scenario each costumer is assigned a probability to be in either of 10 clusters of the retail store.
-

There are four ways we can do clustering, of which centroid based methods are most commonly used.

- **Connectivity models:** As the name suggests, these models are based on the notion that the data points closer in data space exhibit more similarity to each other than the data points lying farther away. These models can follow two approaches. In the first approach, they start with classifying all data points into separate clusters & then aggregating them as the distance decreases. In the second approach, all data points are classified as a single cluster and then partitioned as the distance increases. Also, the choice of distance function is subjective. These models are very easy to interpret but lacks scalability for handling big datasets. Examples of these models are hierarchical clustering algorithm and its variants.
- **Centroid models:** These are iterative clustering algorithms in which the notion of similarity is derived by the closeness of a data point to the centroid of the clusters. K-Means clustering algorithm is a popular algorithm that falls into this category. In these models, the no. of clusters required at the end have to be mentioned beforehand, which makes it important to have prior knowledge of the dataset. These models run iteratively to find the local optima.
- **Distribution models:** These clustering models are based on the notion of how probable is it that all data points in the cluster belong to the same distribution (For example: Normal, Gaussian). These models often suffer from overfitting. A popular example of these models is Expectation-maximization algorithm which uses multivariate normal distributions.
- **Density Models:** These models search the data space for areas of varied density of data points in the data space. It isolates various different density regions and assign the data points within these regions in the same cluster. Popular examples of density models are DBSCAN and OPTICS.

Program – 9

Applications of Clustering

- Recommendation engines
- Market segmentation
- Social network analysis
- Search result grouping
- Medical imaging
- Image segmentation
- Anomaly detection

k-means clustering

It is a method of vector quantization, originally from signal processing, that is popular for cluster analysis in data mining. k-means clustering aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster. This results in a partitioning of the data space into Voronoi cells.

The problem is computationally difficult (NP-hard); however, efficient heuristic algorithms converge quickly to a local optimum. These are usually similar to the expectation-maximization algorithm for mixtures of Gaussian distributions via an iterative refinement approach employed by both k-means and Gaussian mixture modeling. They both use cluster centers to model the data; however, k-means clustering tends to find clusters of comparable spatial extent, while the expectation-maximization mechanism allows clusters to have different shapes.

The algorithm has a loose relationship to the k-nearest neighbor classifier, a popular machine learning technique for classification that is often confused with k-means due to the name. Applying the 1-nearest neighbor classifier to the cluster centers obtained by k-means classifies new data into the existing clusters. This is known as nearest centroid classifier or Rocchio algorithm.

To process the learning data, the K-means algorithm in data mining starts with a first group of randomly selected centroids, which are used as the beginning points for every cluster, and then performs iterative (repetitive) calculations to optimize the positions of the centroids

It halts creating and optimizing clusters when either:

- The centroids have stabilized—there is no change in their values because the clustering has been successful.
- The defined number of iterations has been achieved.

```
Initialize k means with random values

For a given number of iterations:
    Iterate through items:
        Find the mean closest to the item
        Assign item to mean
        Update mean
```

The k-mean pseudo-code

Program – 9

Code

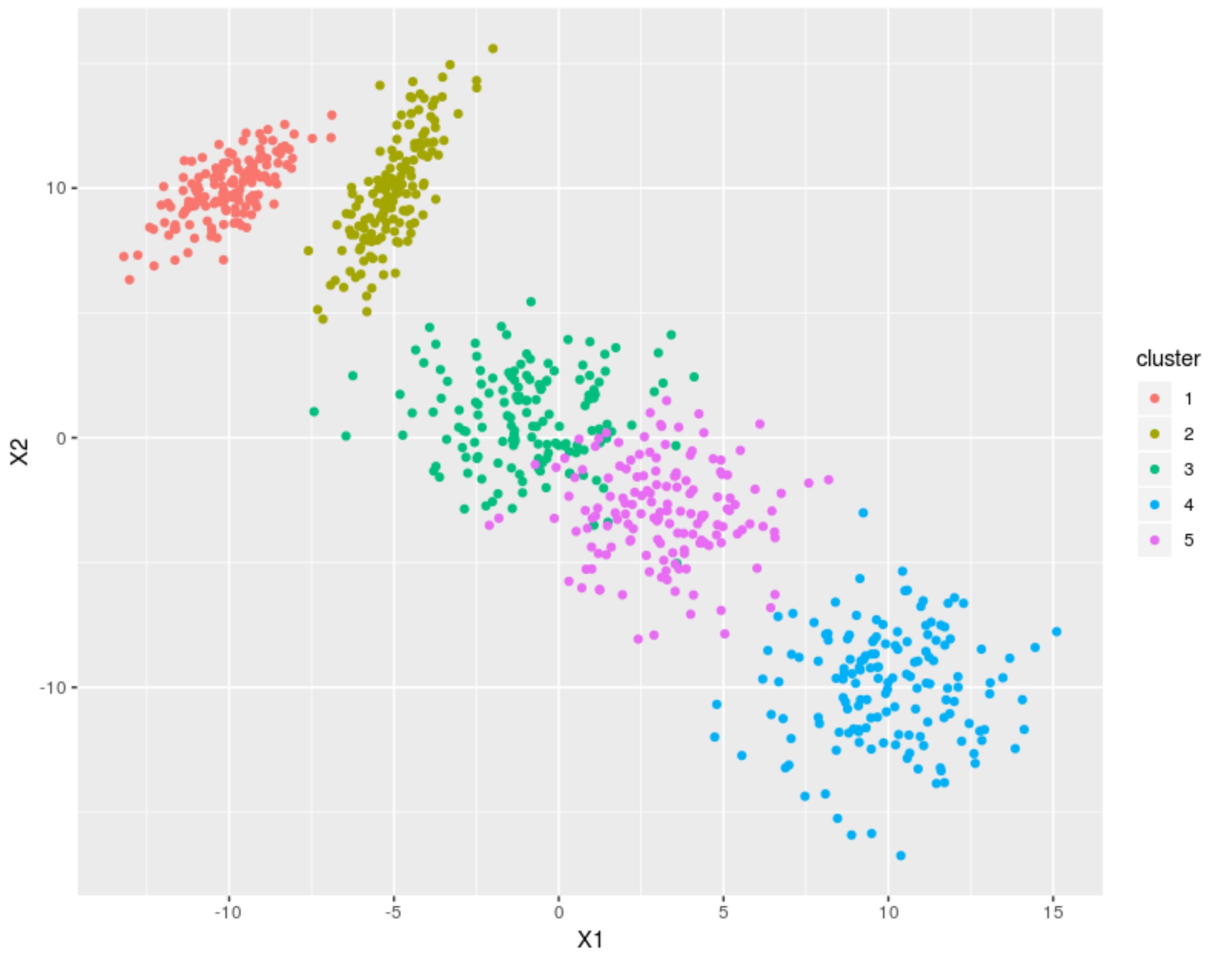
Clustering.R

```r
require(ggplot2)
require(MASS)
set.seed(42)
set1 = MASS::mvrnorm(n = 150, c(-10,10), matrix(c(1.5,1,1,1.5),2))
set2 = MASS::mvrnorm(n = 150, c(-5,10), matrix(c(1,2,2,6),2))
set3 = MASS::mvrnorm(n = 150, c(-1,1), matrix(c(4,0,0,4),2))
set4 = MASS::mvrnorm(n = 150, c(10,-10), matrix(c(4,0,0,4),2))
set5 = MASS::mvrnorm(n = 150, c(3,-3), matrix(c(4,0,0,4),2))
DF = 
data.frame(rbind(set1,set2,set3,set4,set5),cluster=as.factor(c(rep(1:5,each=150))))
ggplot(DF,aes(x=X1,y=X2,color=cluster))+geom_point()

kmeans = function(data,K=4,stop_crit=10e-3)
{
  #Initialisation of clusters
  centroids = data[sample.int(nrow(data),K),]
  current_stop_crit = 1000
  cluster = rep(0,nrow(data))
  converged = F
  it = 1
  while(current_stop_crit>=stop_crit & converged==F)
  {
    it=it+1
    if (current_stop_crit<=stop_crit)
    {
      converged=T
    }
    old_centroids=centroids
    ##Assigning each point to a centroid
    for (i in 1:nrow(data))
    {
      min_dist=10e10
      for (centroid in 1:nrow(centroids))
      {
        distance_to_centroid=sum((centroids[centroid,]-data[i,])^2)
        if (distance_to_centroid<=min_dist)
        {
          cluster[i]=centroid
          min_dist=distance_to_centroid
        }
      }
    }
    ##Assigning each point to a centroid
    for (i in 1:nrow(centroids))
    {
      centroids[i,]=apply(data[cluster==i,],2,mean)
    }
```

Program – 9

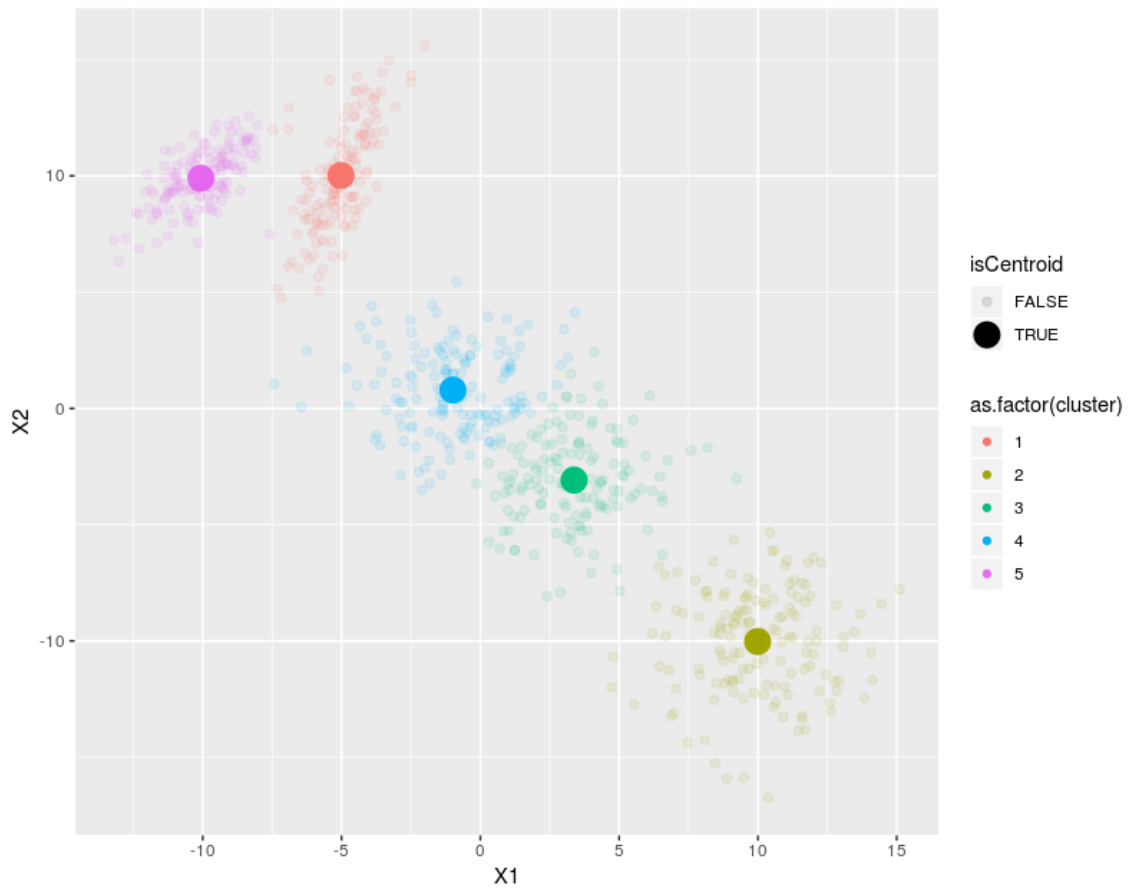
```
    current_stop_crit=mean((old_centroids-centroids)^2)
  }
  return(list(data=data.frame(data,cluster),centroids=centroids))
}

res<-kmeans(DF[1:2],K=5)
res$centroids$cluster=1:5
res$data$isCentroid=F
res$centroids$isCentroid=T
data_plot=rbind(res$centroids,res$data)
ggplot(data_plot,aes(x=X1,y=X2,color=as.factor(cluster),size=isCentroid,alpha
=isCentroid))+geom_point()
```

Results & Outputs



Sample data

Program – 9



Final cluster assignment

Findings and Learnings:

1. K-means is one of the simplest unsupervised learning algorithms that solve the well-known clustering problem.
2. R provides easy to use tools for performing cluster analysis.
3. We have successfully implemented clustering in R