Title: Report of Implementing K-means algorithm

Professor: Dr. Yuan

Name: Kaikai Bian

Date: 10/01/2016

K-means clustering algorithm is a partitional clustering algorithm. It identifies different partitions of the space for a fixed number of clusters.

Firstly, provide an integer k specifying the number of clusters.

Secondly, initialize the k cluster centroids:

* Select k data points from the data set at random
* Select k data points from the space at random

Then, for each point in the data set, assign it to the closest cluster center. We can calculate the distance between the data point and each centroid, and then choose the centroid with shortest distance. We also need to update each centroid based on the data points that are assigned to it. We compute the mean of all points’ individual feature, which are assigned to a specific centroid.

Lastly, if any data point has changed its cluster centroid, we repeat this algorithm.

Firstly, I used the built-in function nextInt() of Random class to generate K integers which represent the indices of centroids, even though they are pseudorandom numbers. I stored the information of all data points in a 2-D array, and hence I can easily use the indices to retrieve the information of centroids.

Since there are 210 data points in the file, I choose the 1st, 2nd, and 3rd. data point. It is more likely to be the worst case. Nevertheless, the result shows that these data points still converge to some centroids. The features of these centroids are:

18.722, 16.297, 0.885, 6.209, 3.723, 3.604, 6.066

14.819, 14.537, 0.881, 5.591, 3.299, 2.707, 5.218

11.989, 13.284, 0.853, 5.227, 2.880, 4.584, 5.074

I think the most important detail of my implementation is the data structure that how to store the features of these data points and centroids. I used a number of 2-D arrays to store them. After the program runs, the centroids will not be from the data set and can be in the middle of data points that are assigned to it. Hence, we need to calculate the features of the new centroids, and then based on these centroids, find out the data points that are close to them.

The starting centroids of the following 5 sets are randomly picked:

1:

Starting Centroids:

12.36, 13.19, 0.8923, 5.076, 3.042, 3.22, 4.605,

18.76, 16.2, 0.8984, 6.172, 3.796, 3.12, 6.053,

12.44, 13.59, 0.8462, 5.319, 2.897, 4.924, 5.27,

Final Centroid:

14.648, 14.460, 0.879, 5.564, 3.278, 2.649, 5.192,

18.722, 16.297, 0.885, 6.209, 3.723, 3.604, 6.066,

11.964, 13.275, 0.852, 5.229, 2.873, 4.760, 5.089,

IV = 313.217

EV = 27309.340

IV/EV = 0.0114692188

2:

Starting Centroids:

11.27, 12.86, 0.8563, 5.091, 2.804, 3.985, 5.001,

13.07, 13.92, 0.848, 5.472, 2.994, 5.304, 5.395,

16.44, 15.25, 0.888, 5.884, 3.505, 1.969, 5.533,

Final Centroid:

11.964, 13.275, 0.852, 5.229, 2.873, 4.760, 5.089,

14.648, 14.460, 0.879, 5.564, 3.278, 2.649, 5.192,

18.722, 16.297, 0.885, 6.209, 3.723, 3.604, 6.066,

IV = 313.217

EV = 27309.340

IV/EV = 0.0114692188

3.

Starting Centroids:

20.1, 16.99, 0.8746, 6.581, 3.785, 1.955, 6.449,

17.55, 15.66, 0.8991, 5.791, 3.69, 5.366, 5.661,

12.79, 13.53, 0.8786, 5.224, 3.054, 5.483, 4.958,

Final Centroid:

18.722, 16.297, 0.885, 6.209, 3.723, 3.604, 6.066,

14.819, 14.537, 0.881, 5.591, 3.299, 2.707, 5.218,

11.989, 13.284, 0.853, 5.227, 2.880, 4.584, 5.074,

IV = 313.734

EV = 27255.565

IV/EV = 0.0115108330

4.

Starting Centroids:

13.37, 13.78, 0.8849, 5.32, 3.128, 4.67, 5.091,

12.21, 13.47, 0.8453, 5.357, 2.893, 1.661, 5.178,

19.15, 16.45, 0.889, 6.245, 3.815, 3.084, 6.185,

Final Centroid:

11.964, 13.275, 0.852, 5.229, 2.873, 4.760, 5.089,

14.648, 14.460, 0.879, 5.564, 3.278, 2.649, 5.192,

18.722, 16.297, 0.885, 6.209, 3.723, 3.604, 6.066,

IV = 313.217

EV = 27309.340

IV/EV = 0.0114692188

5.

Starting Centroids:

18.45, 16.12, 0.8921, 6.107, 3.769, 2.235, 5.794,

12.76, 13.38, 0.8964, 5.073, 3.155, 2.828, 4.83,

12.21, 13.47, 0.8453, 5.357, 2.893, 1.661, 5.178,

Final Centroid:

18.722, 16.297, 0.885, 6.209, 3.723, 3.604, 6.066,

11.964, 13.275, 0.852, 5.229, 2.873, 4.760, 5.089,

14.648, 14.460, 0.879, 5.564, 3.278, 2.649, 5.192,

IV = 313.217

EV = 27309.340

IV/EV = 0.0114692188

The starting centroids of the following sets are picked manually:

Starting Centroids:

15.26, 14.84, 0.871, 5.763, 3.312, 2.221, 5.22,

14.88, 14.57, 0.8811, 5.554, 3.333, 1.018, 4.956,

14.29, 14.09, 0.905, 5.291, 3.337, 2.699, 4.825,

Final Centroid:

18.722, 16.297, 0.885, 6.209, 3.723, 3.604, 6.066,

14.819, 14.537, 0.881, 5.591, 3.299, 2.707, 5.218,

11.989, 13.284, 0.853, 5.227, 2.880, 4.584, 5.074,

IV = 313.734

EV = 27255.565

IV/EV = 0.0115108330

Basically, we evaluate the results by IV/EV, which means minimize IV while maximizing EV. Through these 6 sets of results, we can find that the fifth set leads to the best result since IV/EV is the smallest. That is because two of its data points have comparatively smaller features such as area, perimeter.

Another finding is that for one centroid, it is always the optimal, and its features are (18.722, 16.297, 0.885, 6.209, 3.723, 3.604, 6.066); for the other two, there are two choices for them, and they can be

(14.819, 14.537, 0.881, 5.591, 3.299, 2.707, 5.218)

(11.989, 13.284, 0.853, 5.227, 2.880, 4.584, 5.074)

or

(14.648, 14.460, 0.879, 5.564, 3.278, 2.649, 5.192)

(11.964, 13.275, 0.852, 5.229, 2.873, 4.760, 5.089).

And the latter pair is less likely to appear.