Experiment No 4

Title: Clustering Algorithms & Evaluation in R

Theory

Clustering algorithms are a part of unsupervised machine learning algorithms. Why unsupervised ? Because, the target variable is not present. The model is trained based on given input variables which attempt to discover intrinsic groups (or clusters).

For example, in healthcare, a hospital might cluster patients based on their tumor size so that patients with different tumor sizes can be treated differently.

This technique helps us organize unstructured data. It can be used on tabular data, images, text data, etc.

Types of Clustering Techniques

There are many types of clustering algorithms, such as K means, fuzzy c- means, hierarchical clustering, etc. Other than these, several other methods have emerged which are used only for specific data sets or types (categorical, binary, numeric).

Among these different clustering algorithms, there exists clustering behaviors known as

1. **Soft Clustering:** In this technique, the probability or likelihood of an observation being partitioned into a cluster is calculated.
2. **Hard Clustering:** In hard clustering, an observation is partitioned into exactly one cluster (no probability is calculated).

These clustering techniques use distance measures to decide the similarity or dissimilarity in the observations.

It follows a simple rule: the closer the observations, the more similar they are, and vice versa.

Distance Calculation for Clustering

There are some important things to keep in mind:

1. With quantitative variables, distance calculations are highly influenced by variable units and magnitude. For example, clustering variable height (in feet) with salary (in rupees) having different units and distribution (skewed) will invariably return biased results. Hence, always make sure to standardize (mean = 0, sd = 1) the variables. Standardization results in unit-less variables.
2. Use of a particular distance measure depends on the variable types; i.e., formula for calculating distance between numerical variables is different than categorical variables.

Distances measures are as follows:

1. Euclidean Distance **2. Manhattan Distance** **3. Hamming Distance**

4. Gower Distance **5. Cosine Similarity**

K means Clustering

This technique partitions the data set into unique homogeneous clusters whose observations are similar to each other but different than other clusters. The resultant clusters remain mutually exclusive, i.e., non-overlapping clusters.

In this technique, "K" refers to the number of cluster among which we wish to partition the data. Every cluster has a centroid. The name "k means" is derived from the fact that cluster centroids are computed as the mean distance of observations assigned to each cluster.

This k value k is given by the user. It's a hard clustering technique, i.e., one observation gets classified into exactly one cluster.

Technically, the k means technique tries to classify the observations into K clusters such that the total within cluster variation is as small as possible. Let C denote a cluster. k denotes the number of clusters. So, we try to minimize:

$$\text{minimum}(C_1..C_K)\{\sum W(C_K)\}$$

But, how do we calculate within cluster variation? The most commonly used method is **squared Euclidean distance**. In simple words, it is the sum of squared Euclidean distance between observations in a cluster divided by the number of observations in a cluster (shown below):

$$W(C_k) = \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2$$

$|C_K|$ denotes the number of observations in a cluster.

Now, we fairly understand what goes behind k means clustering. But how does it start ? Practically, it takes the following steps:

1. Let's say the value of k = 2. At first, the clustering technique will assign two centroids randomly in the set of observations.
2. Then, it will start partitioning the observations based on their distance from the centroids. Observations relatively closer to any centroid will get partitioned accordingly.
3. Then, based on the number of iterations (after how many times we want the algorithm to converge) we've given, the cluster centroids will get recentered in every iteration. With this, the algorithm will try to continually optimize for the lowest within cluster variation.
4. Based on the newly assigned centroids, assign each observation falling closest to the new centroids.
5. This process continues until the cluster centers do not change or the stopping criterion is reached.

How to select best value of k in k means?

Determining the best value of k plays a critical role in k means model performance. To select best value of k, there is no "one rule applies all situation." It depends on the shape and distribution of the observations in a data set.

Intuitively, by finding the best value of k, we try to find a balance between the number of clusters and the average variation within a cluster.

Following are the methods useful to find an optimal k value:

1. Cross Validation **2. Elbow Method**

3. Silhouette Method **4. X means Clustering**

Hierarchical Clustering

In simple words, hierarchical clustering tries to create a sequence of nested clusters to explore deeper insights from the data. For example, this technique is being popularly used to explore the standard plant taxonomy which would classify plants by family, genus, species, and so on.

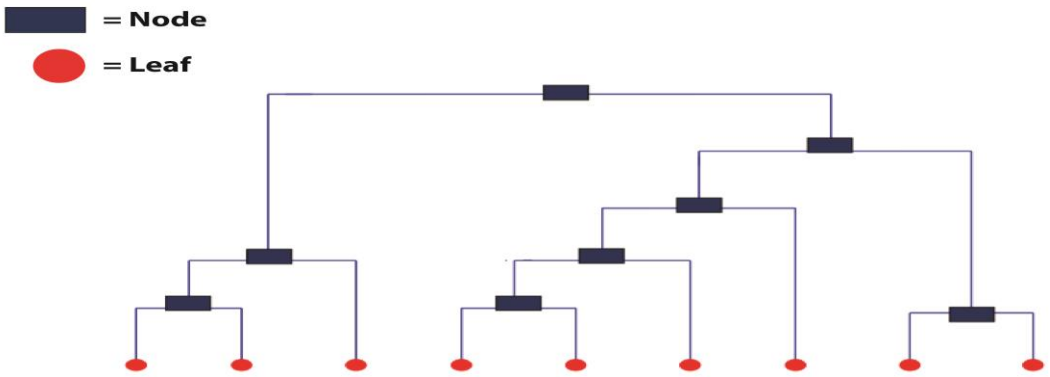
Hierarchical clustering technique is of two types:

1. Agglomerative Clustering – It starts with treating every observation as a cluster. Then, it merges the most similar observations into a new cluster. This process continues until all the observations are merged into one cluster. It uses a bottoms-up approach (think of an inverted tree).

2. Divisive Clustering – In this technique, initially all the observations are partitioned into one cluster (irrespective of their similarities). Then, the cluster splits into two sub-clusters carrying similar observations. These sub-clusters are intrinsically homogeneous. Then, we continue to split the clusters until the leaf cluster contains exactly one observation. It uses a top-down approach.

Agglomerative clustering

This technique creates a hierarchy (in a recursive fashion) to partition the data set into clusters. This partitioning is done in a bottoms-up fashion. This hierarchy of clusters is graphically presented using a Dendogram (shown below).



This dendrogram shows how clusters are merged / split hierarchically. Each node in the tree is a cluster. And, each leaf of the tree is a singleton cluster (cluster with one observation). So, how do we find out the optimal number of clusters from a dendrogram?

Let's understand how to study a dendrogram.

As you know, every leaf in the dendrogram carries one observation. As we move up the leaves, the leaf observations begin to merge into nodes (carrying observations which are similar to each other). As we move further up, these nodes again merge further.

Always remember, lower the merging happens (towards the bottom of the tree), more similar the observations will be. Higher the merging happens (toward the top of the tree), less similar the observations will be.

To determine clusters, we make **horizontal cuts** across the branches of the dendrogram. The number of clusters is then calculated by the number of vertical lines on the dendrogram, which lies under horizontal line. As seen above, the horizontal line cuts the dendrogram into three clusters since it surpasses three vertical lines. In a way, the

selection of height to make a horizontal cut is similar to finding k in k means since it also controls the number of clusters.

But, how to decide where to cut a dendrogram? Select the cut with a better accuracy.

The advantage of using hierarchical clustering over k means is, it doesn't require advanced knowledge of number of clusters. However, some of the advantages which k means has over hierarchical clustering are as follows:

- It uses less memory and converges faster.
- Unlike hierarchical, k means doesn't get trapped in mistakes made on a previous level. It improves iteratively.
- k means is non-deterministic in nature, i.e.. after every time you initialize, it will produce different clusters. On the contrary, hierarchical clustering is deterministic.

Note: K means is preferred when the data is numeric. Hierarchical clustering is preferred when the data is categorical.

What are the evaluation methods used in cluster analysis ?

The methods to evaluate clustering accuracy can be divided into two broad categories:

Internal Accuracy Measures*:* As the name suggests, these measures calculate the cluster's accuracy based on the compactness of a cluster. Following are the methods which fall under this category:

1. Sum of Squared Errors (SSE) **2. Scatter Criteria**

External Accuracy Measures : These measures are calculated by matching the structure of the clusters with some pre-defined classification of instances in the data. Let's look at these measures:

1. **Rand Index** 2. **Precision Recall Measure**

R Program

```
require("datasets")
data("iris") # load Iris Dataset
str(iris) #view structure of dataset
summary(iris) #view statistical summary of dataset
head(iris) #view top  rows of dataset
iris.new<- iris[,c(1,2,3,4)]
iris.class<- iris[,"Species"]
head(iris.new)
head(iris.class)
normalize <- function(x){
  return ((x-min(x))/(max(x)-min(x)))
}
iris.new$Sepal.Length<- normalize(iris.new$Sepal.Length)
iris.new$Sepal.Width<- normalize(iris.new$Sepal.Width)
iris.new$Petal.Length<- normalize(iris.new$Petal.Length)
iris.new$Petal.Width<- normalize(iris.new$Petal.Width)
head(iris.new)
result<- kmeans(iris.new,3)
result$size # gives no. of records in each cluste
result$centers
result$cluster
table(result$cluster,iris.class)
```

```
par(mfrow=c(2,2), mar=c(5,4,2,2))

plot(iris.new[c(1,2)],

col=result$cluster

plot(iris.new[c(1,2)], col=iris.class

plot(iris.new[c(3,4)], col=result$cluster

plot(iris.new[c(3,4)], col=iris.class)

tunek <- kmeansruns(scaled_wd,krange = 1:10,criterion = "ch")

tunek$bestk

> tunekw <- kmeansruns(scaled_wd,krange = 1:10,criterion = "asw") > tunekw$bestk
```

Output of the program and analysis

Load and view dataset

```
require("datasets")

data("iris") # load Iris Dataset

str(iris) #view structure of dataset

## 'data.frame':    150 obs. of  5 variables:

##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...

##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...

##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...

##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...

##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...

summary(iris) #view statistical summary of dataset

##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
```

```
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
##  Median :5.800   Median :3.000   Median :4.350   Median :1.300
##  Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
##  3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
##  Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
##        Species
##  setosa    :50
##  versicolor:50
##  virginica :50
```

head(iris) #view top rows of dataset

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

Preprocess the dataset

Since clustering is a type of Unsupervised Learning, we would not require Class Label(output) during execution of our algorithm. We will, therefore, remove Class Attribute "Species" and store it in another variable. We would then normalize the attributes between 0 and 1 using our own function.

iris.new<- iris[,c(1,2,3,4)]

iris.class<- iris[,"Species"]

head(iris.new)

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1          5.1         3.5          1.4         0.2
```

2 4.9 3.0 1.4 0.2

3 4.7 3.2 1.3 0.2

4 4.6 3.1 1.5 0.2

5 5.0 3.6 1.4 0.2

6 5.4 3.9 1.7 0.4

head(iris.class)

[1] setosa setosa setosa setosa setosa setosa

Levels: setosa versicolor virginica

```
normalize <- function(x){
 return ((x-min(x))/(max(x)-min(x)))
}
```

```
iris.new$Sepal.Length<- normalize(iris.new$Sepal.Length)
iris.new$Sepal.Width<- normalize(iris.new$Sepal.Width)
iris.new$Petal.Length<- normalize(iris.new$Petal.Length)
iris.new$Petal.Width<- normalize(iris.new$Petal.Width)
```

head(iris.new)

Sepal.Length Sepal.Width Petal.Length Petal.Width

1 0.22222222 0.6250000 0.06779661 0.04166667

2 0.16666667 0.4166667 0.06779661 0.04166667

3 0.11111111 0.5000000 0.05084746 0.04166667

4 0.08333333 0.4583333 0.08474576 0.04166667

5 0.19444444 0.6666667 0.06779661 0.04166667

6 0.30555556 0.7916667 0.11864407 0.12500000

Apply k-means clustering algorithm

result<- kmeans(iris.new,3) #aplly k-means algorithm with no. of centroids(k)=3

result$size # gives no. of records in each cluster

[1] 61 39 50

result$centers # gives value of cluster center datapoint value(3 centers for k=3)

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1    0.4412568   0.3073770   0.57571548  0.54918033
## 2    0.7072650   0.4508547   0.79704476  0.82478632
## 3    0.1961111   0.5950000   0.07830508  0.06083333
```

result$cluster #gives cluster vector showing the custer where each record falls

```
##   [1] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
##  [36] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [71] 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 2 2
## [106] 2 1 2 2 2 2 2 2 1 2 2 2 2 2 1 2 1 2 1 2 2 1 1 2 2 2 2 2 1 1 2 2 2 2 1 2
## [141] 2 2 1 2 2 2 1 2 2 1
```

table(result$cluster,iris.class)

```
##    iris.class
##     setosa versicolor virginica
##   1      0         47        14
##   2      0          3        36
##   3     50          0         0
```

Result of table shows that Cluster 1 corresponds to Virginica, Cluster 2 corresponds to Versicolor and Cluster 3 to Setosa.

Total number of correctly classified instances are: 36 + 47 + 50= 133

Total number of incorrectly classified instances are: 3 + 14= 17

Accuracy = 133/(133+17) = 0.88 i.e our model has achieved 88% accuracy!