Modeling in Revolution R Enterprise Module 9: K-Means Clustering

Revolution Analytics

Introduction to Clustering

Clustering allows us to group data based on similarity between observations with respect to certain variables. Clustering is the general name for any of a large number of classification techniques that involve assigning observations to membership in one of two or more clusters on the basis of some distance metric.

- We can use clustering to compare and contrast different groups.
- Further, we can use clustering to break up big data into more manageable chunks.

Introduction to Clustering

K-means clustering assigns each observation membership to one of k clusters in such a way that minimizes the distance between each observation and its cluster's mean.

1. Place K points in the space defined by the data being clustered. These points are the initial group centroids.
2. Assign each observation to the group that has the closest centroid.
3. When all observations have been assigned, recalculate the positions of the K centroids as centers of the clusters resulting from the previous step.
4. Repeat steps 2 and 3 until the centroids no longer move. Points are moved from cluster to another so as to minimize sums of squares

REVOLUTION ANALYTICS

Introduction to Clustering

We can perform distributed K-means in ScaleR by the function rxKmeans, even when the data is stored on the HDFS.

The function call requires a formula and data, in addition to the user specifying the number of clusters desired. Note that the initial centers for clusters are chosen at random):

```
rxKmeans(formula, data, numClusters, ...)
```

Example: Clustering with K-Means

Let's use the K-means method on a subset of variables in the Bank data set. In this example, let's consider age and balance once again, and specify the number of clusters, numClusters, as equal to three. Note that some of the output has been suppressed:

```
infile <- file.path("data", "BankXDF.xdf")
BankDS <- RxXdfData(file = infile)


Kmeans <- rxKmeans(~duration + balance, data = BankDS, numClusters = 3)


## Rows Read: 10000, Total Rows Processed: 10000, Total Chunk Time: 0.002 seconds
## Rows Read: 10000, Total Rows Processed: 20000, Total Chunk Time: 0.001 seconds
## Rows Read: 10000, Total Rows Processed: 30000, Total Chunk Time: 0.001 seconds
## Rows Read: 10000, Total Rows Processed: 40000, Total Chunk Time: 0.001 seconds
## Rows Read: 5211, Total Rows Processed: 45211, Total Chunk Time: Less than .001 seconds
## Rows Read: 10000, Total Rows Processed: 10000, Total Chunk Time: 0.002 seconds
...
```

Example: Clustering with K-Means

Considering the rest of the output…

```
Kmeans



## Call:
## rxKmeans(formula = ~duration + balance, data = BankDS, numClusters = 3)
##
## Data: BankDS
## Number of valid observations: 45211
## Number of missing observations: 0
...
```

Example: Interpretation

All observations of both variables have been divided into three distinct categories, as defined in the function call:

- Cluster 1 - duration Mean: 255.9709, balance Mean: 25501.3851
- Cluster 2 - duration Mean: 256.2109, balance Mean: 643.8847
- Cluster 3 - duration Mean: 276.5923, balance Mean: 6365.6297

Finally, the within cluster sum of squares (i.e. distance from the mean) for each cluster is:

- Cluster 1: 32360806640
- Cluster 2: 47748356032
- Cluster 3: 32526119735

Example: Interpretation

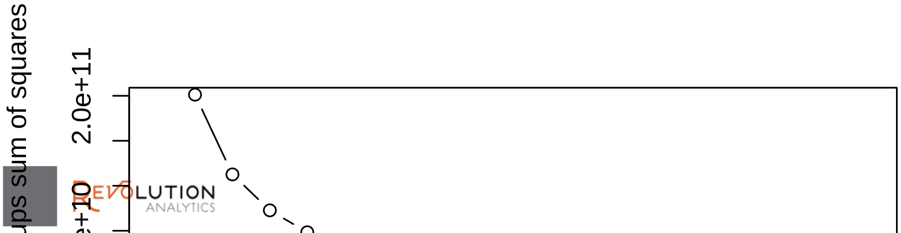
Note that requiring additional clusters naturally decreases the sum of squares, since multiple means would imply shorter distances to other observations.

```
SS <- rep(NA, 20)
for (i in 2:20) SS[i] <- rxKmeans(~duration + balance, data = BankDS, numClusters = i)$tot.withi
plot(1:20, SS, type = "b", xlab = "Number of Clusters", ylab = "Within groups sum of squares")
```

Example: Interpretation

Below we plot the Total Sum of Squares as a function of number of Clusters (from 2 to 20).

```
## Rows Read: 10000, Total Rows Processed: 10000, Total Chunk Time: 0.001 seconds
## Rows Read: 10000, Total Rows Processed: 20000, Total Chunk Time: 0.001 seconds
## Rows Read: 10000, Total Rows Processed: 30000, Total Chunk Time: 0.001 seconds
## Rows Read: 10000, Total Rows Processed: 40000, Total Chunk Time: 0.001 seconds
## Rows Read: 5211, Total Rows Processed: 45211, Total Chunk Time: Less than .001 seconds
## Rows Read: 10000, Total Rows Processed: 10000, Total Chunk Time: 0.001 seconds
...
```



Exercise: Clustering with K-Means

In this exercise, construct three clusters using K-Means, considering variables:

- age
- day

Recap

Let’s review some of the concepts covered in this module:

- What is clustering?
- How does K-Means solve a clustering problem?

REVOLUTION ANALYTICS

Thank you

Revolution Analytics is the leading commercial provider of software and support for the popular open source R statistics language.

www.revolutionanalytics.com, 1.855.GET.REVO, Twitter: @Revo-lutionR