Unsupervised Learning using K-Means Clustering

Shivang Dave, Ronald Barahona

University of Houston - Clear Lake

Abstract

When it comes to learning from data, there may an instance where your dataset may not have classification or categorization included in the observations. The method of learning from such 'unlabeled' datasets is called Unsupervised learning. There are various ways to approach unsupervised learning such as: Clustering, Anomaly detection, Neural Networks etc. Clustering is the task of grouping a set of objects in such a way that objects in the same cluster are more similar to each other than to those in other clusters. It is widely used for statistical data analysis.

1. K-Means Clustering

K-Means is one of the simplest unsupervised learning algorithm for clustering of datasets. It allows classification of a given dataset through a certain number (k) of clusters. Interesting aspect of using k-means clustering is that it informs you about data rather than you informing it.

1.1. ***Algorithm***

```
1. Input: k, set of points $x_1$ ... $x_n$
2. Place random centroids $c_1$ ... $c_k$ at random location
3. Repeat until convergence:
        for each point $x_i$:
            find nearest centroid $c_j$ by euclidean distance = argmin D($x_i$,$c_j$)
            assign the point $x_i$ to cluster j
        for each cluster j = 1 ... k: $c_j(a) = \frac{1}{n_j} \sum_{x_j \to c_j} x_j(a)\ \forall a = 1...d$
            centroid $c_j$ = mean of all points $x_i$ assigned to j
4. End; when cluster assignments for data points don't change
```

1.2. ***How does it work?***

To get started with the algorithm it requires following two parameters; number of clusters k and dataset. Once required parameters have been provided, it places initial c_k centroids on different locations. The main idea is to define k centers, one for each cluster. These centers should be placed in a cunning way because of different location causes different result. Hence, the better choice is to place them as much as possible far away from each other.

Next step is to assign each data point to the closest cluster depending on the euclidean distance between each data point and each centroid of respective clusters. Once these points are assigned to a cluster, a new centroid for each cluster is calculated. Mean value of all the data points in each cluster becomes new centroid(s).

$$D(x_i, c_j) = \sqrt{\sum_{n=1}^{N}(x_n - c(x_n))^2}$$

Above steps are repeated until the algorithm converges. Convergence can be defined in various ways depending on the application. Usually if cluster assignments for data points doesn't change after repetition then it converges. You can also predefine maximum number of iterations to force convergence in an environment with restricted resources.

1.3. ***Error Function***

Error function to define in-sample error is fairly simple. E_{in} can be defined as:

$$E_j = \sum_{x_n \in S_j} ||x_n - c_j||^2$$

$$E_{in}(S_1, ..., S_k; c_1, ..., c_k) = \sum_{j=1}^{k} E_j$$

$$E_{in} = \sum_{n=1}^{N} ||x_n - c(x_n)||^2$$

S = Collection of data points, E_j = Error in cluster j, c(x_n) = Centroid of the cluster to which x_n belongs.

2. Implementation

Our goal was to implement the algorithm in a way that it would work for any data set with minimal changes to the code (if any). Not only that but we also wanted to make sure that our model would predict something. Without class labels in the data set to cross validate, training the model for any kind of prediction was challenging. We decided to use nearest neighbor approach to predict the cluster for testing data points.

2.1. ***Dependencies:***

- Python 2.7
- numpy (used for various array operations)
- sklearn (used for splitting data into train and test sets)
- matplotlib (used for plotting the results)

Note: we tried not to use methods from numpy and wrote methods from scratch. But due to structural difference between numpy arrays and arrays we used, we decided to use numpy methods to complement numpy arrays.

2.2. ***How to run:***

- Move to the project directory in terminal and type following:

  python main.py
- Enter the file name and hit return:

  python main.py

  Enter filename with its extension: small_data.txt
- Enter number of clusters you want to create as:

  python main.py

  Enter filename with its extension: small_data.txt

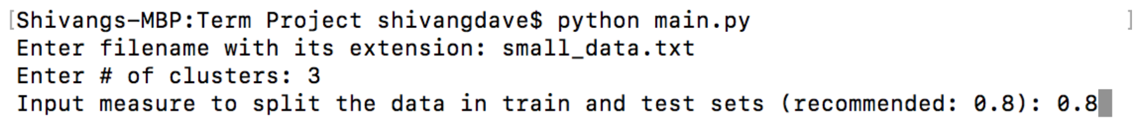
  Enter # of clusters: 3
- Program will ask for % measure for splitting data:

  python main.py

  Enter filename with its extension: small_data.txt

  Enter # of clusters: 3

  Input measure to split the data in train and test sets (recommended: 0.8): 0.8



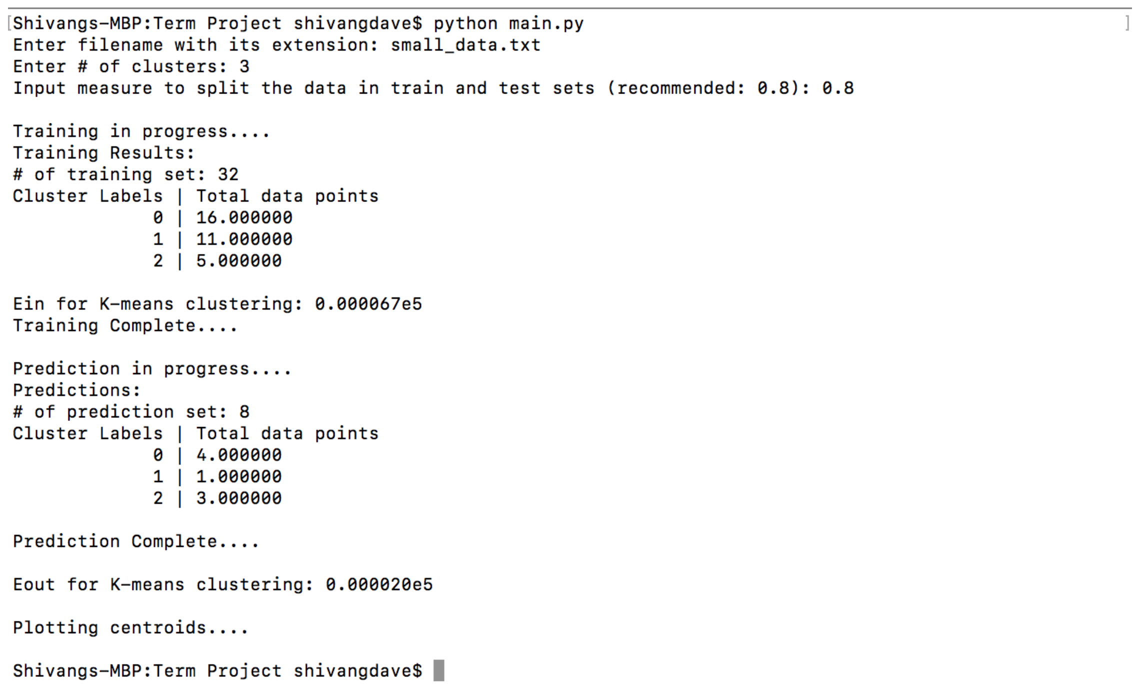
```
Shivangs-MBP:Term Project shivangdave$ python main.py
Enter filename with its extension: small_data.txt
Enter # of clusters: 3
Input measure to split the data in train and test sets (recommended: 0.8): 0.8
```

Figure 1: Input

2.3. ***Results:***

Once all the inputs have been validated, algorithm starts training the model. As soon as the training finishes, it returns an array of centroids, labels for training data points (cluster labels) and in-sample error E_{in} using error function stated above. It prints out training results and this also means that our model is ready for predictions.

As mentioned earlier without any class labels in the data set to cross validate, training a classifier for clustering is complicated. Thus, we are treating array of centroids returned



```
Shivangs-MBP:Term Project shivangdave$ python main.py
Enter filename with its extension: small_data.txt
Enter # of clusters: 3
Input measure to split the data in train and test sets (recommended: 0.8): 0.8

Training in progress....
Training Results:
# of training set: 32
Cluster Labels | Total data points
             0 | 16.000000
             1 | 11.000000
             2 | 5.000000

Ein for K-means clustering: 0.000067e5
Training Complete....

Prediction in progress....
Predictions:
# of prediction set: 8
Cluster Labels | Total data points
             0 | 4.000000
             1 | 1.000000
             2 | 3.000000

Prediction Complete....

Eout for K-means clustering: 0.000020e5

Plotting centroids....

Shivangs-MBP:Term Project shivangdave$ 
```

Figure 2: Output

by our training method as a classifier to predict clusters for out-of sample data points i.e. testing data points.

Once our classifier is ready, program calculates the distance between our classifier and testing data point. At the end, it returns predictions in form of an array of labels for testing data points and prints it.

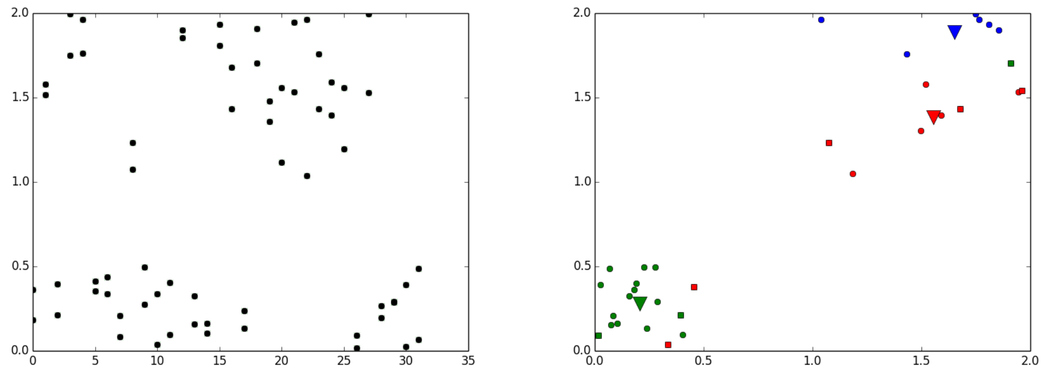


Figure 3: Graph

3. Summary:

K-means clustering algorithm largely depends on initial values for centroids. It can be observed by rerunning the program with same values for k for the same data set. K-Means clustering generates a specific number of disjoint and flat clusters. Hence, it is well suited for numerical datasets. And with large number of features, k-means is one of the fastest options in terms of computation.